Une approche sémantique pour la réutilisation et l'adaptation de donnée 3D

Ioan Marius Bilasco

N° attribué par la bibliothèque
|__|__|__|__|__|__|__|__|__|__|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER DE GRENOBLE

Spécialité : Informatique

préparée au Laboratoire Informatique de Grenoble
dans le cadre de l'École Doctorale
Mathématiques, Informatique, Sciences et Technologie de l'Information

présentée et soutenue publiquement

par

Ioan Marius BILASCO

le 19 décembre 2007

Une approche sémantique pour la réutilisation et l'adaptation de données 3D

Directeur de Thèse : Hervé Martin

JURY

Président	M. Jacques Le Maître
Rapporteur	M. Claude Chrisment
Rapporteur	M. Mohand-Saïd Hacid
Directeur de thèse	M. Hervé Martin
Co-encadrant	Mme Marlène Villanova-Oliver

Je tiens à remercier :

Monsieur Jacques LE MAITRE, Professeur à l'Université du Sud Toulon-Var, qui a accepté de présider le jury et d'examiner mon travail.

Monsieur Claude CHRISMENT, Professeur à l'Université Paul Sabatier de Toulouse, et Monsieur Mohand-Saïd HACID, Professeur à l'Université Claude Bernard de Lyon, qui m'ont fait l'honneur de rapporter mon travail et dont les remarques m'ont permis d'en améliorer certains aspects.

Monsieur Hervé MARTIN, Professeur à l'Université Joseph Fourier de Grenoble, mon directeur de thèse, pour son encadrement, ses nombreux conseils et son soutien. La confiance qu'il m'a accordée m'a permis de m'épanouir en tant que jeune chercheur et de mener à bon port cette thèse.

Madame Marlène VILLANOVA-OLIVER, Maître de Conférence à l'Université Pierre Mendès France de Grenoble, ma co-encadrante, pour toute sa coopération et toutes les discussions qui ont beaucoup contribué à l'avancement de mon travail de thèse.

Je remercie également à tous ceux qui m'ont soutenu et encouragé tout au long de ces années de thèse :

Les permanents de l'équipe STEAMER : Jérôme GENSEL, Paule-Annick DAVOINE, Ahmet LBATH, Danielle ZIEBELIN. Une pensée particulière pour Jérôme qui est celui qui a conduit mes pas vers l'équipe STEAMER (anciennement SIGMA).

Mes anciens professeurs de l'Université Babes Bolyai de Cluj-Napoca : Florin BOIAN, Ionut LAZAR, Bazil PARV, Ioan CHIOREAN, Doina TATAR, qui ont su ouvrir mon appétit pour la recherche et ont rendu possible mon arrivée à Grenoble, auquel titre je remercie également Alain LECOMTE de m'avoir accueilli dans le cadre du programme ERASMUS à l'Université Pierre Mendès France pour ma première année d'étude en France.

Le personnel administratif et technique qui entoure notre équipe : Carol, Pascale, Martine, Christiane, François, Gilles.

Mes collègues de l'ESISAR de Valence et nottament Jean-Luc qui m'a accompagné lors de mes premiers pas faits dans l'enseignement lors de mon monitorat.

Mes collègues de l'IUT1 : Alain, Cyrille, Denis, Emil, Frank, Yves, Jean-Marc, Nechar, Manon, Olivier, Patrick, Pierre, Philippe, Zouhir et notre chère Marie-Pierre pour leur chaleureux accueil au sein du département RT.

Mes collègues d'équipe : Angela, Aurélie, Bogdan, Carlos, Céline, Christine, Dia, Edicarsia, José, Laurent, Manuele, Raphaël, Olivier, Samira, Sandro, Thierry, Windson avec qui j'ai pu partager des moments de travail et de nombreuses pauses café.

Mes collègues du laboratoire et notamment : Corbier, Fred, Justi, Mohammad, Nico, Sorin, Vincent, Vianney, Yan avec qui j'ai passé d'agréables moments de détente.

Mes amis roumains et grenoblois que je n'ai pas mentionnés ci-dessus mais dont la compagnie et le soutien n'ont pas été les moindres.

Ma belle famille qui a su m'accueillir à bras ouverts et qui m'a permis de me sentir ici comme chez moi.

Mes parents, mes grands parents et mon frère qui malgré leur éloignement géographique ont su être toujours près de moi.

En dernier, je tiens à remercier tout particulièrement ma femme, Céline, qui a su m'offrir tout le réconfort, la compréhension et l'amour dont j'ai eu besoin pour mener à bout ce travail.

Marius

Table des Matières

Introduction

Etat de l'art

Proposition

Table des Illustrations

Figures

Tableaux

Introduction

I Contexte

L'évolution rapide de la puissance de calcul des ordinateurs laisse présager une large utilisation du 3D qui s'impose comme un média encore plus riche que les images, le son ou la vidéo.

Les premières approches vers la troisième dimension ont consisté à ajouter une hauteur sur chacun des points d'une forme en 2D. Ce type d'approche est connue sous le nom de 2,5D car elle se trouve à mi-chemin entre le 2D et le 3D. La principale limitation de ce type de démarche est de ne permettre qu'une seule altitude z en tout point de la scène géographique modélisée, c'est-à-dire pour tout couple (x,y) du plan. Le 3D pur est décrit comme une collection de formes dont les points, les surfaces et les volumes sont tous définis par rapport aux trois dimensions de l'espace.

Le 3D rend possible la création, la modification ou l'exécution de simulations dans un monde virtuel d'une manière plus réaliste qu'une représentation linéaire ou 2D de l'information. De nombreuses applications issues des domaines de l'aménagement du territoire, des transports, des télécommunications, de la défense, de l'architecture ou du tourisme pourront bénéficier de l'apport informationnel fourni par des scènes 3D réalistes qui, également, donnent lieu à des interactions plus complexes (visualisation, exploration, navigation) entre utilisateurs et applications.

L'expérience 3D peut être rendue encore plus intéressante si l'apport informationnel fourni par la scène s'enrichit avec des données sémantiques relatives aux objets présentées dans la scène. Au même titre que la description de la géométrie et de l'apparence, la sémantique peut faire partie intégrante des documents modélisant les scènes 3D comme illustré dans [Polys *et al.*, 2004][Hetherington *et al.*, 2004][Cruz *et al.*, 2005][Pittarello *et al.*, 2005], etc. Ces propositions ont en commun l'utilisation de X3D [Web3D, 2004], le standard de représentation de scènes 3D résultant des efforts du Consortium Web3D[1]. Le X3D a été crée afin de pérenniser et faciliter le déploiement du contenu 3D à travers le Web. À présent, le standard vise à assurer le transfert et le partage de scènes 3D à travers le Web entre différents acteurs, en maximisant le réalisme de la scène indépendamment de la nature de la scène. Le standard permet également d'associer, pour un domaine d'application spécifique, des métadonnées utilisables par des scripts qui gèrent l'interactivité de la scène.

[1] http://www.web3d.org

Les métadonnées, au-delà de leur rôle de vecteur d'information utilisable par des scripts dans le cadre d'une application 3D, peuvent également faciliter la gestion des données 3D à différents moments de leur exploitation (recherche, réutilisation, déploiement…). Par exemple, lors du déploiement de la scène, la sémantique contenue dans les métadonnées peut servir à effectuer certaines transformations sur la scène, à des fins d'adaptation essentiellement. Ces transformations peuvent trouver plusieurs motivations. Ainsi, exclure des objets d'une scène peut être envisagé pour permettre à l'utilisateur d'assimiler au mieux les informations pertinentes (on exclut les objets non pertinents afin de limiter la surcharge cognitive). Une exclusion d'objets, ou leur dégradation, peut aussi s'avérer utile dans le cas de dispositifs aux caractéristiques matérielles contraignant le déploiement de la scène originale. Ces deux exemples illustrent en quoi les métadonnées servent une adaptation des contenus 3D, à l'utilisateur et au dispositif respectivement. En amont du déploiement des objets 3D exploitant les métadonnées, nous nous intéressons à la conception de scènes 3D par une réutilisation d'objets 3D exploitant les métadonnées et le sens qu'elles confèrent à ces objets. La réutilisation de données 3D et de leur sémantique dans différentes scènes et applications est donc au cœur de cette thèse.

II Problématique

L'engouement de l'industrie au tour des données 3D (U3D [3DIF, 2006], XAML [MacVittie, 2006] ou 3D XML [Versprille, 2005]) laisse présumer de l'importance d'une meilleure gestion du contenu 3D déjà existant. La recherche d'un contenu 3D suivant un certain nombre de critères, la réutilisation de contenus 3D, la personnalisation de scènes 3D (autant sur les aspects géométriques que ceux liés à la navigation et l'interaction), l'intégration de contenus 3D au sein des applications multimédia sont, dans notre vision, les thèmes prioritaires de recherches dans les années à venir dans le domaine de l'information 3D. Chacun de ces thèmes a déjà été abordé dans le monde multimédia pour d'autres types de données (texte, son, image, vidéo). Toutefois, la richesse de l'information 3D nécessite un investissement particulier et des méthodes adaptées et spécifiques de travail.

II.1. Réutilisation

La réutilisation de contenus 3D est primordiale car la construction des objets 3D requiert beaucoup de temps et un investissement conséquent. Les concepteurs disposent aujourd'hui d'environnements tels que *Maya* par *Alias|wavefront*[2] ou *3D Studio Max* par *Discreet*[3] qui permettent de créer des objets 3D avec un haut niveau de réalisme. Cependant, les possibilités de réutilisation de scènes 3D au sein de différentes applications restent à ce jour limitées, et ce en raison d'une non dissociation quasi systématique de la sémantique et de la géométrie. Afin d'illustrer les manques actuels et les défis à relever, nous considérons deux types de réutilisation.

En premier lieu, la réutilisation peut correspondre au processus qui extrait un objet 3D d'une scène existante et l'inclut tel quel dans une nouvelle scène. Par exemple, le même objet 3D modélisant un bâtiment peut être utilisé dans plusieurs scènes. Il s'agit du type de réutilisation le plus répandu. Les critères sur lesquels se basent la recherche et la réutilisation sont généralement simples et l'acte de réutilisation consiste à référencer l'adresse physique du fichier contenant l'objet réutilisable.

En second lieu, on considère la réutilisation d'un même modèle 3D dans différentes applications, ou tout du moins, à des fins différentes. Selon l'application, des informations

[2] http://www.alias.com
[3] http://www.autodesk.com

sémantiques (métadonnées) différentes peuvent être associées au même objet 3D. Ainsi, alors que dans une application d'aménagement urbain on associe à un bâtiment (objet 3D) le nombre de personnes pouvant y vivre, dans une application d'évaluation du risque sismique, ce même objet se verra associer des informations sur sa vulnérabilité.

L'obtention de l'objet à réutiliser repose, soit sur une recherche à base de mots-clés, soit sur une recherche par exemple, comme dans les travaux de [Del Bimbo *et al.*, 1998][Kim *et al.*, 2004][Ohbuci *et al.*, 2003][Zaharia *et al.*, 2002]. Dans ces propositions, les modèles sont indexés de façon automatique. Des requêtes formulées au moyen d'exemples sont utilisées pour retrouver les objets indexés similaires, mais similaires du point de vue de leur géométrie et/ou de leur apparence uniquement. Ces solutions ne s'appuient en effet que sur l'indexation de descriptions liées à la géométrie et à l'apparence des modèles 3D et ne prennent pas en compte des critères sémantiques. L'absence de l'utilisation de critères sémantiques dans ce type d'approche est dûe à la complexité inhérente au processus d'extraction automatique de connaissances à partir d'un document multimédia [Chang, 2002]. Cependant, un certain nombre d'approches semi-automatiques résumées dans [Hobson *et al.*, 2006] laissent entrevoir des solutions dans ce sens.

II.2. Adaptation

Bien que la construction d'une scène consomme beaucoup de temps et de ressources, le contenu 3D est habituellement conçu pour un dispositif spécifique dans un contexte spécifique et n'est pas facilement réutilisable sur d'autres dispositifs et dans d'autres contextes. Récemment, les catalogues 3D en ligne, tels que 3DWareHouse[4] de Google Sketchup ou Teapotters[5], proposent aux concepteurs des méthodes de recherche à base de mots-clés et libellés. Mais, la façon selon laquelle l'information est disponible (type d'encodage, taille du document, etc.) ne correspond pas toujours aux possibilités du dispositif d'accès ou aux intérêts de l'utilisateur. Les scènes qui ont été conçues pour des dispositifs de grande taille (ordinateur de bureau, etc.), nécessitent une transformation préalable pour éviter les problèmes qui apparaîtraient à leur visualisation sur de petits dispositifs (PDA, téléphone, etc.). Des scènes couvrant de très grands espaces physiques et/ou qui contiennent une grande masse d'information (comme la version 3D de Paris disponible sur les Pages Jaunes) ne conviennent pas aux utilisateurs seulement intéressés par une petite partie de l'espace (par exemple, les Champs Elysées). Un processus d'adaptation devrait être mis en place entre l'entité fournissant les données et les dispositifs d'accès qui les consomment.

L'adaptation des données 3D est relativement différente de l'adaptation de documents multimédia 2D. Tandis que dans un document multimédia 2D le nom et les propriétés de chaque ressource média impliquée sont généralement connus et introduits de manière explicite dans le document par l'auteur, dans un document 3D nous devons traiter de l'information principalement géométrique qui est éparpillée dans tout le document. À notre connaissance, il n'existe aucune approche qui détermine automatiquement les objets (réels) et les primitives géométriques correspondant aux objets dans le document 3D. Pour cette raison, les moteurs d'adaptation existant considèrent la scène dans l'ensemble et ils appliquent des transformations uniformes (par exemple des dégradations géométriques ou de texture) à tous les éléments de la scène. Par conséquent, le même procédé d'adaptation est appliqué à un arbre, à un bâtiment ou à une surface 3D (par exemple un modèle numérique de terrain) contenus dans la même scène 3D.

Le procédé d'adaptation devrait combiner différentes techniques comme la dégradation géométrique, le filtrage sémantique ou la substitution d'un objet par son équivalent moins

[4] http://sketchup.google.com/3dwarehouse
[5] http://www.3dvia.com/home.php

complexe afin d'obtenir des résultats plus précis et adéquats. La capacité de combiner plusieurs techniques d'adaptation dans la même scène est cruciale parce que tous les objets ne peuvent pas être adaptés de manière uniforme. Par exemple, une technique d'adaptation conçue pour la réduction d'un Modèle Numérique de Terrain (MNT) dégradera excessivement un objet représentant un bâtiment inclus dans la même scène que le MNT.

III Aperçu de la proposition

III.1. Modèle sémantique de caractérisation de données 3D

Dans ce contexte, notre objectif est d'offrir une solution pour une meilleure gestion des données 3D et de la sémantique que l'on peut leur associer. Notre solution offre une gestion flexible de la sémantique qui est maintenue de façon indépendante de la géométrie et de l'apparence des données 3D. Cette séparation des informations permet par ailleurs aux concepteurs, lors de la création d'une application 3D (par réutilisation ou non d'objets existants), d'ajouter de la sémantique aux scènes 3D.

Une première étape consiste à localiser des objets sujets aux futures adaptations. Un tel objet est défini par plusieurs primitives géométriques qui, considérées ensemble, correspondent à une entité du monde réel. Plusieurs approches [Lorenz *et al.*, 2006][Pittarello *et al.*, 2006] portent sur la localisation des objets ou des régions de l'espace. Nous explorons l'utilisation des repères structuraux et/ou spatiaux pour identifier les objets éligibles pour la réutilisation ou l'adaptation.

Nous anticipons la réutilisation et l'adaptation par l'ajout de l'information sémantique liée aux objets et aux régions. Cette information, qui constitue les bases d'un processus de classification, peut être extraite directement à partir des métadonnées des objets dans les scènes 3D ou associée plus tard au contenu 3D au moyen d'annotations. Dans les deux cas, les informations peuvent être organisées dans des profils sémantiques liés aux domaines spécifiques d'application.

III.2. Réutilisation

Nous proposons une plate-forme de réutilisation d'objets 3D en nous appuyant sur des critères portant sur l'ensemble des caractéristiques d'un objet : la géométrie, l'apparence, la sémantique. Une approche de génération dynamique de scènes enrichies d'informations sémantiques (selon les besoins de l'application) est adoptée par cette plate-forme. La plate-forme vise à donner aux utilisateurs (logiciels ou matériels) la possibilité d'indiquer les catégories d'objets qu'ils souhaitent récupérer ainsi que l'information sémantique qu'ils veulent associer à chaque catégorie d'objets.

Afin de rendre ce processus de réutilisation indépendant des différents contextes de diffusion, qui ne cessent de se diversifier, nous proposons également des outils et formalismes permettant la mise en place d'un processus d'adaptation.

III.3. Adaptation

Dans cette thèse, nous décrivons une solution pour une prise en compte de l'adaptation au niveau des modèles 3D. La contribution principale est la construction d'un cadre d'adaptation basé sur des règles d'adaptation formulées sur des critères sémantiques.

Associer individuellement à chaque objet ou région d'une scène, une technique d'adaptation spécifique exige des efforts significatifs. Cependant, à l'intérieur de la même scène 3D, chaque catégorie d'objets 3D est généralement adaptée de manière semblable (par exemple, tous les arbres sont dégradés uniformément). En s'appuyant sur la notion de catégorie d'objets sémantique, nous formulons des règles logiques d'adaptation ou des stratégies complexes d'adaptation en rapport avec la sémantique d'un domaine d'application spécifique, sans avoir forcement des connaissances sur les langages 3D. Une règle d'adaptation est construite en deux parties : la première représente un critère de choix et la seconde correspond au type d'adaptation qui s'applique aux objets qui se conforme aux critères de sélection. Les règles d'adaptation présentées ici concernent les transformations appliquées à une scène avant son déploiement. Les techniques d'optimisation (par exemple, niveaux de détail, mip-mapping [Williams, 1983], etc.) sont orthogonales à notre proposition. Une stratégie d'adaptation correspond à un ensemble de règles d'adaptation ordonnées.

IV Plan de la thèse

L'état de l'art est composé de quatre chapitres qui concernent respectivement : la modélisation des scènes 3D, la sémantique dans les scènes 3D, la réutilisation et finalement l'adaptation de données 3D. Le but de cette première partie du mémoire et de mettre en exergue les principaux caractéristiques des données 3D, d'analyser les méthodes d'exploitation de ces données et de souligner les manques dans le domaine de la gestion des données 3D notamment en ce qui concerne la réutilisation et l'adaptation.

Le premier chapitre de l'état de l'art consacré à la modélisation des scènes 3D offre un aperçu général des principales notions impliquées dans la construction d'une scène 3D : le graphe de scène, la géométrie, l'apparence attachées aux données 3D. Une étude des principales modalités de définition d'une scène 3D est également réalisée en détaillant notamment les approches documentaires de type XML dont la structure ouvre des perspectives intéressantes pour la mise en œuvre des traitements évolués applicables (réutilisation et adaptation) aux documents 3D.

Le deuxième chapitre recueille les propositions qui s'intéressent à l'intégration et à l'exploitation de la sémantique relative aux données 3D. Premièrement, nous passons en revue les principaux moyens de description de la sémantique dans un cadre générique à l'aide des standards tels que RDF, OWL ou spécifiques aux données multimédia MPEG-7. Nous étudions également les moyens utilisés pour représenter la sémantique relative aux scènes 3D, ainsi que les modalités de son exploitation dans un contexte applicatif autre que celui pour lequel les informations sémantiques ont été introduites.

Le troisième chapitre concerne l'étude des travaux qui s'intéressent à la recherche et à la réutilisation de contenu 3D. Nous survolons les différents types de critères (au niveau signal et au niveau contenu) utilisés dans le processus de recherche de contenus. Nous essayons d'identifier les moyens permettant de contrôler la manière dont la réutilisation est mise en place à travers les différents systèmes présentés.

Les propositions s'intéressant au déploiement adapté et à la personnalisation des scènes 3D en accord avec le contexte de diffusion, sont étudiées dans le quatrième chapitre. Ce chapitre introduit des notions telles que le contexte, le type d'adaptation, la stratégie d'adaptation. Un aperçu des techniques, définies dans le domaine des données multimédia 2D mais qui jouissent d'un important potentiel d'adaptation aux données 3D, est présenté avant de discuter des spécificités des adaptations 3D.

L'exposé de nos propositions vient ensuite. La Figure 9.1 donne un aperçu de la façon dont nos contributions s'organisent les unes par rapport aux autres.

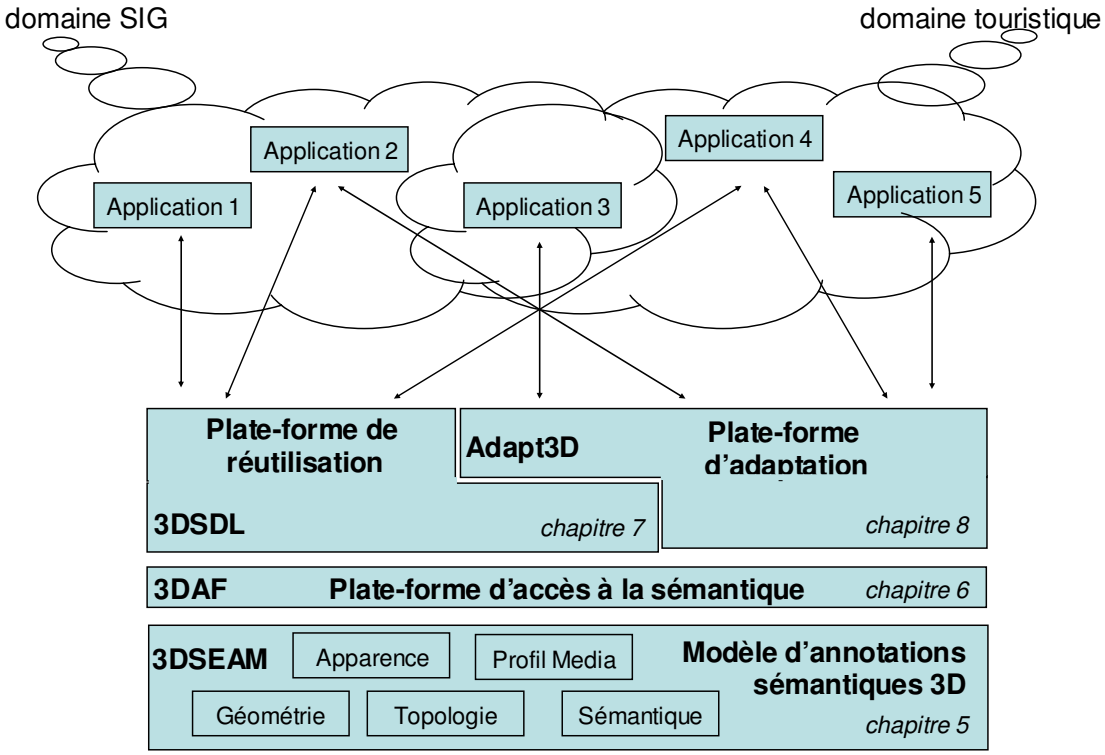


Figure 9.1 Aperçu de notre contribution.

Le cinquième chapitre est dédié à la présentation détaillée du modèle d'annotations sémantiques (*3D SEmantic Annotation Model* – 3DSEAM) que nous proposons dans le but de formaliser et stocker les informations sémantiques relatives aux données 3D. Le modèle couvre l'ensemble des dimensions d'une donnée : la géométrie, l'apparence, la topologie, le document, la sémantique relative à un domaine applicatif, la sémantique de la donnée au sein de la scène la contenant. La prise en compte de ces informations est essentielle pour enrichir les moyens d'expression des techniques de réutilisation et d'adaptation.

Dans le chapitre six nous proposons une plate-forme de gestion d'annotations (*3D Annotation Framework* – 3ADF) permettant d'extraire les informations stockées dans un entrepôt de descriptions contenant les instances du modèle 3DSEAM. Cette plate-forme introduit un niveau d'abstraction entre la représentation effective des connaissances et leur utilisation. Un langage d'interrogation 3DSEAM OQL étendant OQL [Cattel, 1994] et couvrant l'ensemble des dimensions du modèle est proposé afin d'offrir aux clients (logiciels ou utilisateurs experts) de la plate-forme un protocole commun d'accès aux caractéristiques.

Le chapitre sept est consacré à l'illustration d'une plate-forme de réutilisation permettant aux utilisateurs de générer des scènes 3D sémantiques (*3D Semantic Data Library* – voir Figure 9.1). Cette plate-forme repose sur les fonctionnalités de 3DAF permettant d'extraire les objets 3D et les informations pertinentes pour l'assemblage de scène 3D sémantiques.

Dans le chapitre huit nous décrivons un formalisme de description de stratégies d'adaptation au moyen de règles d'adaptation. Les critères, ainsi que les paramètres d'adaptation rentrant dans la formulation d'une règle, peuvent être définis en rapport avec l'ensemble des connaissances (géométrie, apparence, etc.) détenues sur les objets de l'entrepôt. Une plate-forme interprétant et mettant en œuvre l'ensemble des règles est décrite (Adapt3D – voir Figure 9.1).

L'implémentation des plates-formes 3DAF, 3DSDL et Adapt3D est en cours. Dans les chapitres qui leur sont consacrés, nous insistons sur l'architecture des plates-formes, sur les modèles de données qu'elles exploitent et sur les conventions de représentation choisies.

Le neuvième chapitre porte sur la validation expérimentale des fonctionnalités retenues dans le cadre des plates-formes 3DAF, 3DSDL et Adapt3D, et d'un entrepôt de descriptions 3DSEAM construit autour du standard MPEG-7. Une version spécifique des modules d'ajout d'annotations et d'interrogation s'appuyant sur les spécificités du standard MPEG-7 [Martinez *et al.,* 2002] est développée. Afin de rendre l'ensemble des informations disponibles aux futurs clients à travers la plate-forme 3DAF, les requêtes génériques 3DSEAM OQL sont traduites en requêtes XQuery [Boag *et al.,* 2007] suivant les choix de représentation et d'organisation des instances du modèle 3DSEAM au sein de l'entrepôt de descriptions. Une application visant la génération adaptée de scènes représentant le campus universitaire de Grenoble est présentée.

Enfin le dixième chapitre dresse le bilan de cette thèse et présente les perspectives de ce travail autour de l'adaptation et la réutilisation de données 3D en considérant une approche sémantique.

Etat de l'art

Cette première partie de la thèse est dédiée à la présentation détaillée des principaux verrous conceptuels et technologiques relatifs à la réutilisation et l'adaptation adéquate de données 3D. Elle est organisée suivant quatre axes de recherche : le document 3D, la sémantique dans les documents 3D, la réutilisation des fragments d'un document 3D et l'adaptation d'un document 3D. Pour chaque axe de recherche, nous mettons en évidence ses principales caractéristiques et les manques qui orientent, par la suite, notre proposition.

L'axe consacré au document 3D présente les principales caractéristiques d'un document 3D, ainsi que les différentes modalités de représentation de la géométrie et de l'apparence d'un document 3D. Cette étude s'appuie sur l'analyse de standards et d'approches de description spécifiques au monde 3D.

La dimension sémantique du document 3D est traitée à part car, dans notre vision, elle est la clé permettant de mettre en œuvre des traitements adéquats portant sur le document (réutilisation, adaptation, …). Cette partie de l'état de l'art analyse également les propositions de représentation et d'analyse de la sémantique dans un contexte général (en utilisant des standards tels que OWL, RDF, etc.), ou bien multimédia (avec MPEG-7).

L'axe portant sur la réutilisation est organisé en deux parties : une première, consacrée à une analyse des critères de recherche existants, afin d'identifier les parties de contenus à réutiliser, et une deuxième, consacrée à une analyse des travaux mettant en œuvre des mécanismes de réutilisation.

L'étude de l'adaptation est également décomposée en plusieurs thématiques de recherche portant respectivement sur : les différents types de techniques d'adaptation, le niveau de généricité des techniques d'adaptation, les formalismes et les solutions architecturales les mettant en oeuvre.

Ainsi, nous montrons les acquis de la communauté 3D en ce qui concerne la représentation de la sémantique, la mise en œuvre des procédés de réutilisation et d'adaptation. Nous mettons également en exergue les barrières qui empêchent une démocratisation de procédés de réutilisation et d'adaptation. C'est par rapport à ces éléments que nous positionnons notre proposition et les solutions qu'elle apporte en termes de sémantique, réutilisation et adaptation.

1. Le document 3D

Dans ce premier chapitre de l'état de l'art nous présentons la notion de document 3D, les principales spécificités de celui-ci, des outils pour sa construction et un standard de représentation de données 3D : X3D [Web3D, 2004]. Le standard X3D que nous mettons au cœur de notre travail – héritier XML [Yergeau *et al.*, 2004] du langage VRML [Web3D, 1997] qui a permis dans les années 90 d'ouvrir le monde Web aux données 3D.

1.1. Du document multimédia au document 3D

Le terme document désigne un ensemble de données organisées suivant un certain nombre de règles [Roisin, 1999]. Ces règles servent à exploiter les données brutes contenues dans les documents et à leur associer une représentation visuelle, auditive ou autre. Lorsque la représentation d'un document regroupe plusieurs objets et comporte plusieurs types de modalités de diffusion (par exemple, l'utilisation simultanée du visuel et du son) on parle de *document multimédia* [Hardman *et al.*, 1993][Roisin, 1998][Chrisment *et al.*, 2000]. Lorsque le document contient également des éléments qui préconisent une visualisation en 3D de certaines données du document on parle de *documents 3D*. Cette définition est très générale et couvre tout document qui nécessite les trois dimensions de l'espace afin de pouvoir être visualisé de manière cohérente. Dans cette thèse nous étudions plus particulièrement les documents 3D qui définissent explicitement la géométrie 3D des objets.

Avant de présenter les caractéristiques générales d'un document 3D nous introduisons le terme de *scène 3D*. La *scène 3D* désigne l'espace construit pour assurer le rendu visuel d'un document 3D. Dans cet espace, l'utilisateur peut naviguer afin de visualiser une partie ou la totalité de l'information 3D contenue dans le document support. Généralement, les scènes 3D sont définies par leur géométrie et leur apparence. D'autres aspects, tels que la caractérisation de l'environnement (lumières, sons), peuvent enrichir la définition du document 3D. Lorsque la géométrie, l'apparence ou l'environnement d'une scène 3D sont susceptibles de varier dans le temps, on parle de scène 3D animée.

1.2. Caractéristiques générales d'un document 3D

Le document 3D est construit autour de la notion de **graphe de scène**. Ce graphe, acyclique et direct organise de manière flexible et extensible les différentes dimensions d'une scène. Nous détaillons les deux principales dimensions d'un document 3D : la géométrie et l'apparence. Une dernière partie de cette section donne un aperçu d'autres dimensions que l'on peut recontrer dans une scène 3D

1.2.1. Graphe de scène

La notion de **graphe de scène** a été proposée au début des années 80 afin d'accélérer le processus de visualisation de scènes 3D sur des ordinateurs d'assez faible puissance, disposant de cartes graphiques standard. Le rendu d'une scène 3D est obtenu en projetant les polygones 3D dans un espace 2D correspondant à la surface d'affichage du dispositif de visualisation. Plus le nombre de polygones à dessiner est important, plus le rendu est lent. Les caractéristiques du *graphe de scène* et son exploitation au sein de scènes 3D se trouvent toujours au centre des préoccupations de la communauté 3D [Strauss, 1999][Sowizral, 2000][Döllner *et al.*, 2002] [Naef *et al.*, 2003][Coehlo *et al.*, 2004][Reitmayr *et al.*, 2005]. Cet engouement autour de la notion de *graphe de scène* est justifié par le fait qu'il reste le principal moyen d'organiser des éléments au sein d'une scène 3D et d'optimiser le rendu visuel d'une scène.

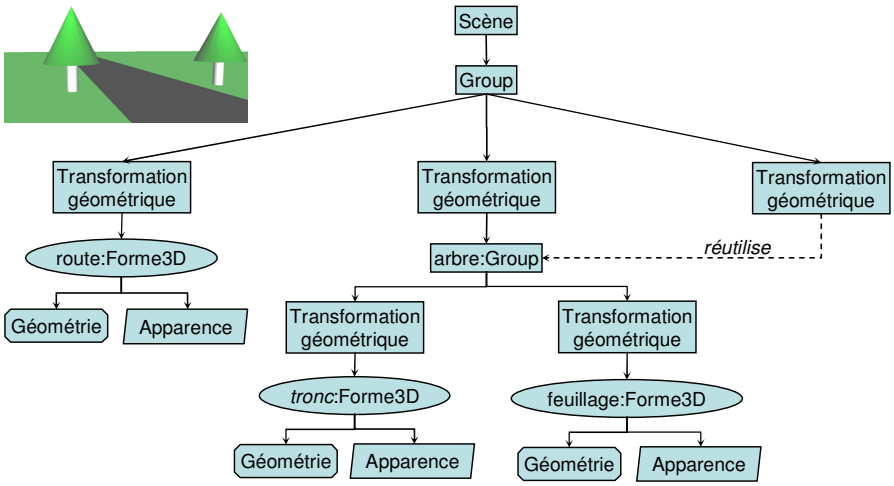


Figure 1.1 Exemple de graphe de scène.

Dans la Figure 1.1 nous illustrons un graphe de scène qui peut correspondre à la scène présentée en haut à gacuhe de la Figure 1.1. Chaque nœud de graphe remplit une fonction bien précise : la constitution de groupes d'objets, la définition de l'emplacement spatial, la définition de la géométrie des objets, la définition de l'apparence des objets, etc.

La fonctionnalité primaire du graphe est celle d'assurer le rendu optimal. Afin d'atteindre ce but, il faut effectuer uniquement le rendu des éléments géométriques potentiellement visibles. L'élagage d'éléments qui ne rentrent pas dans le champ de vision de l'utilisateur, précède le processus de projection. Plus l'élagage est efficace et sélectif, plus le calcul des projections 2D et le rendu visuel sont rapides. Les éléments à élaguer sont identifiés par différents calculs de visibilité (par exemple, la technique *ray tracing* [Levoy *et al.*, 1990]) appliqués à chaque élément. Les éléments élagués sont ceux situés en dehors du champ de vision de l'utilisateur et ceux qui sont complètement cachés par d'autres éléments visibles dans la scène. Lorsqu'aucune structure (hiérarchique ou autre) ne régit l'organisation des éléments constituant la géométrie de

la scène, chaque élément doit être pris en compte pour effectuer les calculs de visibilité. Pour limiter le nombre d'éléments à prendre en compte lors des calculs de visibilité, les éléments sont regroupés en éléments composites suivant la décision du concepteur de la scène. Ce regroupement doit être fait par rapport à la proximité des objets au sein de la scène afin que cela soit efficace. À leur tour, les éléments composites peuvent faire partie d'autres éléments composés. Ainsi, les éléments d'une scène sont organisés selon une structure hiérarchique. Afin de faciliter la réutilisation des éléments (par exemple, des arbres ayant les mêmes caractéristiques) à travers plusieurs branches de la structure hiérarchique, il est possible que plusieurs éléments composites partagent les mêmes sous-élémentss.

Tous les éléments (simples ou composites) sont caractérisés par **la boîte englobante** (*bounding box - bbox*) qui correspond à la plus petite boîte incluant spatialement tous les éléments le constituant. Les dimensions et la position de cette boîte sont utilisées lors de calculs de visibilité en vue d'élagage. Si la boîte ne se trouve pas dans le champ de vision, tous les éléments qu'elle contient sont élagués. Les éléments y étant inclus sont forcément hors du champ de vision. Ainsi, suivant le nombre d'éléments contenus dans chaque élément composé, le processus d'élagage peut se révéler très rapide.

Même si initialement le concept de graphe de scène a été conçu pour accélérer le rendu d'une scène 3D, il est devenu au cours du temps un moyen standard de description de modèles 3D grâce à sa capacité d'isoler tant la description de différentes caractéristiques de la scène, que l'organisation spatiale en diverses régions au sein de la scène.

Dans les sections suivantes, nous étudions en détail les principales dimensions descriptives d'un document 3D : la géométrie et l'apparence. Une dernière partie donne un aperçu d'autres dimensions que l'on peut retrouver dans les scènes 3D.

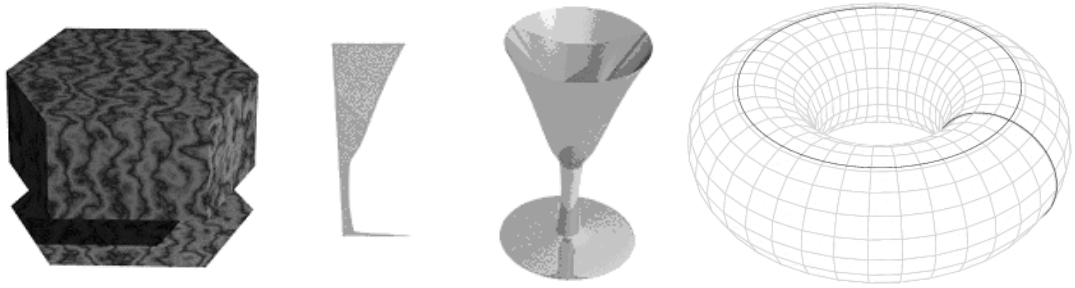
1.2.2. Géométrie

La **géométrie** est la dimension fondamentale de toute document 3D. Elle constitue le squelette de la scène. Les autres dimensions (apparence, lumières, etc.) viennent s'y ajouter afin d'accroître le niveau de réalisme de la scène.

Suivant la manière dont les éléments géométriques sont décrits, nous pouvons isoler trois grandes familles de description géométrique : la description **par balayage**, la description **surfacique** et la description **volumique**.

Description par balayage

La description par balayage permet de définir des objets 3D en utilisant une surface 2D. Cette surface suit une translation (voir Figure 1.2a), une rotation (voir Figure 1.2b) ou bien une trajectoire quelconque (voir Figure 1.2c).



a) balayage par translation b) balayage par rotation c) balayage d'une trajectoire

Figure 1.2 Construction d'objets 3D par différentes techniques de balayage d'après [Ramos, 2003].

L'espace balayé par la surface 2D en suivant la trajectoire, définit l'objet 3D. Dans la Figure 1.2a, la forme de base est l'hexagone qui se trouve en bas de l'objet 3D, et la trajectoire correspond à la hauteur de l'objet 3D. Dans la Figure 1.2b, la surface 2D correspond au polygone à gauche de la figure et la trajectoire correspond à un cercle. Dans la Figure 1.2c la forme de base est représentée par le cercle vertical situé dans la partie droite de la figure. La trajectoire correspond au cercle dessiné au-dessus de l'objet.

Description surfacique

La description surfacique correspond à la définition d'objets géométriques à l'aide de surfaces polygonales. Il y a deux types de description surfacique : la description par *contours* et la description par *frontières*. La description par *contours*, également appelée *la technique du fil de fer*, correspond à la définition d'un objet à partir d'un ensemble de sommets et d'arêtes. Ce genre de construction est ambigu car au moment de la visualisation il est difficile de distinguer les parties pleines des contours des parties vides. Ainsi, lorsque l'on regarde le cube dans la partie gauche de la Figure 1.3, on ne peut pas savoir à quel cube de la partie droite cette représentation filaire correspond.

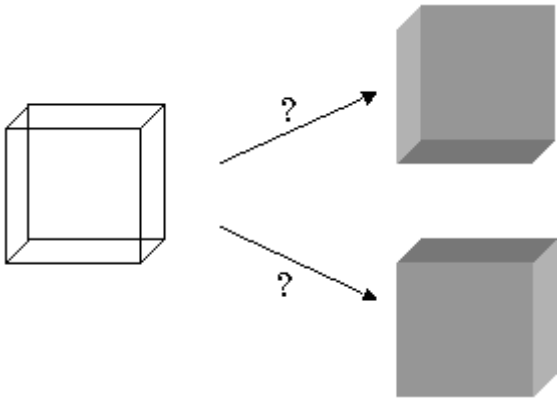


Figure 1.3 Ambiguïté de la représentation de type fil de fer d'après [Ramos, 2003].

Une alternative est la description par *enveloppes ou frontières*. Dans la littérature on rencontre aussi le terme *B-Rep* (de l'anglais *Boundary Representation*) pour ce type de description. Ce type de description introduit, en plus des arêtes et des sommets, la notion de facette. Ceci permet de délimiter des volumes et de faire la distinction entre l'intérieur et l'extérieur d'un objet. Les facettes peuvent être définies en tant que polygones, surfaces courbées, surfaces quadratiques, etc. Dans la Figure 1.4, nous illustrons la construction d'un cube au moyen de six polygones. Cela permet de construire des objets 3D plus complexes que la technique du *fil de fer*. Cependant, les deux techniques de description surfacique présentent l'inconvénient que chaque élément (facette, arête, sommet) doit être défini indépendamment des autres. Le document décrivant les objets est par conséquence de taille importante.

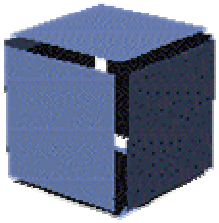


Figure 1.4 Représentation B-Rep d'un cube comportant 6 facettes.

Description volumique

La description volumique correspond à la technique la plus évoluée de construction d'objets 3D. Elles utilisent des éléments géométriques qui couvrent les trois dimensions de l'espace. Plusieurs techniques de construction d'objets 3D entrent dans cette catégorie.

La technique appelée *Regularized Boolean Set Operation* (RBSO) définit des objets complexes en appliquant une série d'opérations ensemblistes (union, différence, intersection) sur des volumes basiques. La Figure 1.5 illustre la construction de trois objets 3D en utilisant trois opérations ensemblistes supportées par RBSO.

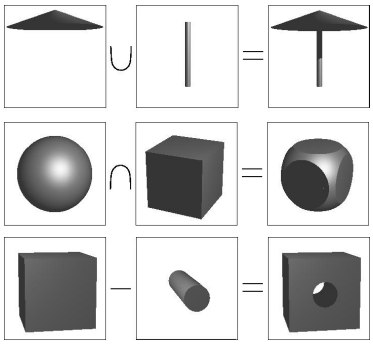


Figure 1.5 Construction d'objets 3D en utilisant la technique volumique RBSO.

Une autre technique de description volumique est appelée *Constructive Solid adaptation* (CSG). Dans le cadre de cette technique, le concepteur dispose d'un certain nombre de primitives bornées (cube, cylindre, sphère, etc.) et non bornées (demi-espace, plan, etc.). Les objets sont définis comme des graphes dont les nœuds représentent des opérateurs booléens ou des transformations géométriques (rotation, translation, mise à l'échelle, etc.). Les opérateurs illustrés dans la Figure 1.5 – union de formes, intersection de formes, différence de formes, décident de la forme des objets composites. Les transformations géométriques décident de la disposition spatiale des objets. Les feuilles correspondent aux primitives bornées ou non bornées. L'existence d'une structure aboutie et exploitable au niveau algorithmique introduit, par ailleurs, le problème de non unicité de la représentation d'un objet 3D. Plusieurs arbres CSG différents peuvent représenter la géométrie d'un même objet 3D.

La partition de l'espace en sous-régions, à différents niveaux de granularité, est à la base d'une troisième technique de représentation volumique de la géométrie 3D : *l'énumération spatiale*. L'objet 3D est considéré comme un solide décomposable en un ensemble de solides élémentaires voisins les uns des autres et qui ne s'intersectent pas. L'unité de base est le *voxel* qui correspond à un élément de volume unitaire. Un cas particulier d'application de cette technique est la représentation par *octree*. L'*octree* correspond à la décomposition de l'espace en utilisant des cubes (voir Figure 1.6).

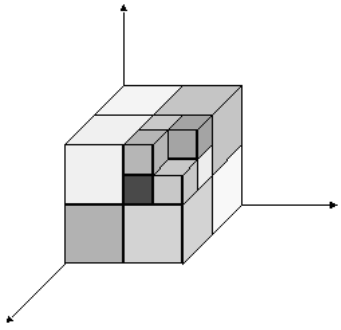


Figure 1.6 Représentation volumique à l'aide d'un octree d'après [Ramos, 2003].

L'avantage de cette technique réside dans la possibilité de distinguer assez facilement l'intérieur et l'extérieur d'un solide et de calculer le volume du solide par énumération de ces volumes unitaires. En contrepartie, la représentation est approximative, et pour des objets dont la représentation est bien détaillée, le nombre de voxels est très important.

Une dernière technique présentée ici est la construction de la géométrie d'une donnée 3D par le processus d'*instanciation de primitives*. Cette technique s'appuie sur un ensemble prédéfini de formes géométriques. L'intérêt de l'instanciation de primitives est qu'elle introduit un niveau d'abstraction entre l'usage fait par le concepteur et les éléments géométriques proprement dits qui la représentent. L'arsenal de primitives mises à la disposition des concepteurs peut couvrir plusieurs techniques de description sans que le concepteur en soit perturbé lors de la modélisation de nouvelles scènes.

Les divers formats de représentation 3D s'appuient sur une ou plusieurs de ces techniques de représentation de la géométrie. Le choix de supporter plusieurs techniques de descriptions est judicieux car aucune technique ne peut être considérée comme étant la meilleure pour tous les domaines d'application.

Dans la section suivante nous nous intéressons aux moyens d'ajouter une apparence au squelette de la géométrie 3D construite.

1.2.3. Apparence

L'apparence d'une scène 3D est donnée par les éléments graphiques qui sont attachés aux éléments géométriques qui décrivent la scène. L'apparence d'une scène peut être porteuse d'information toute aussi importante que la géométrie et la forme des objets qu'elle décore. Dans le domaine de la visualisation de données à l'aide de scènes 3D les couleurs jouent un rôle très important tel qu'illustré dans [Robertson *et al.*, 1991][Sebrechts *et al.*, 1999][Wiza *et al.*, 2005]. L'apparence d'une scène 3D est introduite à travers le type de matériel et les textures qui sont associés aux éléments géométriques.

Le type de matériel décide de comment l'élément géométrique interagit avec la lumière lors du processus de visualisation. Suivant le type de matériel, plusieurs comportements peuvent être imposés aux supports géométriques : refléter, absorber ou filtrer la lumière, etc. Le comportement des objets géométriques vis-à-vis de la lumière doit être pris en compte au moment du calcul de visibilité en vue d'un rendu efficace.

Si le matériel régit les propriétés intrinsèques des objets géométriques, leur aspect extérieur est sous l'emprise des textures. Une texture peut être définie à l'aide d'une ou plusieurs couleurs, d'une image ou d'un objet vidéo. Pour chaque objet géométrique on peut associer une ou plusieurs textures. Lorsque deux ou plusieurs objets graphiques sont utilisés pour décorer un objet géométrique, on parle de multi-texture. Dans le cadre d'une multi-texture, soit on emploie plusieurs textures sur la même région spatiale, soit on divise l'objet géométrique en sous-régions et on associe à chacune sa propre texture. Dans le cas où plusieurs textures sont associées à la même région spatiale, il est possible d'associer des niveaux de transparence et des opérateurs qui décident de la manière dont les couches sont exploitées afin de déterminer la couleur résultante dans chaque point de la texture.

1.2.4. Autres dimensions

D'autres dimensions caractérisant les espaces 3D peuvent être inclues dans les modèles 3D. Par exemple, afin d'accroître le réalisme de la scène, des lumières peuvent être disposées dans la scène. La présence de divers sons peut être *apposée* dans certaines parties de la scène.

Pour donner vie aux modèles 3D, des animations peuvent être associées aux objets géométriques. Ainsi, on peut retrouver : des transformations géométriques introduites par le biais d'interpolations, des changements de l'apparence des objets, des variations dans l'intensité de la lumière, etc. De la même façon que l'apparence, le temps peut être introduit en associant des intervalles de vie directement aux objets géométriques [Duecker *et al.*, 1997][Hetherington *et al.*, 2004].

Toutes ces dimensions peuvent être introduites directement dans le document 3D en s'appuyant sur la flexibilité offerte par le graphe de scène. Par exemple, dans [Hetherington *et al.*, 2004] les auteurs introduisent la notion de temps au sein de modèles 3D au moyen d'intervalles de temps associés aux descriptions géométriques.

Après cet aperçu des principales caractéristiques de l'information 3D, dans la suite de ce chapitre nous nous intéressons à une famille d'outils, de standards et de langages de descriptions de documents 3D.

1.3. Outils de modélisation de scènes 3D

Dans cette section nous présentons les différents outils, langages et solutions logicielles permettant de générer, décrire et créer les différentes dimensions d'un document 3D, présentées dans la section précédente.

Les premiers outils de conception 3D sont issus du domaine de la Conception Assistée par Ordinateur (CAO). Initialement, l'accent était mis sur la géométrie et l'apparence des différents ensembles modélisés. La description des interactions avec les objets et la notion de scène n'étaient pas intégralement supportées. Maintenant, de plus en plus d'applications s'axent sur une description complète de la scène 3D : la géométrie des objets, leur apparence, leur comportement (à travers des animations temporelles et la réactivité des objets 3D) et l'environnement.

Avec l'avènement de grandes quantités d'informations géographiques sur le Web, de nouveaux outils, plus ouverts au grand public, ont vu le jour. Par exemple, le logiciel *SketchUp*[6] permet de créer des scènes 3D basiques avec très peu de moyen et en très peu de temps.

Lorsque les besoins en termes d'interaction deviennent trop complexes et difficilement exprimables à l'aide d'outils de conception, une solution alternative est constituée par des bibliothèques logicielles ou des API (*Application Programming Interface*) orientées 3D. Communément, ces bibliothèques sont utilisées afin d'accroître les possibilités des concepteurs de mettre en œuvre des animations et des processus d'interactions complexes au sein des scènes 3D. Elles offrent principalement une interface qui permet aux concepteurs d'interagir et de s'abstraire de la réalisation géométrique des objets. Les bibliothèques peuvent être cataloguées en deux grandes familles. Les bibliothèques de bas niveau (OpenGL[7], DirectX D3D[8]) s'intéressent plus particulièrement à la construction et au rendu géométrique. Les bibliothèques de haut niveau (Java3D[9], Open Inventor[10], …) visent à exploiter les objets 3D au sein d'une scène 3D définie comme un espace fortement interactif. Pour le rendu géométrique, elles s'appuient sur les bibliothèques de bas niveau.

[6] http://www.sketchup.com
[7] http://www.sgi.com/products/software/opengl/
[8] http://www.microsoft.com/windows/directx/
[9] http://java.sun.com/products/java-media/3D/
[10] http://oss.sgi.com/projects/inventor/

Les bibliothèques de bas niveau n'offrent pas toutes les possibilités de constructions géométriques présentées dans la section 1.2.2. La plupart d'entre elles n'utilisent que les polygones et plus précisément des triangles pour assurer le rendu. Ce choix, dicté par le besoin d'accroître les performances en terme de rendu, n'est pas réducteur. Tout objet peut être remplacé de manière approximative par une ou plusieurs surfaces polygonales. Le nombre de polygones utilisés dans le processus croît dans le même sens que le degré de fidélité souhaité.

Dans la famille des bibliothèques de haut niveau nous retenons Java3D et la famille de produits *Sillicon Graphics*[11] (SGI), avec *Open Inventor*, *OpenGL Performer* et *OpenGL Optimiser*. Dans ce genre de bibliothèque, on retrouve des classes et des fonctions qui assurent la gestion complète d'une scène 3D. La notion de graphe de scène se retrouve au premier rang dans toutes ces approches. Le graphe de scène assure l'accès direct à toutes les dimensions d'une scène 3D.

Cependant, ces solutions, qui se montrent très efficaces d'un point de vue de la visualisation, car conçues sur mesure, restent uniquement à la portée des initiés. Afin de créer des scènes, les concepteurs doivent maîtriser la complexité des langages de programmation (tels que C ou Java) et des plates-formes associées. De plus, ces solutions logicielles n'offrent que des possibilités limitées de transfert ou d'échange de scènes ou de parties de scènes entre plusieurs concepteurs.

Les langages de description constituent une ouverture du monde 3D aux utilisateurs novices. La caractéristique principale de ces langages est la description explicite de toutes les dimensions d'un document 3D : géométrie, apparence, animations, etc. Si l'encodage choisi est de type texte [Web3D, 1997] ou XML [Web3D, 2004], le concepteur peut décrire textuellement la scène 3D en se concentrant sur sa géométrie et son apparence et non pas sur le processus de visualisation. Cependant lorsque la taille et le nombre de détails présents dans la scène deviennent conséquents, le concepteur fait souvent appel à l'utilisation d'outils de conceptions capables de générer des fichiers échangeables entre plusieurs acteurs. Les langages de description facilitent ainsi le partage et la réutilisation des informations 3D. Des langages propriétaires, tels que 3DS et DXF développés par Autodesk [Autodesk, 2007], LWO proposé par [Hasting *et al.*, 1994] ou COB proposé par [Caligari, 2002], ont sensibilisé la communauté aux avantages offerts par l'utilisation d'un format d'échange de données 3D. L'inconvénient majeur de ces langages propriétaires réside dans le fait qu'ils ne peuvent être lus que par les applications pour lesquelles ils ont été développés. De plus, leur succès et pérennité dépendent directement de la compagnie logicielle qui les supporte.

Dans ce qui suit, nous focalisons notre étude sur le standard de description X3D [Web3D, 2004] qui jouit du statut de standard public ISO et qui bénéficie d'un encodage de type XML ce qui lui confère une pérennité beaucoup plus importante car ce standard est supporté par toute une communauté.

1.4. Le standard X3D

La communauté 3D bénéficie d'un important support offert par le Web3D Consortium[12]. Ce dernier implique activement de grands acteurs industriels (NASA, HP, nVIDIA, Sun Microsystem) et de nombreux laboratoires de recherche (*Communications Research Center of Canada*, *GIS Research Center at Feng Chia University*, ...). Les travaux de recherche du

[11] http://www.sgi.com
[12] http://www.web3d.org

consortium sont dirigés vers le développement d'un standard universellement reconnu visant à uniformiser la diffusion de l'information 3D sur le Web.

Le standard *eXtensible 3D* (X3D) [Web3D, 2004] a été initialement proposé en 2002. X3D a été présenté comme une version améliorée de *Virtual Reality Modelling Language* (VRML97) [Web3D, 1997]. En juillet 2002, le Web3D Consortium a rendu publique la version finale qui a été soumise pour standardisation à l'*International Standardization Organisation* (ISO). X3D a été accepté au mois d'août 2004 en tant que standard ISO/IEC 19775 subvenant aux besoins de communication pour les scènes 3D en temps réel. Les spécifications finales ont été rendues publiques en octobre 2004. Etant dès le départ conçu comme un standard fortement extensible, X3D a subi des évolutions au fil du temps afin de combler les manques relevés par la communauté 3D. Ainsi, en avril 2006 un amendement [Web3D, 2006] a été apporté à la spécification initiale afin de prendre en compte les nombreuses suggestions apportées par la communauté 3D impliquée dans le consortium Web3D.

Ce standard définit un environnement d'exécution et un mécanisme de déploiement pour le contenu 3D et les applications 3D qui fonctionnent au-dessus d'un réseau. Il combine la description de la géométrie et de l'apparence, la description du comportement et des dispositifs de contrôle offrant des possibilités d'interaction lors de la visualisation de la scène. Le standard propose différents types de sérialisation [Web3D, 2005], y compris un codage en XML [Yergeau *et al.*, 2004]. Nous utilisons ce type d'encodage afin de faciliter la compréhension des exemples présentés.

X3D est extensible et est organisé en profils. Chaque profil contient un ensemble bien défini de composants. Chaque composant introduit une collection spécifique de nœuds. Les futures extensions du standard impliquent la spécification de nouveaux composants et le regroupement de ceux-ci en nouveaux profils. À ce jour, il existe six profils. Le profil *Core* regroupe uniquement les éléments nécessaires à la construction d'une scène simple. Cela vise à favoriser l'implémentation de plusieurs navigateurs qui supportent une description X3D minimale, en éliminant la complexité d'une implémentation X3D complète. Le profil *Interchange* ajoute la possibilité d'échanger de la géométrie et des animations entre différents auteurs. Le profil *Interactive* contient les éléments décrivant le comportement d'objets 3D et des moyens basiques d'interactions. Le profil *MPEG-4 interactive* fournit les éléments pour l'interopérabilité avec le standard MPEG-4 [Koenen, 1999]. Dans les scènes décrites conformément au profil *Immersive*, l'utilisateur peut être complètement plongé au sein de la scène. Il est muni d'outil de contrôle de la navigation et peut interagir librement avec l'environnement. Le profil *Full* couvre toutes les fonctionnalités décrites par le standard X3D.

Un document X3D représente une scène 3D comme un graphe acyclique direct. Le graphe contient les primitives géométriques (`Cube`, `Box`, `IndexedLineSet`...) qui composent la scène, ainsi que les transformations géométriques qui imposent des translations, des mises à l'échelle ou des rotations aux primitives (`Transform`). La description de l'apparence est incluse dans la définition de chaque primitive géométrique. Quelques éléments non géométriques sont inclus : objets composites (`Group`), éléments de l'environnement (`Light`, `Viewpoint`...), mécanismes de réutilisation (`DEF/USE`, `Prototype`), métadonnées (`WorldInfo`, `MetadataSet`, `MetadataString`, …). Mises à part la géométrie, l'apparence et la description de l'environnement qui permettent de créer des scènes 3D statiques d'une grande fidélité, le standard X3D permet aussi d'animer les scènes en introduisant de l'interactivité par le biais de mécanismes de contrôle et de réactivité (`Sensors`, `Routes`, `Scripts`). L'utilisateur peut également se trouver plongé au cœur de l'univers 3D à travers l'utilisation d'avatars [Bogdanovych *et al.*, 2005][Celentano *et al.*, 2004][Chittaro *et al.*, 2003]. Notons que ces aspects dynamiques de scènes 3D ne font pas l'objet de cette thèse.

En termes de modélisation géométrique, X3D propose plusieurs possibilités de représentations. La modélisation à base de primitives (`Box`, `Cone`, `Cylinder`, `Sphere`, …) est complétée par la modélisation à base de frontières polygonales (`IndexedFaceSet`, `[Indexed]TriangleSet`, …) et la technique du fil de fer (`[Indexed]LineSet`, `PointSet`, …). Des similitudes avec la technique de modélisation *Constructive Solide Geometry* peuvent être aussi observées. La construction d'objets complexes suit les mêmes règles de construction qu'un arbre dont les feuilles sont les primitives ou objets géométriques et les nœuds sont des transformations géométriques solides (`Transform`) appliquées aux nœuds fils.

L'apparence des objets géométriques est régie par la nature du matériel (`Material`) et l'association d'une ou plusieurs textures (`PixelTexture`, `ImageTexture`, `MovieTexture`, respectivement `Multi-texture`). Le matériel caractérise le comportement de l'objet géométrique associé vis-à-vis de la lumière (la lumière émise, la lumière reflétée, la transparence, la brillance, etc.). La texture décide de l'aspect visuel de l'objet géométrique. Lorsque plusieurs textures sont associées au même objet, des opérateurs prédéfinis peuvent être employés afin de décider du résultat du mélange de textures.

Afin d'illustrer l'utilisation de X3D, nous présentons dans la Figure 1.7 une scène représentant deux bâtiments à plusieurs étages situés des deux côtés d'une route. Un bâtiment est représenté comme un groupe de boîtes (lignes 4-17 pour le bâtiment gris et lignes 18-20 pour le bâtiment noir). La position et la taille des boîtes sont imposées par les éléments `Transform` (ligne 5 dans la Figure 1.7) qui contiennent la définition de chacune des boîtes. Nous utilisons quelques variations de la couleur principale pour chaque bâtiment afin de mieux distinguer les planchers. La couleur de chaque boîte est définie par la valeur de l'élément `Material` (ligne 9 dans la Figure 1.7). La scène contient également deux rues modélisées par deux boîtes aplaties (voir la Figure 1.7).

batiments.x3d

```
 1:<X3D profile="Core" ...>
 2: <Scene>
 3:   <Group id="GRIS">
 4:     <Transform id="GRISrdc"
 5:         translation="-23 1 -12" scale="2 1 12">
 6:      <Shape>
 7:       <Box/>
 8:       <Appearance>
 9:        <Material diffuseColor="0.9 0.9 0.9"/>
10:       </Appearance>
11:      </Shape>
12:     </Transform>
13:     ...
14:     <Transform id="GRIS3eme"
15:         translation="-23 7 -12" scale="2 1 12">
16:      ...
17:     </Transform>
18:   </Group>
19:   <Group id="NOIR">
20:      ...
21:   </Group>
22:   <Group id="RUES">
23:      ...
24:   </Group>
25: </Scene>
26:</X3D>
```

graphe de scène partiel

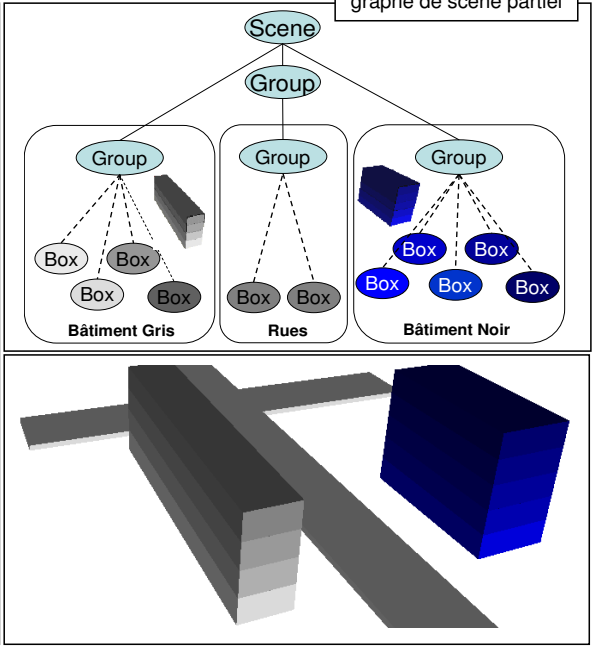


Figure 1.7 Extrait du fichier X3D modélisant une scène urbaine.

1.5. Autres langages de description de scènes 3D

D'autres standards qui mettent en avant des techniques de compression et d'encodage efficaces sont également disponibles et nous en présentons brièvement trois d'entre eux (U3D [3DIF, 2006], XAML[MacVittie, 2006], 3DXML[Versprille, 2005]).

U3D [3DIF, 2006]

Issu d'un groupe de travail du Web3D Consortium, le *3D Industry Forum*[13] a proposé fin 2004, en collaboration avec l'organisation *European Computer Manufacturers Association* [14](ECMA), un nouveau standard 3D : *Universal 3D* [3DIF, 2006]. U3D essaie de répondre d'une manière pertinente aux besoins spécifiques de l'industrie en s'intéressant notamment aux données de type CAO 3D. La motivation principale de ce travail a été de répondre aux besoins accrus de réutilisation de données 3D. Les entreprises souhaitent disposer d'outils leur permettant de réutiliser les ressources 3D existantes.

Parmi les spécificités de U3D nous citons : un encodage binaire des fichiers, des techniques de compression spécifiques au domaine, la prise en compte de niveaux de détail continus, l'accès progressif à l'information, le support pour l'animation. L'encodage binaire et les techniques de compression spécifiques au domaine permettent d'obtenir des fichiers de petites tailles. L'existence de niveaux de détail continus et l'accès progressif à l'information offrent la possibilité d'adapter directement, au moment de la diffusion, la quantité d'information visualisée suivant le contexte de diffusion.

XAML [MacVittie, 2006]

eXtensible Application Markup Language (XAML) [MacVittie, 2006] est un langage déclaratif développé pour les besoins du nouveau système d'exploitation Windows Vista de Microsoft. Les interfaces utilisateur des futures applications développées pour Windows Vista sont créées avec XAML. Basé sur le langage XML [Yergeau *et al.*, 2004], il facilite le travail du développeur et permet de décrire aisément l'interface d'une application Web. L'idée est en effet de séparer la construction de l'interface utilisateur du code sous-jacent. XAML inclut également des fonctionnalités pour manipuler des objets en trois dimensions, de manière analogue à X3D. De ce fait, il est potentiellement possible d'écrire et modifier les applications XAML à l'aide seulement d'un éditeur de texte classique. Cependant, comme il est compliqué d'écrire du X3D à la main, des outils tels que *Microsoft Visual Studio*, prennent en charge la production de code XAML. Par rapport à X3D, l'intérêt de ce langage est qu'il fonctionne directement dans le navigateur Internet Explorer sur les plates-formes Windows sans installer de *plugins* spécifiques. Cependant, la portabilité des scènes XAML vers d'autres systèmes est impossible, car chaque primitive utilisée dans XAML trouve son implémentation dans les classes du système Windows.

3DXML [Versprille, 2005]

3DXML [Versprille, 2005] proposé par *Dassault Systems* est également un format qui s'appuie sur le langage XML pour modéliser et partager des données 3D. S'appuyant sur le standard XML [Yergeau *et al.*, 2004], ce langage autorise tout programme logiciel à lire, écrire et enrichir les contenus 3DXML à l'aide d'outils standard. Ce choix de conception vise à encourager une large adoption de 3DXML et une réduction des coûts de conversion de fichiers depuis des formats 3D existants. Ce langage propose des fonctionnalités telles que la multi-représentation de structures 3D et une compression d'éléments géométriques complexes,

[13] http://www.3dif.org
[14] http://www.ecma-international.org/

accélérant ainsi la transmission de fichiers et leur chargement. Cette efficacité d'encodage et de transmission est acquise grâce à l'encodage de la géométrie au moyen de surfaces de type NURBS [Piegl, 1991] au lieu de polygones.

Fondamentalement, il n'y a pas une grande différence au niveau conceptuel entre le langage 3DXML et X3D. La différence est relative aux futures évolutions et au support technologique accordé aux deux langages. L'évolution et l'adoption du langage 3DXML reposent entièrement sur le succès de l'entreprise *Dassault Systems*.

1.6. Synthèse

Dans ce chapitre nous avons décrit les principales caractéristiques d'une scène, ainsi que les moyens de constructions de scènes 3D. Nous pouvons remarquer qu'à ce jour les concepteurs disposent d'une large palette d'outils et formalismes leur permettant de créer les scènes 3D.

Par la suite nous considérons pour notre étude uniquement les solutions définissant les scènes au moyen de documents structurés. L'ensemble des standards de représentation proposent une description autour de la notion de graphe de scène, ce qui permet d'exploiter cette structuration afin de mettre facilement en œuvre des opérations concernant la gestion de documents 3D proches de celle que l'on rencontre dans le cadre des documents structurés.

Nous voulons être en mesure d'analyser, d'extraire ou bien d'appliquer des transformations aux éléments d'une scène afin d'assurer la maintenance de celle-ci pendant son cycle de vie : création, déploiement, réutilisation, adaptation. **Pour faciliter l'illustration des divers problèmes et solutions concernant la gestion des données 3D, nous privilégions l'étude de solutions proposées autour des langages de description XML, et plus particulièrement sur X3D.**

Dans cette section nous avons porté notre attention sur la définition des caractéristiques de base d'une scène 3D : sa géométrie, son apparence… À ce niveau, **les scènes sont dépourvues de toute information concernant la nature ou la sémantique des objets qui y sont inclus**. Ceci n'est pas gênant dans la mesure où les scènes ont été conçues pour être visualisée par des humains. Cependant, si l'on veut mettre en place des traitements automatiques effectués par une entité logicielle, **il est nécessaire de rajouter une couche sémantique permettant de bien isoler et caractériser les objets contenus**.

La section suivante est dédiée à l'analyse de moyens qui permettent d'enrichir la description brute d'une scène avec des informations sémantiques.

2. La sémantique dans les mondes 3D

L'expérience 3D peut être rendue encore plus intéressante si l'apport informationnel fourni par la scène s'enrichit avec des données sémantiques relatives aux objets présentés dans la scène. Au même titre que la description de la géométrie et de l'apparence, la sémantique peut faire partie intégrante des documents modélisant les scènes 3D comme illustré dans [Polys *et al.*, 2004][Hetherington *et al.*, 2004][Cruz *et al.*, 2005][Pittarello *et al.*, 2005], etc. Dans [Polys *et al.*, 2004], les auteurs s'intéressent à la visualisation des parties du squelette humain en incluant des données médicales en tant que métadonnées dans les documents X3D. Dans [Hetherington *et al.*, 2004] les auteurs incluent la dimension temporelle dans les scènes 3D en ajoutant des métadonnées indiquant la durée de vie de chaque objet de la scène. Des applications dans le domaine de l'architecture s'intéressent également à la sémantique des composants géométriques. L'objectif est de mieux assurer la cohérence du processus de collaboration entre les acteurs impliqués dans le design 3D et la construction effective des bâtiments [Cruz *et al.*, 2005]. Il peut également s'agir d'enrichir l'expérience de navigation (en proposant le type de navigation recommandée - marcher ou voler - pour chaque région de l'espace 3D) et les modes d'interaction (en informant l'utilisateur sur la région où il se trouve ou en mettant en œuvre un accès conditionnel) au sein des modèles déjà construits [Pittarello *et al.*, 2005].

Ces observations nous confortent dans notre choix de traiter la sémantique comme une dimension à part du contenu 3D. Dès que les concepteurs de scènes 3D veulent mettre en place des actions plus complexes au sein d'une scène 3D (transformation, interaction, dynamisme, intégration), autres que la simple visualisation des objets, ils sont obligés de stocker et de prendre en compte la sémantique (nature, propriétés, relations) des objets 3D.

Dans ce chapitre nous étudions la manière dont cette préoccupation de prise en compte et d'exploitation d'information sémantique se reflète dans la communauté 3D.

Tout d'abord, nous nous intéressons aux modalités de représentation de la sémantique dans les applications 3D. Par rapport à l'endroit où se trouve matérialisée la sémantique relative aux objets 3D dans une application 3D, nous identifions deux approches de stockage de la sémantique :

- **sémantique interne** – stockée à l'intérieur du document 3D en utilisant les moyens propres aux standards de modélisation tels que X3D ;
- **sémantique externe** – stockée dans un document autre que la scène elle-même en utilisant des standards de représentation de la sémantique tels que RDF, OWL ou MPEG-7.

Ensuite pour chaque proposition nous mettons en exergue le formalisme de représentation choisi et les dimensions du document 3D concernées par la sémantique :

- **la description de haut niveau de la géométrie**,
- **la description de haut niveau de l'apparence**,
- **la description de haut niveau de l'organisation spatiale**,
- **la description des propriétés des objets réels modélisés par objets de la scène** – *sémantique générale,*
- **la description de relations sémantiques entre les objets de la scène** – *sémantique locale.*

Pour les propositions qui adoptent une approche basée sur la sémantique externe nous nous intéressons également aux **moyens utilisés pour localiser un objet 3D au sein d'une scène**. En effet, lorsque les informations sémantiques sont stockées à l'extérieur du document il est nécessaire de disposer d'un moyen d'indiquer la partie du document 3D concernée par la sémantique.

Ainsi, dans un premier temps nous présentons dans ce chapitre quelques considérations, standards et langages s'intéressant à la sémantique dans un cadre général indépendant des spécificités des documents 3D. Dans un second temps, nous explorons les moyens d'ajouter de la sémantique dans le cadre des langages et standards de modélisation de contenus 3D. Enfin, nous réalisons un survol des applications et travaux de recherche qui s'intéressent plus particulièrement à l'exploitation de la sémantique des contenus 3D en considérant plusieurs domaines d'application : la gestion de l'espace, la modélisation de formes 3D, les données médicales, les données temporelles, etc.

2.1. Standards pour la description sémantique de documents

Dans cette section, nous étudions trois standards qui œuvrent pour la formalisation des moyens de caractérisation sémantique des ressources. Les deux premiers, RDF [Manola *et al.*, 2004] et OWL [McGuinness *et al.*, 2004] offrent des solutions génériques pour la description d'une ressource quelconque. Le troisième, MPEG-7 [Martinez *et al.*, 2002] offre un cadre formel dédié au cas particulier du rattachement des informations sémantiques aux ressources multimédia. Pour chacun de ces standards nous nous intéressons aux moyens d'expression de la sémantique et de sa mise en relation avec les fragments d'un document en général.

2.1.1. Resource Description Framework

Le *Resource Description Framework* (RDF) [Manola *et al.*, 2004] est un langage pour la représentation de l'information concernant les ressources disponibles sur le Web. Les informations associées à une ressource couvre une large palette d'attributs simples tels que le titre, l'auteur, la date de modification, jusqu'aux données plus complexes telles que le contenu ou bien les droits d'auteurs. Le langage RDF a été développé par le consortium W3C. Depuis

2004, ce langage est considéré comme un standard par ce même consortium dans le cadre de ses activités organisées autour du Web sémantique [Berners-Lee *et al.*, 2001]. RDF rend possible l'interopérabilité sémantique entre applications qui échangent des données dans un langage machine. En effet, il permet de donner du sens aux ressources et a été conçu afin d'être facilement lu et compris par les machines.

Présentation générale

RDF repose sur un modèle logique qui permet d'énoncer des propositions qui décrivent des ressources (*i.e.* des composants documentaires) en utilisant des triplets {*ressource*, *propriété*, *valeur*}. Les triplets peuvent être également représentés par un graphe étiqueté, orienté, aux propriétés mathématiques bien définies. Ces triplets constituent la construction de base en RDF :

- la *ressource* est toute entité d'information pouvant être référencée en un bloc, par un nom symbolique (littéral) ou un identificateur (URI[15]). Les URI utilisés ne pointent pas forcément vers des documents consultables sur le Web et peuvent référencer n'importe quel objet identifiable (tel qu'une page Web, un livre, un morceau de musique). En plus, les propriétés RDF elles-mêmes peuvent être associées à des URI afin de pouvoir définir de manière précise les relations entre les items liés par la propriété en question ;
- la *propriété* est un aspect spécifique, une caractéristique, un attribut ou une relation utilisée pour décrire une ressource. Chaque propriété a une signification spécifique, des valeurs autorisées, des types de ressources qu'elle peut décrire et des relations avec d'autres propriétés ;
- la *valeur* de la propriété.

Une ressource spécifique associée à une propriété et une valeur constituent ensemble une expression ou proposition RDF. Ces trois parties (ressource, propriété, valeur) sont appelées respectivement sujet, prédicat et objet.

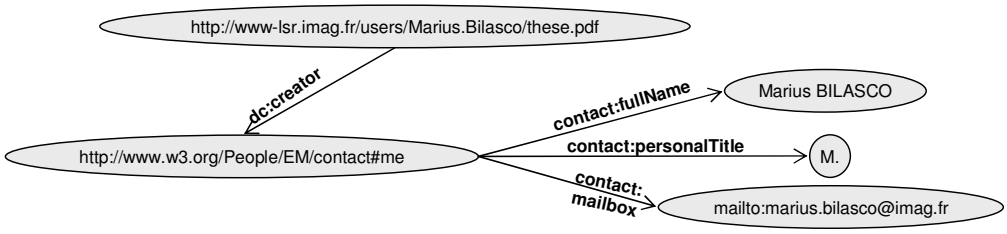


Figure 2.1 Graphe RDF pour la caractérisation d'une ressource Web.

Dans la Figure 2.1, nous présentons un graphe RDF qui exprime les propositions suivantes : je suis le créateur du document identifié par l'URI `http://www-lsr.imag.fr/users/Marius.Bilasco/these.pdf`; mon nom est `Marius Bilasco` et mon adresse email est `marius.bilasco@imag.fr`. Afin de simplifier l'illustration, deux espaces de noms `dc` (`http://purl.org/dc/elements/1.1/`) et `contact` (`http://www.w3.org/2000/10/swap/pim/contact#`) sont utilisés.

RDF offre également une syntaxe de type XML (nommée *RDF/XML*) pour l'enregistrement de ces graphes. Tout comme HTML, RDF/XML est facilement lu et interprété par un ordinateur. L'exemple de la Figure 2.2 est un extrait de la description en RDF/XML

[15] Universal Resource Identifier

correspondant au graphe de la Figure 2.1. L'élément `dc:creator` est défini par le consortium *Dublin Core* [Weibel *et al.*, 1998].

```
<?xml version="1.0"?>
<rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <rdf:Description
         rdf:about="http://www-lsr.imag.fr/users/ Marius.Bilasco/these.pdf">
    <dc:Creator>
         <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
            <contact:fullName>Marius Bilasco</contact:fullName>
            <contact:mailbox rdf:resource="mailto:marius.bilasco@imag.fr"/>
            <contact:personalTitle>M.</contact:personalTitle>
         </contact:Person>
     </dc:Creator>
   </rdf:Description>
</rdf:RDF>
```

Figure 2.2 Exemple de description RDF.

Un des grands avantages de RDF est son extensibilité, à travers l'utilisation de schémas RDF (*RDF Schema*) qui peuvent s'intégrer et ne s'excluent pas mutuellement grâce à l'utilisation du concept d'espace de noms. Les déclarations RDF définissent des relations entre des objets (nœud d'un graphe) qui appartiennent à un univers sémantique. À chacun de ces univers sémantiques correspond un domaine nominal, identifié par un préfixe particulier. Un tel domaine, pour lequel sont définies des propriétés spécifiques et des catégories conceptuelles, est appelé un schéma. La notion d'espace de noms est une notion fondamentale puisque c'est grâce à elle que l'on peut étendre les schémas de descriptions existants.

Extensibilité du langage RDF

Tout utilisateur a la possibilité de créer un nouveau schéma, de modifier, et notamment d'enrichir et d'affiner un schéma existant. Ceci permet de définir un nouveau schéma personnalisé qui répond mieux aux besoins de caractérisation des propriétés d'objets existant dans un nouveau contexte.

RDFS (pour RDF Schema) offre les moyens de définir un schéma de métadonnées qui permet de :

- donner du sens aux propriétés associées à une ressource en utilisant la construction `rdfs:class` qui permet de déclarer une nouvelle classe de ressources ;
- formuler des contraintes sur les valeurs associées à une propriété :
  - en utilisant la construction `rdfs:range` qui permet d'introduire une restriction du domaine de valeurs d'une propriété – par exemple, pour une propriété qui caractérise l'âge d'une personne on peut imposer que les valeurs soient des entiers ;
  - ou bien en utilisant la construction `rdfs:domain` pour restreindre le domaine d'application de propriétés – par exemple, pour une propriété qui caractérise un auteur par sa date de naissance, on peut exiger que les valeurs de cette propriété soient une référence à une personne.

Une propriété spécifiée dans un schéma RDFS est vue comme une ressource dont le type `rdf:type` est `rdf:property`. Dans la Figure 2.3, nous définissons la propriété *date_de_naissance*

comme étant associable à une `personne` et ayant une représentation de sa valeur en tant que `literal`.

```
<rdf:property
     rdf:about="http://www-lsr.imag.fr/STEAMER/rdf-schema#date_de_naissance">
  <rdfs:domain
       rdf:resource="http://www-lsr.imag.fr/STEAMER/rdf-schema#personne"/>
  <rdfs:range
       rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
</rdf:property>
```

Figure 2.3 Exemple de restriction du domaine et des valeurs d'une propriété en RDF Schema.

Ces possibilités d'extensibilité de l'ensemble des descripteurs RDF présentent un intérêt certain pour le domaine 3D, car elles facilitent la construction de descripteurs propres aux mondes 3D ou bien à des domaines spécifiques rattachés aux diverses applications de la 3D. Afin de pouvoir exploiter automatiquement les nouveaux descripteurs il est souhaitable que la signification que l'on associe soit clairement définie.

Les anciennes solutions qui s'appuient sur les DTD et les schémas XML [Fallside *et al.*, 2004] suffisent lorsque l'on s'est mis d'accord à l'avance sur la structure et la nature des données échangées. Cependant, ces solutions ne sont pas efficaces lorsque les vocabulaires ne sont pas complètement définis dès le début, s'ils évoluent, ou encore si le sens d'un terme change suivant le contexte d'utilisation. Les standards RDF et *RDF Schema* entrouvrent quelques voies vers la résolution de ce problème en permettant de caractériser des termes au moyen de descriptions sémantiques. Avec *RDF Schema*, on peut définir des classes qui ont des sous-classes, des classes mères, des sous-propriétés, des domaines et des intervalles. Cependant, ces descriptions restent relativement simples alors qu'une sémantique plus riche est nécessaire pour aboutir à l'interopérabilité entre les nombreux schémas développés de manière autonome par différentes communautés. Par exemple, *RDF Schema* ne peut pas spécifier que les classes *Personne* et *Bicyclette* sont disjointes, ou bien qu'une bicyclette a exactement deux roues. RDF propose uniquement un support minimal pour l'interopérabilité en s'appuyant sur les propriétés `rdfs:subClassOf` et `rdfs:subPropertyOf`.

Un nouveau langage doit fournir des primitives pour mettre en relation les différentes représentations, permettant ainsi, la traduction du concept en différentes ontologies. Le langage devrait être capable d'exprimer une large variété de concepts, mais en contrepartie il devrait également encourager et faciliter le développement d'outils capables de raisonner sur les concepts. Dans ce sens, nous présentons dans la section suivante le standard OWL [McGuinness *et al.*, 2004].

2.1.2. Web Ontology Language

Dans cette section, nous montrons comment les ontologies viennent compléter les standards de représentation présentés précédemment afin d'assurer des fondements solides pour le Web sémantique [Berners-Lee *et al.*, 2001]. Le Web sémantique [Berners-Lee *et al.*, 2001] repose sur la représentation de la sémantique à l'aide d'ontologies partagées sur le Web. Le Web sémantique est une vision du Web dont toute information a un sens explicit qui permet aux machines de la traiter et de l'intégrer aux autres informations de manière automatique. Les ontologies offrent une modalité flexible de représentation qui facilite l'interopérabilité et le transfert de connaissances entre diverses communautés et domaines d'application. En effet, les ontologies sont un élément fondamental, voire critique dans toute application (entreprise) qui

concerne la recherche et l'intégration des informations produites au sein de différentes communautés [Schuster *et al.*, 2001].

Dans un premier temps, nous donnons un aperçu de la définition d'une d'ontologie, de ses caractéristiques et des bénéfices apportés par son utilisation. Dans un deuxième temps, nous présentons brièvement OWL [McGuinness *et al.*, 2004] le standard de représentation d'ontologies sur le Web proposé par le W3C.

Définition d'une ontologie

Une ontologie [Gruber, 1995] définit les termes utilisés pour décrire et représenter une partie de la connaissance. Les ontologies sont utilisées par les humains ou bien les applications qui ont besoin de partager de l'information dans un domaine spécifique (tel que la médecine, l'architecture, le management financier, la géographie, etc.). Les ontologies comprennent des définitions de concepts de base ainsi que des relations entre eux dans une forme compréhensible par une machine. Les ontologies doivent spécifier les concepts suivants : les classes, les relations entre ces classes, les propriétés des classes et leurs relations.

Les ontologies peuvent intervenir dans l'amélioration de la recherche de contenus multimédia sur le Web. Des annotations sémantiques [Chrisment *et al.*, 2002] (souvent représentées comme libellés ou nœuds *meta*) peuvent être associées aux collections d'images, de sons, de vidéo, ou bien de données 3D. La plupart du temps, ces annotations peuvent être réalisées par des personnes ou logiciels différents dans des contextes différents. Ainsi, il est important que le processus d'indexation et de recherche puisse proposer des solutions plus pertinentes qu'une recherche standard à base de mots-clés.

Deux principaux types d'ontologies multimédia peuvent être considérés : les ontologies liées au médium et les ontologies liées au contenu. Les ontologies décrivant le médium peuvent contenir des taxonomies de différents média et de leurs propriétés (les régions d'une scène 3D [Pittarello *et al.*, 2006]). La forme la plus basique d'une telle ontologie est constituée par les définitions de types MIME. Les ontologies décrivant le contenu s'intéressent au sujet d'une ressource (par exemple, la ville couverte par une représentation 3D spécifique). Compte tenu du fait que ces ontologies décrivant le contenu ne sont pas spécifiques à un type spécial de médium, elles peuvent être réutilisées pour d'autres types de documents (images, vidéo) couvrant le même contenu.

Présentation générale du standard OWL

Tout le monde peut annoter les contenus éparpillés sur le Web en utilisant RDF ou d'autres langages sémantiques, ce qui aboutit à une grande hétérogénéité du point de vue des classes, propriétés et instances RDF utilisées pour décrire une même ressource.

OWL a été conçu afin de maîtriser l'hétérogénéité inhérente au processus de descriptions. Il permet d'exploiter les informations contenues dans les documents directement par les applications sans intervention humaine. OWL est un vocabulaire XML basé sur RDF. Ce langage est aussi un moyen standard de décrire les relations entre informations ce qui peut amener par la suite à la construction d'un réseau de confiance à propos de ces informations.

OWL représente de manière explicite le sens des termes et les relations entre eux, aux moyens de vocabulaires interconnectés. OWL est beaucoup plus expressif en termes de sémantique que les normes XML, *XML Schema* [Fallside *et al.*, 2004], RDF ou *RDF Schema*. XML offre uniquement une syntaxe pour structurer les documents. *XML Schema* ajoute des restrictions sur la structure des documents XML et permet la création de nouveaux types de données. Mais ces deux langages ne s'intéressent qu'à la syntaxe descriptive, aucun des deux

n'impose de contraintes sémantiques sur le sens des documents. RDF ajoute une sémantique simple au modèle de données permettant de décrire des ressources et les relations entre-elles. *RDF Schema* est un vocabulaire RDF qui permet de décrire et organiser les propriétés et les classes de ressources au sein d'une hiérarchie. OWL va plus loin que ces langages car il est capable de représenter de l'information interprétable par les agents logiciels sur le Web en considérant, entre autres, les relations existantes entre classes et/ou instances, leurs cardinalités, les caractéristiques des propriétés (symétrie, transitivité), etc.

OWL s'impose également en permettant de déclarer des équivalences entre classes d'information, de construire de nouvelles hiérarchies et de nouvelles classes en utilisant des opérations ensembliste telles que l'union, la disjonction, la différence, etc.

OWL est une révision du langage des ontologies Web DAML+OIL [Hendler *et al.*, 2000][Fensel *et al.*, 2001] suite aux expériences de conception et d'utilisation de ce langage dans des applications sémantiques du langage. OWL est défini en trois profils de plus en plus expressifs, chacun étant une extension du précédent : *OWL Lite*, *OWL DL*, *OWL Full*.

Le profil OWL Lite

OWL Lite est un ensemble minimal de construction OWL permettant aux utilisateurs d'organiser les informations, les propriétés et les classes de ressources dans une hiérarchie et d'ajouter des contraintes simples sur la cardinalité des rôles (un ensemble est limité à 0 ou 1 élément) dans une classification hiérarchique. *OWL Lite* permet de réaliser des annotations sur les classes, les propriétés, les individus ainsi que les définitions des ontologies. *OWL Lite* reprend toutes les fonctionnalités de *RDF Schema* liées à la définition et la construction des hiérarchies de classes et propriétés, aux annotations et la définition de nouveaux types de données.

OWL Lite ajoute des constructions permettant de définir l'équivalence entre les extensions de classes (`owl:equivalentClass`) et de propriétés (`owl:equivalentProperty`) ou de se prononcer sur les différences ou les similitudes entre les instances (`owl:sameAs`, `owl:differentFrom`, `owl:AllDifferent`, `owl:distinctMembers`). Ces outils facilitent la mise en correspondance d'ontologies car des relations (synonymies, antinomies, etc.) peuvent être définies entre concepts issus de plusieurs ontologies.

OWL Lite fait la distinction entre les propriétés caractérisant les instances d'une classe au moyen d'autres instances (`owl:ObjectProperty`) et celles liant les instances aux données simples (`owl:DataProperty`).

OWL Lite introduit des restrictions (`owl:Restriction`, `owl:onProperty`) sur les valeurs d'une propriété. La portée de la restriction peut être restreinte à un sous-ensemble (`owl:someValuesFrom`) ou bien elle s'applique à toutes les valeurs (`owl:allValuesFrom`).

OWL Lite permet également de construire de nouvelles classes en considérant les instances communes de deux classes existantes. Ce premier niveau de langage OWL se propose comme un langage permettant une migration rapide des thésaurus et autres taxonomies vers OWL. *OWL Lite* est le point de départ pour l'introduction d'outils plus expressifs.

Le profil OWL DL

Par rapport à *OWL Lite*, de nouvelles constructions de *OWL DL* permettent de définir des classes énumérées.

Les classes sont définies par l'ensemble de leurs instances (`owl:one of` – la classe *JourDeLaSemaine* est définie par ses instances *Lundi*, *Mardi*, …, *Dimanche*). La définition de

nouvelles classes en considérant les opérations ensemblistes ne se restreint plus à l'intersection (`owl:intersectionOf`) sur les instances issues de deux classes existantes, mais s'étend également à l'union (`owl:unionOf`), au complément (`owl:complementOf`) et à la disjonction (`owl:disjointWith`).

Dans le même souci d'enrichissement de l'expressivité du langage, il est désormais possible de définir l'équivalence, ou la relation d'héritage, entre des classes anonymes construites comme indiqué précédemment.

OWL DL se base sur les Logiques de Description (*Description Logic* – DL) [Baader *et al.*, 2002] et s'adresse aux utilisateurs qui veulent un maximum d'expressivité en garantissant la complétude (toute conclusion est calculable) et la décidabilité (toute conclusion est calculée dans un temps fini) de tous les raisonnements.

OWL DL inclut toutes les constructions du langage OWL, mais leur utilisation est soumise à certaines restrictions. Par exemple, une classe ne peut pas être une instance ou une propriété d'une autre classe, une propriété doit être définie explicitement en tant que propriété soit de type individu (`owl:objectProperty`) soit de type donnée (`owl:dataProperty`).

Présentation du profil OWL Full

À la différence d'*OWL DL*, la version *Full* du langage ne garantit ni la calculabilité ni la décidabilité des conclusions. Par exemple, une classe peut être considérée simultanément comme une collection d'individus et un individu en soi. Ceci est dû au fait qu'une ressource peut être tant une classe qu'une instance d'une autre classe. Il est peut probable qu'un raisonneur puisse prendre en compte tous les aspects d'*OWL Full* à cause de la liberté d'expression que le langage offre.

La puissance d'expressions de solutions mariant OWL, RDF et *RDF Schema* sont intéressantes par le fait qu'elles peuvent couvrir l'ensemble des propriétés sémantiques que l'on peut associer à une ressource Web (par exemple, données multimédia ou données 3D). Cependant, il n'existe pas dans ces langages, de structures prédéfinies capables d'accueillir les caractéristiques des données multimédia et, en conséquence, il faut donc les définir explicitement. Dans la sous-section suivante, nous présentons MPEG-7 [Martinez *et al.*, 2002], un standard universel des description qui contient de constructions spécifiques aux données multimédia.

2.1.3. Moving Pictures Expert Group-7

L'attachement d'informations sémantiques à des objets multimédia n'est pas un sujet récent dans la communauté multimédia. Des efforts importants de recherche ont été faits pour la caractérisation (des propriétés de bas niveau, ou sémantiques dites de haut niveau) de fichiers audio, d'images ou de vidéo. Les descripteurs et les schémas de description MPEG-7 [Martinez *et al.*, 2002] sont largement acceptés en tant qu'outils standard de description de données multimédia. Sa flexibilité fait de MPEG-7 un bon candidat pour remplir la tâche qui consiste à associer des annotations sémantiques aux différents types de contenus média y compris les contenus 3D.

Nous commençons par introduire les concepts sur lesquels repose le standard MPEG-7 et nous analysons les schémas de descriptions visuelles et multimédia proposés par MPEG-7.

Présentation générale de MPEG-7

Issu des efforts de standardisation du groupe de travail *Moving Picture Experts Group* (MPEG), MPEG-7 est une norme qui vise la description sémantique des ressources média. Même si les descripteurs proposés dans MPEG-7 couvrent plus particulièrement les ressources de type audio et vidéo, MPEG-7 est extensible et peut couvrir d'autres types de média.

MPEG-7 fournit une série d'outils de description de contenus audio-visuels multimédia regroupés dans les catégories suivantes : descripteurs (D), schémas de description (DS) et un langage de définition de descriptions (DDL). Un *descripteur* est une unité d'indexation décrivant les caractéristiques primaires visuelles, audio ou sémantiques des objets. Les *schémas de description*, qui constituent des descripteurs de haut niveau, regroupent plusieurs D et tout autre DS en unités structurées et sémantiques. Le DDL définit la syntaxe pour créer de nouveaux DS. Dérivé de *XML Schema* [Fallside *et al.*, 2004], le DDL assure l'extensibilité de la norme MPEG-7.

Les DS définis dans MPEG-7 couvrent actuellement les catégories suivantes : la description visuelle (VDS), la description audio (ADS) et la description structurelle de contenu multimédia (MDS). Les VDS et les ADS décrivent les structures physiques, logiques ou sémantiques d'un document multimédia. Ces structures sont construites en utilisant les DS offerts par MDS. Nous nous intéressons plus particulièrement aux descripteurs issus de VDS et MDS. Les descripteurs VDS peuvent être utilisés afin de caractériser l'apparence des données 3D. Les descripteurs MDS servent à identifier des fragments cohérents d'un point de vue sémantique au sein d'une scène 3D.

Schémas de description visuelle

Les schémas de description visuelle décrivent les caractéristiques spécifiques au contenu visuel telles que l'image et la vidéo.

Afin de caractériser une image ou une vidéo d'une manière précise, il est possible de les décomposer en région spatio-temporelle à l'aide de trois schémas de décomposition spatiale et deux schémas de décompositions temporelles. Le *GridLayout* sectionne une image en régions rectangulaires de mêmes dimensions afin de décrire individuellement chacune d'entre elles en termes de couleur, texture, etc. Les *2D/3D Multiple Views* préconisent une structure combinant des descripteurs 2D qui représentent les caractéristiques visuelles d'un objet 3D selon différents points de vue. Le *Spatial 2D Coordinates* définit un système de coordonnées 2D dans lequel on référencie de manière absolue les limites de la région concernée. Les deux derniers éléments sont les *Time Series* et *Temporel Interpolation*. Ils se prêtent plus particulièrement aux données vidéo caractérisant les aspects temporels.

Les descripteurs visuels MPEG-7 s'intéressent à la caractérisation d'une région et couvrent les caractéristiques suivantes :

- la couleur : *Colour Space* (l'espace de couleurs), *Colour Quantisation* (le nombre de couleurs uniques), *Dominant Colour*, *Scalable Colour* (histogramme de couleur basé sur le modèle de couleurs HSV : hue (type de couleur: rouge, bleu, vert), saturation (niveau de saturation), value (l'intensité)), *Colour Layout* (la distribution spatiale des couleurs), *Colour-Structure* (similaire à *Colour Layout*, lie la structure et la couleur du contenu), *GoF/GoP Colour* (étend le *Scalable Colour* à un ensemble d'images, voire une vidéo).

- la texture : *Homogenous Texture* – description quantitative précise, *Texture Browsing* – régularité, direction, rugosité, *Edge Histogram* – répartition spatiale de cinq types d'arêtes (verticale, horizontale, diagonale à 45° et 135° et une isotopique).
- la forme : *Region Shape* – caractérise la forme de la région (une région pleine, un ensemble de régions pleines ou régions avec trous ou avec discontinuités), *Contour Shape* – utilise un histogramme de courbures ce qui assure : la généralisation de formes, la robustesse aux mouvements non rigides, la robustesse au recouvrement partiel, l'invariance en rapport avec des transformations de perspective, *Shape 3D* – décrit comme un maillages de polygones ou un codage de maillage 3D (MPEG-4).
- le mouvement : *Camera Motion* – décrit le mouvement de la caméra, *Motion Trajectory* – caractérise la trajectoire d'un mouvement, *Parametric Motion* – décrit un déplacement paramétré et *Motion Activity* – signale et analyse un déplacement.
- la localisation : *Region Locator* – décrit des régions spatio-temporelles au sein des images à l'aide de boîtes 2D (*Box*) ou de polygones 2D (*Polygon*), *Spatio Temporal Locator* – décrit des régions spatio-temporelles dans une séquence vidéo (objets en mouvement) et offre des possibilités de localisations applicables seulement aux vidéo.
- la reconnaissance de visages.

Ces descripteurs peuvent être attachés à n'importe quelle partie de contenu d'un document multimédia. Dans la sous-section suivante, nous présentons les schémas de description permettant de décrire la structure d'un document.

Schémas de description multimédia

Les MDS de MPEG-7 constituent des structures de métadonnées qui s'intéressent aux entités génériques ou multimédia. Le MDS est organisé selon les axes suivants :

- l'organisation du contenu (*Content Organisation*)

  L'axe *Content Organisation* compte les structures permettant d'organiser les collections de segments, événements et/ou objets, et de décrire les propriétés communes présentes dans le *Collection Structure DS* et dans divers *Model DS*. Diverses relations peuvent être exprimées entre les éléments d'une même collection ou bien de collections distinctes telles que : l'ordre temporel, la disposition spatiale, le niveau de similarité, etc. De plus, en utilisant différents modèles et mesures statistiques, une collection peut être caractérisée par un ensemble d'attributs communs.

- la navigation et l'accès (*Navigation and Access*)

  L'accès et la navigation (axe *Navigation and Access*) sont mis en œuvre en utilisant des *sommaires, vues, partitions* et *variantes* des contenus média. Les sommaires hiérarchiques organisent le contenu en niveau de détail successifs. Les sommaires séquentiels correspondent à une présentation de type diaporama du contenu. Les *View DS* décrivent une vue structurelle, une partition ou une décomposition de contenu suivant l'emplacement spatial, le scénario temporel ou la fréquence audio. Les *Variation DS* introduisent les variantes de contenu telles que les sommaires, les résumés, les versions compressées (ou de moindre qualité), ou les diverses modalités disponibles (audio, vidéo, image, texte, etc.).

- l'interaction avec l'utilisateur (*User Interaction*)

À l'aide de l'axe *User Interaction*, l'interaction utilisateur se manifeste à travers les préférences utilisateur et l'historique de l'utilisation relative à l'usage et aux modalités de diffusion du matériel multimédia. MPEG-7 contient des descripteurs de contenu qui peuvent être comparés aux préférences utilisateurs pour personnaliser l'accès, la présentation et la modalité de diffusion.

- les éléments de base (*Basic Elements*)

  L'axe *BasicElements* inclut les composantes et les structures nécessaires pour le développement de schémas de description plus évolués. Des éléments permettant la description temporelle et spatiale, l'annotation textuelle, la description des groupes et des individus sont également considérés par cet axe. Il est également possible de construire des schémas de classification.

- la description et la gestion de contenu (*Content Description and Management*)

  Le cycle de vie des contenus multimédia, de la création au codage (stockage, formats de fichier) jusqu'à la diffusion et l'usage, est décrit en utilisant les outils de *Content Management*. Le *Creation Information DS* est composé d'information concernant la création et la classification du contenu. Les informations relatives à la création sont le titre, les annotations textuelles, le(s) créateur(s), le lieu et la date de la réalisation. La classification se fait par rapport au genre, au sujet, au but, au langage, etc. Puisque ce type d'information est rarement inclus dans le contenu, les descripteurs doivent être saisis manuellement. Les *Media Description DS* s'intéressent au stockage des média (format, techniques de compression et du codage, etc.). Ils identifient la source principale (*Master Source*) associée au contenu à partir de laquelle des variantes (*Media Profiles*) peuvent être dérivées en utilisant différents formats et codages. Les informations relatives à l'utilisation de contenus (droits de diffusion, disponibilité, droits d'enregistrement, etc.), sont décrites avec les *Content Usage DS*.

- la description de contenu (Content *Description*)

  L'axe *Content Description* fournit des schémas de descriptions de la structure physique et logique du contenu, ainsi que de la sémantique en utilisant des concepts du monde réel. Nous détaillons ces schémas dans la suite de cette section.

Définition de segments multimédia

Les éléments structurels du contenu sont les *segments* qui correspondent à un partitionnement spatial, temporel ou spatio-temporel. Un segment est associé à une partie du contenu. Le segment peut être décomposé en sous-segments ce qui mène à définir une structure de segmentation hiérarchique suivant plusieurs niveaux de détail.

Dans la Figure 2.4, nous illustrons ce type de décomposition (partie gauche de la Figure 2.4) en considérant une image (partie droite de la Figure 2.4) issue d'une scène 3D représentant un `bureau` qui contient une `table`, une `chaise` et des `livres`. La table est décomposée à son tour en : `plan` de travail, *pied g*auche et `pied d`roit.

Le *Segment DS* constitue le type de base abstrait qui est ensuite dérivé pour obtenir des sous-classes telles que : les segments audio, les segments vidéo, les segments audio-visuel, les régions fixes/en mouvements. Chaque segment est caractérisé en utilisant les autres outils de descriptions proposés par MPEG-7 (les descripteurs visuels/audio/… et les schémas de descriptions).

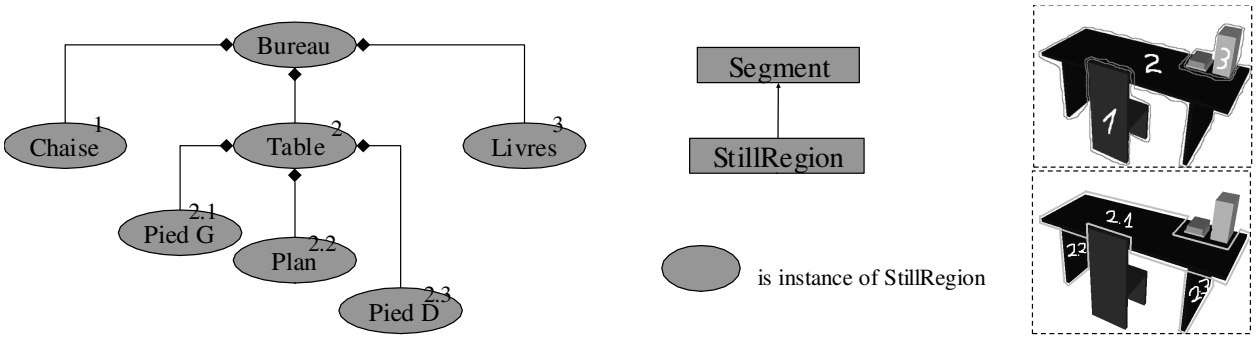


Figure 2.4 Décomposition hiérarchique des segments identifiés sur une image.

Association de la sémantique aux segments multimédia

Le MDS assure également la description sémantique. La sémantique associée à une entité est introduite par les éléments sémantiques de MPEG-7 regroupés dans le schéma de description *Semantic Base DS*. Les éléments sémantiques disponibles dans ce schéma de description sont :

- les objets (*Object DS*),
- les événements (*Event DS*),
- les lieux (*Semantic Place DS*),
- les instants temporels (*Semantic Time DS*) et
- les concepts abstraits (*Concept DS*) représentés par des libellés, des définitions, des propriétés et des relations.

Ces concepts abstraits (libellés, définitions, propriétés, relations) peuvent être instanciés en utilisant des termes faisant référence à des ressources décrites en RDF. Le terme lui-même peut être vu comme une ressource caractérisée par un ensemble de propriétés le caractérisant. Ainsi, on combine la puissance de caractérisation de contenus multimédia spécifiques à MPEG-7 et à RDF. De manière semblable à la description de la structure logique du contenu média, une nouvelle description de la décomposition du contenu à base de segments sémantiques peut être réalisée. Les nœuds de la structure ainsi obtenus représentent les concepts identifiés au sein de la scène. Les arêtes correspondent aux liens existants entre concepts.

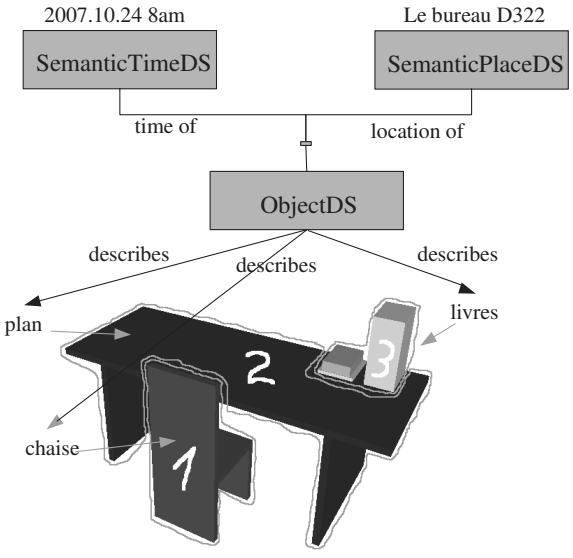


Figure 2.5 Une description sémantique MPEG-7 d'une image représentant un bureau.

Le schéma structurel (*Structure DS*) et le schéma sémantique (*SemanticBase DS*) peuvent être interconnectés. Ainsi, la description d'un contenu média peut inclure l'organisation logique d'éléments du contenu ainsi que leur sémantique.

Une description sémantique du bureau présenté dans la Figure 2.4 est illustrée dans la Figure 2.5. La description sémantique comporte la définition de trois objets sémantiques : la `chaise`, les `livres` et la `table`. Les descripteurs *spatial* et *temporel* définissent l'endroit et le moment auxquels les objets sont rattachés.

Dans la suite de ce chapitre, nous concentrons notre attention sur les spécificités des données 3D au regard de l'attachement et de l'exploitation de la sémantique.

2.2. Support pour la sémantique dans les documents X3D

Le but principal d'un modèle de scène X3D est de donner une représentation précise et riche en termes de primitives géométriques et environnementales (lumières, bruits ambiants, etc.). Malgré la précision et la richesse qu'ils peuvent apporter, les aspects sémantiques sont toutefois à peine abordés. Ce choix de modélisation peut être justifié par le fait que le type d'information sémantique associé aux objets 3D contenus dans une scène change d'un domaine d'application à un autre. Cependant, le standard fournit la possibilité de rajouter de la sémantique dans un document 3D au moyen de métadonnées associées aux différentes parties du document.

X3D permet d'associer, au moyen de nœuds (`WorldInfo, Metadata[Set, Int, String, …]`), de l'information textuelle aux objets d'une scène 3D. Ces nœuds peuvent être attachés à n'importe quel élément dans le graphe de la scène. L'ajout de la sémantique dans les scènes X3D à l'aide de nœuds porteurs de la sémantique sous forme de métadonnées a été exploré entre autres par [Hetherington *et al.*, 2004] et [Polys *et al.*, 2004]. Dans [Hetherington *et al.*, 2004], les auteurs associent aux nœuds `WorldInfo` des informations sur la durée de vie des objets 3D afin d'ajouter une dimension temporelle dans les scènes 3D. [Polys *et al.*, 2004] ajoutent de l'information médicale sur les objets 3D représentant les différentes parties du corps humain en utilisant les éléments de la famille `Metadata[Set,Int,…]`. Nous remarquons que ces solutions sont limitées à une application spécifique dans laquelle la structure et l'utilisation des métadonnées sont connues à l'avance. Pour répondre à la variabilité des métadonnées dans différentes applications, les noeuds `WorldInfo` et `Metadata[Set,…]` doivent alors pointer vers des fichiers externes qui contiennent les métadonnées.

D'une manière similaire, les données géospatiales peuvent être introduites dans les scènes X3D en tant que métadonnées ou bien en tant qu'éléments de description géographique spécifiques à X3D. Des métadonnées géographiques (`coordinateSystem`, `ellipsoid`, `extent`, `resolution`, `originator`, etc.) peuvent être introduites en utilisant le nœud `GeoMetadata`. Des fichiers comportant des descriptions géospatiales externes peuvent être également pris en compte au sein de scènes X3D au moyen des URL des nœuds `GeoMetadata`.

Bien que le codage XML d'un fichier X3D facilite la récupération d'informations attributaires (position, apparence...), des requêtes complexes (incluant la sémantique) ne peuvent pas encore être traitées (par exemple, une requête de type : trouver l'étage accueillant le bureau des étudiants). Une description complémentaire, au moyen d'annotations, augmente l'utilisation des critères de recherche sur la base d'aspects sémantiques.

Après avoir présenté les possibilités d'expression des informations sémantiques dans le standard X3D, nous effectuons un survol des usages et solutions de gestion de la sémantique à travers des applications sémantiques 3D issus de différents domaines d'application organisés en

deux grandes catégories selon le moyen de stockage de la sémantique : à l'intérieur ou à l'extérieur des documents 3D ciblés.

2.3. La sémantique interne aux documents 3D

L'information sémantique dans les scènes 3D est généralement exploitée à des fins de visualisation. Généralement, des scripts *ad hoc* permettent de la matérialiser de façon dynamique à l'aide de primitives géométriques ou de libellés textuels générés à la volée suite à une interaction entre l'utilisateur et les objets de la scène [Polys *et al.*, 2004]. Parfois, l'apport sémantique sert dans le processus de transformation d'une scène en vue d'une diffusion temporisée [Hetherington *et al.*, 2004]. L'information sémantique peut également être utilisée afin de contrôler les zones ouvertes à l'exploration et d'aider l'utilisateur lors de la navigation au sein de la scène [Pittarello *et al.*, 2005]. Tout au long de cette section, nous présentons ces applications en détaillant, pour chacune, la manière dont on fait usage et dont on gère l'information sémantique.

2.3.1. PathSim [Polys *et al.*, 2004]

Dans [Polys *et al.*, 2004], les auteurs explorent la visualisation des parties du squelette humain (géométrie et apparence) en utilisant des modèles 3D ainsi que des données biomédicales relatives aux différentes parties (par exemple, la concentration de lymphocytes au niveau de la cavité buccale). Ces données médicales sont incluses dans le document modélisant la scène 3D à l'aide des nœuds de la famille `Metadata[*]` disponibles dans le profil de base du standard X3D.

2.3.2. 3D + Time [Hetherington *et al.*, 2004]

Dans [Hetherington *et al.*, 2004], les auteurs incluent la dimension temporelle dans les scènes 3D en ajoutant des métadonnées indiquant la durée de vie de chaque objet de la scène. La scène contient l'intégralité des primitives correspondant à toutes les périodes de temps modélisées.

```
<Group DEF="TimeTwo">
        <WorldInfo info="1950" title="Year"/>
        <WorldInfo info="Year of the gale and the roof was blown away" title="Description"/>
        <WorldInfo info="Brick" title="Walls"/>
        <WorldInfo info="None" title="Roof"/>
        <Transform USE="walls"/>
        ……
</Group>
```

Figure 2.6 Extrait de fichier X3D contenant la sémantique préconisée par [Hetherington et al.*, 2004].*

Dans la Figure 2.6, nous illustrons les modalités d'ajout d'information sémantique servant à temporiser la visualisation des éléments de la scène. Le groupe `Time Two` introduit la représentation d'un bâtiment tel qu'il était en 1950. Comme les scènes sont stockées en utilisant l'encodage XML de X3D, des feuilles de transformation XSLT sont appliquées afin d'enlever les objets et primitives géométriques dont la durée de vie ne correspond pas à l'intervalle sélectionné.

2.3.3. Semantic description of 3D environments [Pittarello *et al.*, 2006]

Dans [Pittarello *et al.*, 2006], les auteurs proposent une solution pour l'ajout d'une description sémantique de haut niveau venant compléter la définition géométrique de bas niveau des objets dans une scène X3D. Les auteurs partagent une partie de nos objectifs qui est d'établir des bases pour une utilisation avancée de données géométriques 3D dans le cadre de processus d'extraction, de sélection, d'interrogation, en utilisant des propriétés de haut niveau des objets contenus dans une scène 3D. L'approche proposée associe de la sémantique aux scènes 3D en intégrant deux standards, l'un issu du monde 3D (X3D) et l'autre issu du Web sémantique (RDF). Une partie de l'information est stockée avec la géométrie dans les documents X3D en faisant appel aux éléments supportant l'inclusion de métadonnées (`Meta`, `WorldInfo`, `MetadataSet`, …). L'autre partie décrivant les relations entre les différentes classes d'objets sémantiques est décrite en *RDF Schema*. Les auteurs visent à enrichir la description des objets 3D physiquement présents dans la scène, ainsi que celle des objets *virtuels* (une porte dans un mur) ou des régions de l'espace (une chambre) qui ne sont pas directement matérialisés dans la scène. Par ailleurs, une description à plusieurs niveaux est considérée afin de pouvoir interroger la scène selon une granularité choisie.

Les auteurs font la distinction entre les objets graphiques définis par des primitives géométriques X3D/VRML et les objets sémantiques auxquels ils s'attachent pour ajouter du sens. Un ou plusieurs objets géométriques peuvent être associés à un objet sémantique dont le sens est partagé par l'auteur et l'utilisateur final de l'annotation. La transformation d'un objet géométrique en objet sémantique se fait par l'introduction dans la définition de l'objet d'un nœud `MetadataSet` dont l'attribut `reference` indique l'identité de l'objet et l'attribut `name` la classe d'objet sémantique (voir ligne 5 de la Figure 2.7). Un complément d'information sémantique peut être ajouté en utilisant les nœuds `MetadataString` dont l'attribut `name` indique le nom de la propriété de l'objet et l'attribut `value` sa valeur (voir lignes 6-7 de la Figure 2.7). Le même mécanisme est employé pour décrire les relations entre objets sémantiques. L'attribut `name` d'un `MetadataString` inclu dans la définition de l'objet sémantique source, indique le nom de la relation et l'attribut `value` indique l'identifiant de l'objet cible. Les concepts, les propriétés et les relations sont déclarés dans une ontologie externe (voir les lignes A-L de la Figure 2.7) liée au document X3D à travers un nœud `meta` présent dans l'entête `head` du document (voir ligne 2 de la Figure 2.7).

X3D

```
01: <head>
02:     <meta name="ontology" content="rooms.rdfs"/>
03: </head>
04: <Transform ...> …
05:     <MetadataSet name="wall" reference="wall01">
06:       <MetadataString name="sharedBy" value="bedroom"/>
07:       <MetadataString name="sharedBy" value="loggia1"/>
08:     </MetadataSet>…
09:     <Shape DEF='Box4'> ... </Shape>
10: </Transform>
```

OWL

```
A:<owl:Class rdf:about="#wall">
B:    <rdfs:subClassOf>
C:        <owl:Restriction>
D:            <owl:onProperty>
E:                <owl:ObjectProperty rdf:about="#containedBy"/>
F:            </owl:onProperty>
G:            <owl:someValuesFrom>
H:                <owl:Class>
I:                <owl:oneOf rdf:parseType="Collection">
J:                    <owl:Class rdf:ID="room"/>
K:                    <owl:Class rdf:about="#loggia"/>
L:                    ...
```

Figure 2.7 Transformation d'un objet géométrique en objet sémantique [Pittarello et al.*, 2006].*

Dans le cas d'un objet sémantique virtuel, qui n'a pas de correspondant géométrique direct dans le document 3D définissant la scène 3D (une chambre est matérialisée par quatre murs, chaque mur étant représenté par une primitive `Box`), il faut introduire un nouvel élément dans la scène afin de pouvoir ajouter l'information relative aux objets sémantiques virtuels. Le moins intrusif semble être le nœud `WorldInfo` qui, par défaut, accueille des métadonnées sur la

scène. Ainsi, on inclut dans la définition du nœud `WorldInfo`, des nœuds `MetadataSet` afin de définir les objets sémantiques virtuels (voir ligne 2 de la Figure 2.8). Ensuite, les composants de l'objet virtuel vont se déclarer eux mêmes comme faisant partie de l'objet virtuel à l'aide d'une relation *sharedBy* ou *containedBy* (comme illustré dans la ligne 6 de la Figure 2.8).

```
01: <WorldInfo ... >
02:     <MetadataSet name="virtual objects">
03:         <MetadataSet name="room" reference="bedroom">
04:         <MetadataString name="description" value="room for sleeping"/>
05:         <MetadataString name="boundedBy" value="bedroomSpace"/>
06:         <MetadataString name="containedBy" value="house1"/>
07:     </MetadataSet>
08: ...
09: </MetadataSet> </WorldInfo>
```

Figure 2.8 Exemple de description d'un objet sémantique virtuel [Pittarello et al.*, 2006].*

Une particularité de l'approche proposée par ce travail consiste dans la description explicite de l'usage de l'espace au sein d'une scène 3D. Un exemple simple est une porte qui, en plus de ses caractéristiques géométriques, définit un espace de passage reliant deux zones de l'environnement 3D. Les auteurs introduisent la notion de région spatiale (ou *Interaction Locus*) décrite par un ensemble d'annotations qui définissent l'usage de l'espace. Initialement proposé dans [Pittarello, 2003] pour mieux contrôler la navigation de l'utilisateur au sein de scènes 3D, le concept est repris dans [Pittarello *et al.*, 2006] sous le terme d'*espace sémantique* et sa description est enrichie afin de répondre à une large palette de besoins. Un espace sémantique est défini de manière explicite par un volume 3D qui correspond à son étendue et il peut être décomposé en sous-espaces afin d'aboutir à une organisation hiérarchique. Un espace sémantique est matérialisé par un nœud `ProximitySensor` qui permet de définir l'étendue de l'espace au moyen d'une boîte 3D et de regrouper la sémantique de l'espace à travers des nœuds `MetadataSet` suivant les mêmes conventions que les objets sémantiques. Le choix de cette construction a été réalisé dans les premiers travaux afin de pouvoir contrôler l'interaction avec l'utilisateur, car un capteur permet de déclencher une série d'actions lorsque l'utilisateur s'en approche. Une valeur spéciale *type* pour l'attribut `name` du nœud `MetadataString` permet de différencier deux types d'espaces sémantiques : celui de passage et celui d'interaction.

Ainsi, les auteurs aboutissent à une technique qui permet de caractériser un espace 3D et les objets qui le peuplent. Leur solution jouit d'un certain degré de généricité car elle ne s'appuie que sur les éléments déjà présents dans le standard X3D (`Meta`, `WorldInfo`, `MetadataSet`, `ProximitySensor`). Elle est également indépendante d'une application spécifique car les termes utilisés sont définis dans une ontologie externe. Cependant, les auteurs ne discutent pas la situation lors de laquelle deux ontologies partagent des noms de concepts identiques mais dont les significations diffèrent. À notre avis, des conventions de noms sont à mettre en place. De plus, si une scène est utilisée dans une application dont les besoins sémantiques sont différents de ceux initialement pris en compte, une modification manuelle de la scène est nécessaire. Un autre reproche que nous faisons à ce travail est qu'aucune aide n'est proposée aux experts d'un domaine mais novices en 3D qui veulent introduire de la sémantique dans les scènes 3D. Les conventions et les règles définies par les auteurs pour la description de la sémantique semblent relativement complexes pour un expert de domaine peu familier avec le langage X3D.

2.4. La sémantique externe aux documents 3D

Dans cette section nous analysons des applications 3D et environnements dédiés à la représentation de la sémantique sur un support externe, autre que les documents 3D contenant les objets caractérisés.

2.4.1. Active 3D [Cruz *et al.*, 2005]

[Cruz *et al.*, 2005] proposent une plate-forme pour la gestion de l'interaction entre différents acteurs (architectes, ingénieurs, utilisateurs finaux) lors de la conception et la maintenance de projets architecturaux 3D en mode collaboratif. Une base de données relationnelle assure la mise en correspondance entre les objets géométriques et leurs propriétés sémantiques. Ces informations sont extraites à partir de documents IFC[16] décrivant les projets architecturaux. Ces documents contiennent la géométrie et un ensemble d'informations sémantiques (propriétés et relations prédéfinies dans le standard IFC). Lorsque les utilisateurs s'intéressent à une activité spécifique, les objets dont la sémantique s'apparente à l'activité sont extraits de la base de données et une scène 3D est générée dynamiquement. La sémantique joue principalement un rôle de filtrage.

Cette solution manque selon nous de flexibilité en termes de gestion de la sémantique. En effet, des informations sémantiques autres que celles prévues par le standard IFC ne peuvent pas être associées aux objets 3D sans reconsidérer l'organisation dans son intégralité de la base de données sous-jacente à la plate-forme.

2.4.2. SEDRIS [Foley *et al.*, 1998]

Dans le domaine de la modélisation et de la simulation d'environnements de synthèse, la *Defence Advanced Research Project Agency* (DARPA) a initié en 1994 le projet SEDRIS[17] qui vise à offrir une solution au problème de la représentation et du transfert de données environnementales. Le projet propose un modèle de données qui couvre de manière exhaustive l'environnement physique (terrain, océan, atmosphère, espace) [Foley *et al.*, 1998]. Une application 3D issue du projet SEDRIS est présentée dans [Schaeffer *et al.*, 2002]. Un entrepôt d'objets *Synthetic Natural Environment* (SNE) (bâtiments, ponts, avions, vehicules, etc.) est construit au-dessus du modèle de données SEDRIS. Un ensemble prédéfini de requêtes par mots-clés est mis à la disposition des utilisateurs à travers une interface Web à base de formulaires.

Comme pour la proposition précédente SEDRIS ne peut pas couvrir d'autres aspects que ceux prédéfinis dans le standard. Cependant, il reste un standard qui peut être utilisé au sein d'autres propositions sémantiques afin de caractériser les propriétés des objets du monde réel.

2.4.3. ISAS [Oliverio *et al.*, 2007]

Integrated Situational Awarness System (ISAS) [Oliverio *et al.*, 2007] est une initiative de *Digital Worlds Institute – University of Floride*. ISAS est un système d'aide à la décision pour les situations de crises (environnementales, civiles ou humanitaires), fondé sur les ontologies et la réalité virtuelle. ISAS combine et met en correspondance les données issues de divers flux de données et capteurs. [Oliverio *et al.*, 2007] explorent l'utilisation d'un environnement de réalité virtuelle fondé sur des ontologies afin d'augmenter la rapidité de transfert et d'acquisition de l'information par les utilisateurs.

[16] Industrial Foundation Classes (http://www.iai-international.org)
[17] http://www.sedris.com

Les auteurs bâtissent leur proposition sur le constat que dans le monde de la réalité virtuelle, on ne doit pas seulement connaître la façon dont un objet est matérialisé mais également ce qu'il représente, ses propriétés, ses caractéristiques, ses liens avec les autres objets présents dans ce monde. Afin de répondre à ce besoin, dans ISAS, les auteurs s'appuient sur un système de gestion d'ontologies nommé Lyra [Beck, 2005] conçu pour les ontologies OWL-DL. Le monde virtuel est vu comme la projection d'une ontologie dans un espace 3D. Le système de gestion d'ontologies fournit les outils pour représenter et classer les objets, leurs propriétés et leurs relations.

Le système peut être utilisé par l'expert d'un domaine pour décrire une scène, avec un niveau de détail très précis, en instanciant les termes et les relations définis dans une ontologie relative à son domaine d'expertise, sans avoir aucune connaissance quant à la modélisation 3D. Par exemple, dans une tâche relative à la modélisation 3D d'un bâtiment, un architecte pourrait fournir toutes les informations sur le bâtiment en créant une ontologie du modèle de bâtiment sans se soucier de la représentation 3D des concepts.

Ces ontologies de domaine, dont une utilisée par les auteurs dans le cadre du projet ISAS est détaillée dans la Figure 2.9, peuvent ensuite être combinées avec une ontologie de représentation 3D construite sur les notions présentes dans des langages de représentations tels que VRML ou X3D. Le bâtiment *Rinker Hall* est vu comme un objet de la classe bâtiment académique (*academic building*), et il se trouve à proximité (*next to*) du bâtiment d'informatique (*computer science building*) et les deux bâtiments utilisent l'eau et l'énergie fournie par la structure *physical plant*.

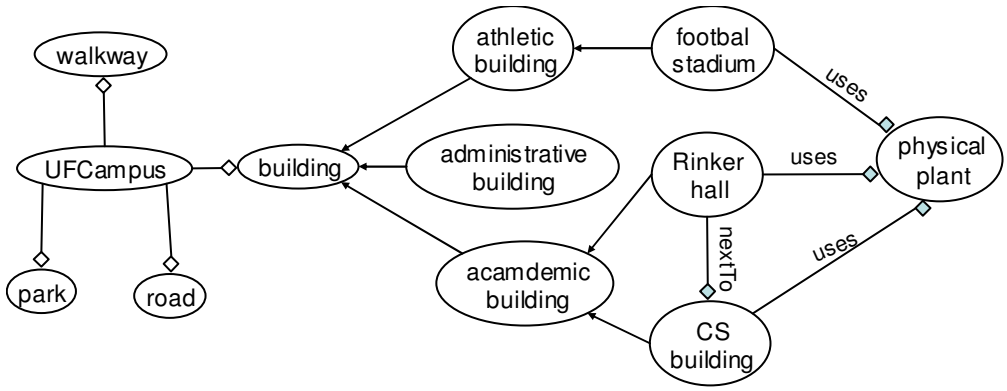


Figure 2.9 Partie de l'ontologie du campus proposé par [Oliverio et al.*, 2007].*

Ce découplage entre la géométrie et la sémantique permet ainsi la mise en évidence de certaines propriétés des objets en fonction de besoins applicatifs précis. Cette mise en évidence portent sur l'altération de leur apparence : transparence, renforcement de l'intensité des couleurs, choix de couleurs, …

Ce que nous reprochons à cette proposition est qu'elle ne traite que de l'information sémantique relative au domaine d'application et en aucun cas elle ne s'intéresse à une caractérisation complète de la scène qui inclut des informations sur la géométrie et l'apparence des objets. Ceci est une conséquence directe de l'approche de génération dynamique de scènes adoptée dans cette proposition. La géométrie et l'apparence d'un objet sont définies par le *mapping* entre les concepts sémantiques et une représentation 3D décidée par un concepteur. Cependant, nous considérons qu'il est important que dans la scène générée l'on dispose des informations sur l'ensemble des dimensions d'un objet 3D afin de pouvoir envisager par la suite une exploitation adéquate de celui-ci.

2.4.4. Semantic Metadata Creation [Halabala, 2003]

Le travail présenté dans [Halabala, 2003] s'intéresse à la description de scènes 2D (notamment les documents SVG) ou 3D (notamment les documents VRML). L'auteur recherche une structure adéquate pour stocker l'information sémantique et la manière dont un utilisateur *lambda* peut la créer. Le standard MPEG-7 est utilisé afin de stocker les graphes sémantiques rendant compte des propriétés et des relations des objets au sein d'une scène 3D. La solution choisie vise à apporter une contribution importante à la mise en place de filtrages sémantiques.

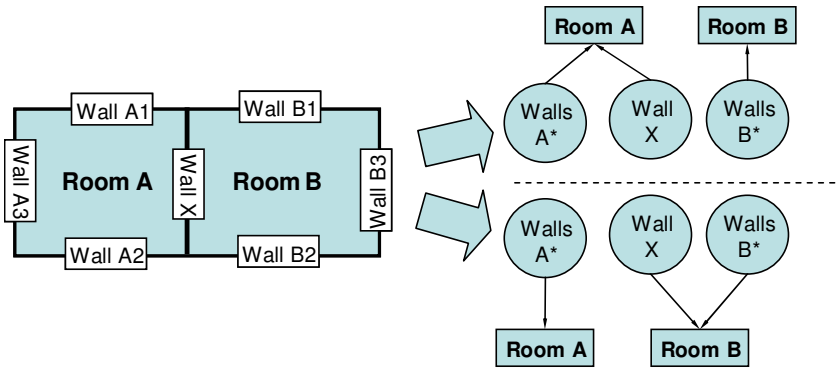


Figure 2.10 Limitations de VRML relatives à l'expression de l'information sémantique [Halabala, 2003].

L'auteur explore les possibilités d'extraire ces informations à partir de la structure logique des documents SVG/VRML ou à partir de certains éléments spécifiques de la scène, comme par exemple les prototypes (`PROTO`), ou les groups d'objets (`Group`), etc. Cependant, ces possibilités restent limitées car comme nous l'avons souligné dans le chapitre 1, le rôle principal de la structure logique d'une scène a été depuis toujours d'optimiser la visualisation de la scène. De plus, la structure logique d'une scène présente certaines limites, comme par exemple l'impossibilité de décrire que deux chambres partagent le même mur (voir Figure 2.10), ou bien qu'on peut passer d'une chambre à une autre. Si dans le sous-graphe correspondant à la deuxième chambre on veut faire apparaître le même mur (*Wall X*), on génère un duplicata de celui-ci.

Afin de combler ces manques, l'auteur propose la réalisation d'une description sémantique externe de type MPEG-7. Les outils de descriptions spécifiques MPEG-7 (entités, attributs et relations sémantiques) sont employés afin de construire un graphe sémantique caractérisant la scène (voir Figure 2.11). Deux types d'entités sont présentés dans le graphe sémantique : les objets (les murs `W*` et les chambres `R1`, `R2`) et les événements (le passage P). Les événements en MPEG-7 sont assimilés aux actions, voire relations, existant entre deux objets. Les murs sont les seuls objets qui existent également dans la scène 3D. Les chambres sont des objets virtuels construits à partir des murs environnants.

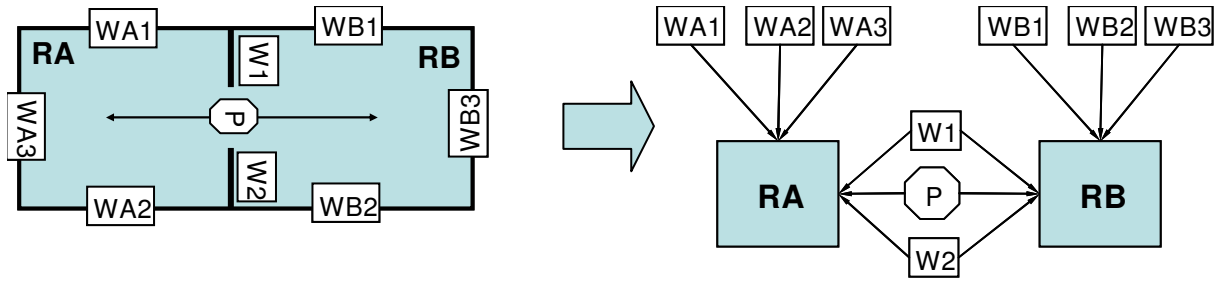


Figure 2.11 Scène VRML et son graphe sémantique d'après [Halabala, 2003].

L'auteur propose également un prototype pour annoter de façon interactive des modèles 2D/3D (SVG/VRML/X3D) et créer les fichiers MPEG-7 correspondants. Un objectif de cet outil est de rendre possible la navigation dans les deux sens, entre objet annoté (SVG/VRML/X3D) et entité lui correspondant (MPEG-7). Lorsque l'utilisateur clique sur l'entité `RA` dans la scène, les objets géométriques lui correspondant sont mis en évidence et vice-versa. Un autre aspect important est une visualisation adaptée des informations sémantiques présentes dans les fichiers MPEG-7. Même pour les petites scènes, les graphes sémantiques comportent beaucoup d'information. Afin de faciliter la tâche d'exploration de celle-ci et de limiter la surcharge cognitive, l'auteur met en place des niveaux de détail (qui s'appuient sur les relations de type *partie de*) et de niveaux d'abstraction (qui s'appuient sur les relations de type *est un*). Lors de la visualisation, des processus de filtrage prennent en compte ces niveaux de détail afin d'alléger la visualisation de la sémantique.

Les limitations de ce travail sont liées principalement aux faibles possibilités de localisation des objets géométriques dans la scène 3D. La seule possibilité d'y faire référence est l'identifiant de l'objet. Ceci rend impossible la description d'un objet sans identifiant. De plus, il n'y a pas de modèle sous-jacent qui permette d'envisager la construction d'un langage capable d'extraire et exploiter l'information sémantique stockée au sein de fichiers MPEG-7. Le travail se concentre uniquement sur la description sémantique des données. Cependant, le choix de MPEG-7 donne également l'opportunité de caractériser l'apparence et la forme d'un objet 3D à l'aide de descripteurs visuels (*DominantColor*, …) et de forme (*Shape3D*) prédéfinis dans MPEG-7.

2.4.5. AIM@SHAPE [Albertoni *et al.*, 2005]

Une proposition [Albertoni *et al.,* 2005] issue des travaux de recherche menés au sein du Réseau d'Excellence AIM@SHAPE[18] s'inscrit dans le domaine de la modélisation et de la gestion de formes. Une forme peut être une image, un modèle 2D ou 3D, une vidéo, etc. Parmi les occupations primordiales du Réseau d'Excellence AIM@SHAPE, on peut noter l'émergence d'outils pour extraire la sémantique implicite des formes, l'encodage et la formalisation des connaissances de domaine au sein d'ontologies dépendantes de contextes. Ces outils sont proposés comme une solution pour maîtriser une future explosion des données 3D caractérisées par une grande complexité et hétérogénéité des ressources en termes de propriétés géométriques, sémantiques, ainsi que par l'usage que l'on en fait.

Les travaux s'intéressant à la caractérisation de formes 3D, dont un bref aperçu est donné dans le chapitre suivant relatif à la recherche à base d'exemple, ont acquis une certaine maturité ces dernières années. Cependant, les diverses approches de caractérisation utilisent des descripteurs dont le sens est implicite et défini uniquement au sein d'une application précise. Le transfert de connaissances entre deux travaux utilisant des descripteurs différents n'est pas possible à ce jour, sauf si une solution logicielle spécifique à chaque cas de figure est codée manuellement. Dans [Attene *et al.*, 2005], les membres du Réseau d'Excellence AIM@SHAPE posent les bases d'une famille d'ontologies qui pourront jouer, dans le futur, le rôle de pivot facilitant la communication entre les diverses approches de caractérisation. Selon [Attene *et al.*, 2005], les caractéristiques des formes ne sont plus entièrement composées d'éléments de bas niveau, mais incluent aussi une description haut niveau facilitant les raisonnements sur les formes (réutilisation, transformation adaptée, etc.). La complexité et l'étendue du domaine rendent incongrue l'idée de construire une seule grande ontologie qui couvre tous les aspects liés à la vie d'une forme.

[18] http://www.aimatshape.net

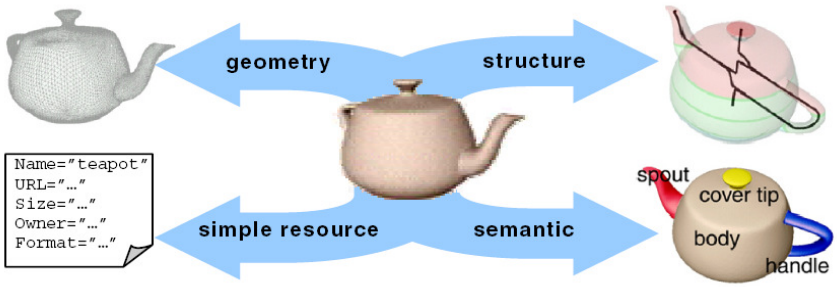


Figure 2.12 Description d'une forme d'après [Albertoni et al.*, 2005].*

Tout d'abord, les auteurs s'intéressent à une ontologie pour l'Acquisition et la Reconstruction (AR) de formes 3D. Cette ontologie définit un certain nombre de concepts, propriétés et relations spécifiques à la tâche d'AR. Les métadonnées associées aux formes 3D, dont les noms et valeurs sont définis au sein de l'ontologie (AR), permettent de renseigner sur : l'acquisition de la forme (dispositif de capture, techniques de capture, etc.), l'information implicite de la forme (ce qu'elle représente) et les éventuels futurs usages de la forme. Les métadonnées sont employées afin de décrire plusieurs catégories de caractéristiques : 1) simples – dans quelle catégorie d'objets se trouvent l'objet annoté ; 2) géométriques – afin d'assurer un bon rendu ; 3) de structure et 4) ce que l'objet représente (à des fins de reconnaissance et classification). La Figure 2.12 illustre la description d'une théière selon les types de caractéristiques énoncées ci-dessus.

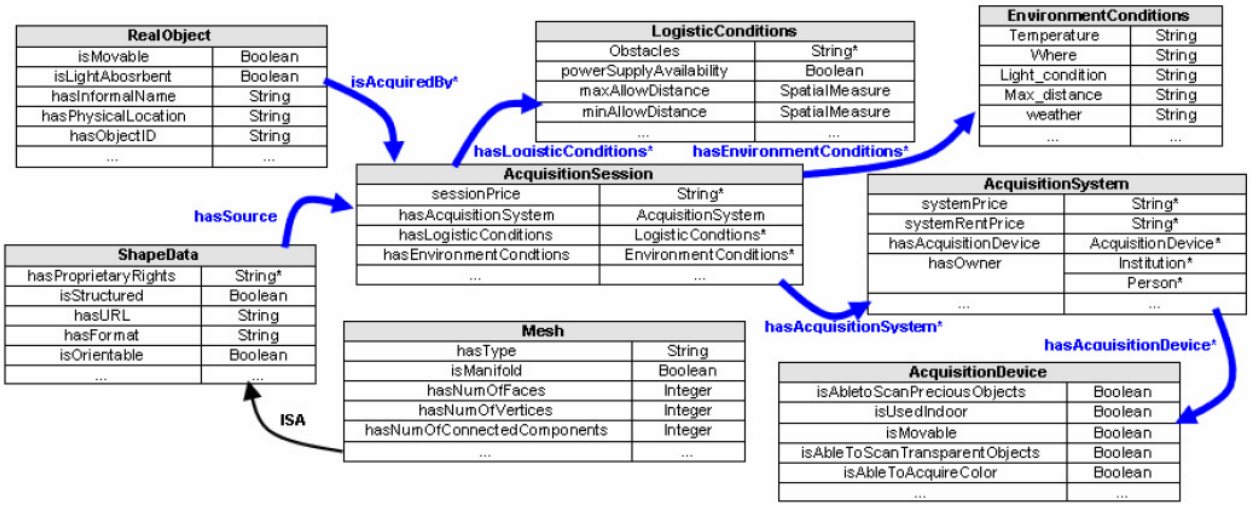


Figure 2.13 Description de l'acquisition d'une forme en utilisant l'ontologie AR [Albertoni et al.*, 2005].*

À titre d'exemple, les auteurs présentent une partie (illustrée dans la Figure 2.13) de l'ontologie AR relative au processus d'acquisition. Les tableaux représentent les concepts. Les lignes représentent les propriétés et les relations instanciées pour chaque concept. Cette partie couvre la phase d'acquisition, notamment le concept *AcquisitionSession* et l'ensemble de ses propriétés et relations. Ce concept est relié à un système d'acquisition (*AcquisitionSystem*) dans des conditions matérielles (*LogisticConditions* – type de lumières, existence de zone d'occlusions, etc.) et environnementales (*EnvironmentCondition* – à l'extérieur, à l'intérieur, humidité, etc.) donnés. Un système d'acquisition est caractérisé par une collection d'attributs (tels que le prix d'acquisition, etc.). Il est constitué d'un ensemble de dispositifs d'acquisition (*AcquisitionDevice*). La description d'une session d'acquisition documente l'acquisition d'un objet réel (*RealObject*) et la génération d'une forme (*ShapeData*) en utilisant un système

d'acquisition particulier (*AcquisitionSystem*). Le concept *ShapeData* contient des informations sur le format, sur l'URL du document qui matérialise la forme, sur la source (session d'acquisition), etc. Ce concept est spécialisé afin de rendre compte de certaines catégories spécifiques de formes (images 2D, modèles 3D *Mesh*, etc.). Le concept de forme 3D *Mesh* est ensuite dérivé afin de couvrir une large variété de formes 3D (*SurfaceMesh, FreeForm, PointSet, etc.*). La liste complète peut être trouvée dans [Attene *et al.*, 2005]. Chacun de ces concepts supporte un ensemble de propriétés spécifiques (le nombre de points pour le *PointSet*, le nombre de sommets et d'arêtes pour un *SurfaceMesh*) qui permet de caractériser les formes de la manière la plus appropriée.

Cette proposition privilégie la description de la phase d'acquisition d'une forme 3D et ne s'intéresse pas à l'ajout d'information sémantique relative aux divers domaines d'application où la forme 3D peut-être réutilisée. Pourtant, l'introduction de concepts pour différencier la forme 3D (`Mesh`) et l'objet réel modélisé par la forme (`RealObject`) offre des bonnes prémisses pour l'ajout et l'exploitation d'information sémantique au niveau local (de la forme) et général (de l'objet réel).

2.4.6. OntoWorld [Mansouri, 2005]

Dans [Mansouri, 2005], l'auteur s'attache à la conception d'un modèle supportant l'interrogation flexible et relative à un domaine spécifique dans le cadre des environnements virtuels. Les performances du processus d'interrogation dépendent principalement de la qualité des métadonnées saisies pendant le processus d'annotation. Cette information est généralement saisie par des experts de domaine et non par les créateurs de l'environnement virtuel. Le processus d'annotation n'est pas limité à un certain nombre de propriétés prédéfinies (type, couleur, hauteur, largeur, profondeur, location…), l'utilisateur est censé pouvoir ajouter autant d'informations qu'il le souhaite. La description ne porte pas uniquement sur les propriétés locales des objets, mais contient également des connaissances relatives au domaine dans lequel les objets sont définis.

Le modèle proposé par Mansouri s'appuie sur l'approche OntoWorld [Kleinermann *et al.*, 2005]. OntoWorld, créé par le groupe de recherche VR-WiSE de *Vrije Universiteit Brussel*, s'intéresse à l'implication des experts de domaine dans la construction des environnements virtuels. L'approche utilise les ontologies comme un moyen intuitif et orienté domaine pour la génération des mondes virtuels. L'auteur l'étend afin de permettre d'annoter des mondes virtuels déjà créés par d'autres moyens.

L'annotation peut être réalisée à plusieurs niveaux. Lorsque l'annotation est relative à un concept, toutes les instances de ce concept héritent de l'information sémantique associée au concept. L'annotation de concepts peut être vue comme une méthode pour donner des descriptions par défaut à toutes les instances. Chaque concept inclut une définition textuelle qui éclaire l'utilisateur sur le sens du concept. Ensuite, plusieurs descriptions à base de couples {mot-clé, valeur} peuvent être associées aux concepts (par exemple : {material, bricks}, {texture, smooth}).

De plus, un système de règles permet de calculer les valeurs précises d'un couple lors de la création d'une nouvelle instance. Une règle est constituée de quatre éléments : un adjectif qualitatif, la propriété concernée, un critère (plus grand, plus petit, etc.), la valeur seuil. Lors de l'instanciation d'une nouvelle entité, le système compare la valeur de la propriété aux valeurs seuil, et il associe l'adjectif à l'instance si le critère est respecté. Par exemple, on veut dire qu'une montre (*une instance*) est chère (*l'adjectif*) si son prix (*la propriété*) est plus grand que (*le critère*) 100 euros (*valeur seuil*). Cette approche, qui repose sur l'association des adjectifs en

considérant la nature des concepts, a l'intérêt de proposer différentes échelles de valeurs : si une montre est *chère* lorsque son prix dépasse 100 euros ce n'est pas le cas d'une voiture.

Par rapport à OntoWorld, l'auteur prend en compte la notion d'objet complexe en utilisant le concept de *conteneurs*. À la différence des objets complexes, les *conteneurs* ne connaissent pas leur fils, car ce sont les instances elles-mêmes qui déclarent leur rattachement à un *conteneur* particulier.

Lors de l'annotation d'une instance, un maximum d'information est collecté et ajouté de manière automatique à la description MPEG-7. Cette description inclut les descriptions héritées de concept parent, les informations relatives aux propriétés de l'instance, son comportement ainsi que les annotations générées par les règles validées.

```
<MultimediaContent>
 <Multimedia id="FrontWall">
  <TextAnnotation type="objectType">
   <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="en">Wall</FreeTextAnnotation>
  </TextAnnotation>
  <TextAnnotation type="mainDescription">
   <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="en">The front wall</FreeTextAnnotation>
  </TextAnnotation>
  <TextAnnotation type="ContainedIn">
   <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="en">House</FreeTextAnnotation>
  </TextAnnotation>
  <TextAnnotation type="Material">
   <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="en">Bricks</FreeTextAnnotation>
  </TextAnnotation>
  <TextAnnotation type="Texture">
   <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="en">Smooth</FreeTextAnnotation>
  </TextAnnotation>
  <TextAnnotation type="LeftOf">
   <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="en">RightWall</FreeTextAnnotation>
  </TextAnnotation>
  <TextAnnotation type="Color">
   <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="en">red</FreeTextAnnotation>
  </TextAnnotation>
  <TextAnnotation type="Height">
   <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="en">4</FreeTextAnnotation>
  </TextAnnotation>
  <TextAnnotation type="High">
   <FreeTextAnnotation phoneticAlphabet="sampa" xml:lang="en">Yes</FreeTextAnnotation>
  </TextAnnotation>
 </Multimedia>
</MultimediaContent>
```

Figure 2.14 Extrait de document MPEG-7 de description d'instances d'après [Mansouri, 2005].

Chaque objet annoté est représenté dans MPEG-7 comme un élément d'une description de type `MultimediaContent` (voir Figure 2.14). L'objet est matérialisé par un élément `Multimedia` muni d'un identifiant unique (*id*). Cet identifiant correspond au nom de l'objet géométrique associé dans le fichier VRML (l'attribut *DEF*). Ensuite, les caractéristiques de chaque objet sont introduites au moyen d'annotations textuelles MPEG-7 (`TextualAnnotation`). Le nom de la caractéristique est précisé par la valeur de l'attribut `type` de l'élément `TextualAnnotation`, sa valeur étant exprimée comme une `FreeTextAnnotation`. Le choix de `FreeTextAnnotation` a été fait parce qu'il supporte l'utilisation des espaces de noms ce qui facilite l'utilisation de termes issus de plusieurs ontologies. Trois types d'objets sont matérialisés dans les fichiers MPEG-7 générés : les concepts, les conteneurs et les instances. Les caractéristiques spéciales `objectType` et `mainDescription` définissent la nature de chaque élément `Multimedia`. Pour les concepts et les conteneurs, la caractéristique `objectType` vaut respectivement « Concept » ou « Container », tandis que pour les instances elle indique le nom du concept parent. Les règles sont décrites sous les éléments dont le `type` est *Rule* et chacune des quatre parties d'une règle doit être définie suivant l'ordre : adjectif, propriété, critère, seuil. L'appartenance à un containeur et la

participation dans une relation spatiale sont définies en utilisant des caractéristiques dont les noms correspondent au nom du containeur, respectivement aux noms des relations (par exemple, *ContainedIn* pour une relation d'inclusion, et *LeftOf* pour une relation spatiale). Les valeurs des caractéristiques correspondent à l'objet cible. Le modèle ne supporte que des relations binaires.

L'originalité de ce travail consiste dans le fait qu'il ne restreint pas l'utilisation de leur outil d'annotation sémantique en retenant uniquement un ensemble prédéfini d'informations sémantiques. Ceci rend possible l'utilisation de leur proposition dans de nombreux domaines d'application. Cependant, nous trouvons que la gestion de ces informations n'est pas suffisamment maîtrisée. Les informations ne sont pas organisées en plusieurs catégories par rapport au domaine pour lequel elles sont définies. Ce manque de structuration rend difficile la visibilité et l'accès aux propriétés sémantiques lorsqu'un ensemble conséquent d'annotations est associé à une donnée 3D.

2.4.7. SeVEn [Otto, 2005]

Un autre travail relatif à la gestion et l'utilisation de la sémantique au sein de scènes 3D est issu du projet SeVEn (*Semantic Virtual Environment*) [Otto, 2005]. SeVEn est un cadre pour le développement d'applications qui peuvent accéder et interagir avec les Environnements Virtuels Sémantiques (SVE). Pour [Otto, 2005], un SVE est une vue abstraite d'un environnement virtuel existant. Cette vue est construite en utilisant des techniques sémantiques du Web pour décrire la sémantique et les interactions possibles au sein de cet environnement réel.

SeVEn doit fournir les services qui couvrent les principaux besoins d'un SVE :

- la gestion des descriptions, des schémas RDF associés et toute autre information relative au SVE;
- l'interaction avec l'environnement à travers des événements envoyés ou reçus sur les canaux de communication environnementaux.

Les deux fonctionnalités s'appuient sur des modèles à composants afin de faciliter la réutilisation. Dans la suite, nous présentons brièvement les fonctionnalités liées à la gestion de l'information sémantique.

Une description SeVEn est un graphe RDF [Manola *et al.*, 2004] qui expose de manière explicite la sémantique de l'environnement : les entités s'y trouvant, leurs types, leurs propriétés ainsi que les relations entre ces entités. La description d'un SVE peut également inclure des liens vers des sources d'informations externes telles que des descriptions complémentaires d'entités, des modèles de présentation, des environnements virtuels connexes. Cet ensemble d'information est destiné aux agents (logiciel ou humain) qui, grâce à la description sémantique de l'environnement, peuvent exécuter des opérations complexes (par exemple, la recherche…). Du fait que ces agents travaillent au niveau sémantique, ils peuvent être réutilisés par ailleurs dans d'autres applications issus du même domaine même si les données et leurs représentations (2D – une photo de la table *versus* 3D – le modèle 3D de la table) ne sont pas identiques.

Un ensemble de concepts et relations spécifiques à la description de SeVEn a été introduit grâce aux schémas RDF et regroupé sous l'espace de noms `sve`. Dans la Figure 2.15 nous reproduisons une description d'environnement sémantique 3D. Parmi les concepts, on retient la classe `sve:Environment` qui décrit l'environnement dans son ensemble. Un environnement peut être lié à un autre grâce à la relation `sve:link`. Les éléments d'un environnement sont répertoriés grâce à la relation `sve:contains`. Chaque élément de l'environnement est associé à une instance de la classe `sve:Entity` (dans l'exemple ci-dessus il y

a une entité de type `fn:Table` et une de type `bib:Book`). Les entités sont caractérisées par une ou plusieurs matérialisations (2D, 3D, texte) qui peuvent leur être associées à l'aide de la propriété `sve:Presentation`. De plus, des informations supplémentaires relatives à la position et aux propriétés géométriques de l'entité dans le monde réel sont prises en compte grâce aux éléments suivants : `sve:transform`, `sve:bounds`, `sve:origin`, `sve:pozX`, `sve:pozY`, `sve:pozZ`.

Figure 2.15 Description partielle d'un SVE d'après [Otto, 2005].

Ce travail présente des caractéristiques très intéressantes parmi lesquelles nous notons la possibilité de relier à travers la sémantique des objets issus d'univers média distincts (2D, 3D, texte). Cependant, nous trouvons relativement pauvre le nombre de concepts propres à SeVEn introduits sous l'espace de noms `sve`. Les concepts ne visent qu'un ensemble minimal de caractéristiques de données 3D. En effet les concepts ne s'intéressent ni aux caractéristiques géométriques ni à l'apparence des objets 3D.

2.4.8. SVE [Gutierrez *et al.*, 2005]

Dans le domaine des applications virtuelles interactives, [Gutierrez *et al.*, 2005] proposent une solution qui se préoccupe tant de la caractérisation sémantique de l'environnement virtuel que de la manière dont les objets (avatars ou objets intelligents) sont contrôlés afin de répondre aux interactions de l'utilisateur.

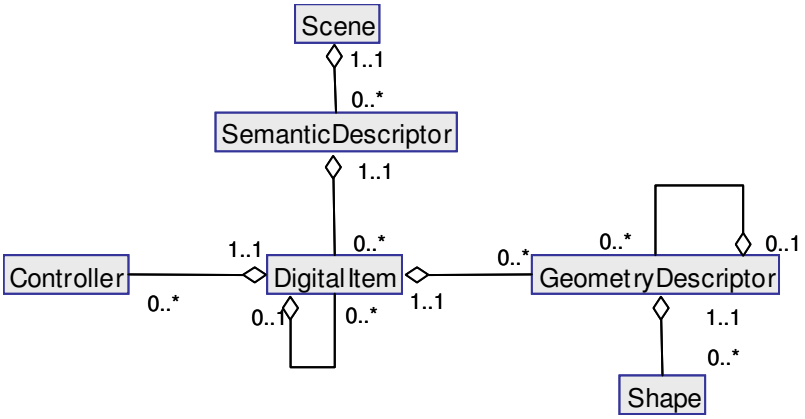


Figure 2.16 Modèle pour la sémantique d'un environnement virtuel interactif [Gutierrez et al.*, 2005].*

Le modèle proposé est illustré dans la Figure 2.16. La classe *Scene* est le principal conteneur du modèle et elle référencie les objets sémantiques de la scène introduits par des descripteurs sémantiques (*SemanticDescriptor*). Un *SemanticDescriptor* fournit l'information sémantique sur certaines parties de la scène (*DigitalItem*s). Les contrôleurs (*Controller*) décident de la forme géométrique et des moyens d'interactions associés aux *Digital Items* visibles dans la scène par rapport à un contexte donné.

Les descripteurs sémantiques constituent la principale source d'information qui guide les contrôleurs dans leur choix. Ceci est dû au fait que les descripteurs sémantiques contiennent toutes les informations relatives à l'objet et aux relations qu'il entretient avec les autres éléments de la scène. Le modèle reflète également les relations existantes entre les objets de l'environnement virtuel. Les descripteurs sémantiques caractérisent chaque objet/avatar dans la scène et permet de définir le graphe de scène utile pour l'optimisation du rendu ainsi que l'extraction des informations sur le contenu. Les objets peuvent contenir à leur tour d'autres objets ou peuvent être reliés à d'autres objets (par exemple il peut y avoir deux objets qui se déplacent ensemble).

Ce modèle a été construit autour de la principale caractéristique d'une entité du monde virtuel – sa géométrie (*Shape*). Les descripteurs géométriques (*GeometricDescriptor*) d'un objet décrivent le type de géométrie effective associée à l'objet 3D : surface déformable, volume articulé (jointures et segments), etc. En fonction d'un contexte donné, la géométrie d'une entité peut changer, ainsi à chaque objet on peut associer plusieurs descripteurs géométriques.

Le modèle est complété par la classe *Controller* qui décrit les différentes méthodes permettant de contrôler les objets (*Digital Items*) : les contrôles interactifs, les animations autonomes, etc.

Nous trouvons que cette proposition manque d'un deuxième niveau plus générique de description de la sémantique. Toutes les propriétés sémantiques sont uniquement définies localement par rapport à une scène spécifique. Alors qu'il serait utile que plusieurs objets représentant une même entité du monde réel dans plusieurs environnements virtuels, partagent un même ensemble de propriétés concernant l'entité. Une autre faiblesse de cette proposition est qu'elle ne traite pas de propriétés visuelles liées à l'apparence des objets et s'intéresse exclusivement aux informations sémantiques et géométriques des objets 3D.

2.5. Synthèse

Dans ce chapitre, nous avons tout d'abord analysé les moyens d'expression et d'exploitation des informations sémantiques dans un contexte général multimédia (MPEG-7) et Web (RDF/OWL). Ensuite, nous avons fait un état des lieux des propositions issues de la communauté 3D s'intéressant à l'exploitation de la sémantique dans le cadre de domaines d'application particuliers.

Les propositions qui s'appuient sur **une approche à base de sémantique externe semblent plus flexibles et capables d'évoluer plus facilement**, car indépendantes de l'encodage X3D de la scène. Cependant, elles souffrent d'**un manque de puissance d'expression en ce qui concerne la localisation des parties de document** concernées par l'information sémantique. En effet, le lien entre l'information sémantique et la partie concernée du document 3D se fait exclusivement à base de l'identifiant de la partie respective dans le document 3D. Ceci réduit les possibilités d'expression, lorsque les parties ciblées ne sont pas introduites dans le document source par des identifiants bien définis (par exemple, des nœuds possédant un attribut `DEF` en X3D) ou lorsque l'organisation de la scène imposée par le concepteur ne convient pas à la vision propre à l'utilisateur (qui ajoute l'information sémantique).

La plupart des propositions considèrent uniquement un seul niveau sémantique, ne faisant pas **la différence entre la sémantique du fragment multimédia** au sein du document 2D ou de la scène 3D et la sémantique de **l'entité physique/réelle représentée par le fragment** en question.

Aucune des propositions que nous avons étudiées **ne s'intéresse à la caractérisation d'une donnée 3D dans sa globalité**, couvrant les dimensions géométrique, d'apparence, et sémantique. AIM@SHAPE [Albertoni *et al.*, 2005] et SVE [Gutierrez *et al.*, 2005] se démarquent des autres en s'attachant à la caractérisation de la géométrie. Cependant, SVE [Gutierrez *et al.*, 2005] n'accorde pas d'attention à la caractérisation de l'apparence. AIM@SHAPE [Albertoni *et al.*, 2005] ne traite pas de la sémantique liée aux potentiels domaines d'application et considère la forme 3D comme étant à l'extérieur de toute scène 3D.

Nous remarquons également que la plupart des propositions étudiées dans cette section sont restreintes à un cadre spécifique dans lequel la nature (structure) et l'utilisation des métadonnées sont connues à l'avance. **La réutilisation de la sémantique** dans des applications autres que celles pour lesquelles elle a été initialement définie **reste limitée**.

Dans le chapitre suivant, nous nous intéressons à la recherche et à la réutilisation des données 3D. Nous analysons le rôle des informations (de bas niveau) extraites directement des caractéristiques (géométrie, apparence) de l'encodage de données 3D par rapport à la sémantique. Nous explorons également les moyens de réutilisation par génération dynamique de scènes.

3. La recherche et la réutilisation de données 3D

Ce chapitre est consacré à l'étude des moyens théoriques et pratiques mis en œuvre afin de faciliter la gestion de données 3D en se focalisant sur : la recherche et la réutilisation de données 3D.

Nous commençons par nous intéresser aux méthodes et critères de recherche relatifs à l'information 3D. Ensuite, nous étudions les propositions existantes en termes de réutilisation.

3.1. La recherche

La recherche du contenu joue un rôle central dans le cadre des processus de gestion des données, quelque soit leur nature. Accéder aux données en vue d'un traitement quelconque (visualisation, diffusion, transformation, etc.) peut se faire selon trois modalités :

a) en donnant l'emplacement exact de la donnée – cas peu fréquent lorsque la masse d'information devient conséquente ;

b) en procédant à une recherche en indiquant de façon explicite les propriétés des données – la recherche à base de propriétés peut être assimilée à une recherche classique par mots-clés [Salton, 1989][Baeza-Yates, 1999] ;

c) en proposant des données similaires – la recherche à base de données similaires s'apparente aux termes déjà consacrés : recherche par l'exemple ou recherche par contenu [Gudivada *et al.*, 1995][Rui *et al.*, 1997].

La *recherche à base de propriétés* permet à l'utilisateur de synthétiser et d'exprimer sa requête en utilisant en partie le langage naturel. Cependant, selon les possibilités d'expression et d'inférence du moteur de recherche utilisé, cette tâche peut se révéler assez difficile. Des

problèmes liés aux faux amis, à la polysémie ou la synonymie peuvent rendre difficile la formulation adéquate des besoins de l'utilisateur. De plus, ce type de recherche demande une étape préliminaire qui consiste à indexer les contenus 3D. Chaque objet 3D est associé à un ensemble de termes qui le caractérise. Cette étape est généralement difficilement réalisable sans intervention humaine.

La *recherche par l'exemple* a l'avantage de s'abstraire d'une formulation à base de termes qui peuvent prêter à confusion, car l'indexation des données de l'entrepôt fouillé est identique à celle appliquée à l'exemple fourni. Dès que les mêmes conventions de caractérisation sont utilisées, les problèmes spécifiques à la recherche à base de mots-clés sont réduits. Afin de caractériser un modèle, on analyse ses propriétés géométriques et/ou son apparence et on établit un ensemble de mesures mathématiques (vecteurs, projections, descripteurs de surfaces, etc.). Un recueil des principales techniques est présenté dans la section suivante. Cependant, les éventuelles similitudes de forme qui existent entre des objets différents (une pyramide et un toit de maison, par exemple) baissent la pertinence des items retrouvés. De plus, il est nécessaire que l'utilisateur construise cet exemple et que l'encodage de ce document exemple soit supporté par le moteur d'indexation, ce qui en réduit la généricité.

Le choix entre les deux approches dépend beaucoup du contexte applicatif par rapport auquel on effectue la recherche. Par exemple, lorsqu'au sein d'une communauté (entreprise, groupe de travail, etc.) il existe un vocabulaire limité et bien défini pour exprimer les besoins dans un domaine spécifique, une approche à base de mots-clés peut se révéler assez efficace. Effectivement, une augmentation du taux de rappel peut être observée car le nombre de problèmes liés aux sens des termes employés est évincé du fait que, lors de l'indexation et de la recherche, les termes utilisés sont porteurs de la même sémantique. Ce type de recherche nécessite une indexation manuelle de chaque objet de la collection. Des techniques qui automatisent l'extraction du sens à partir d'images ou de scènes 3D ne sont pas encore au point. Des solutions semi-automatiques d'indexation à base de mots-clés ont également été explorées [Wenyin *et al.*, 2001][Handschuh *et al.*, 2002]. L'utilisateur indexe un certain nombre d'objets de l'entrepôt. Ensuite, les termes sont associés par diffusion aux autres objets de l'entrepôt suivant le degré de similarité entre l'objet source et l'objet concerné. Le degré de similarité est calculé par rapport à la différence entre la signature des objets.

Dans la sous-section suivante nous analysons différents critères de recherche que nous avons rencontrés dans la littérature. Nous nous intéressons premièrement aux critères utilisés dans les recherches à base d'exemples qui reposent sur la caractérisation de données au niveau signal. Ensuite, nous passons en revue les approches utilisant des critères sémantiques. Pour conclure, nous examinons les propositions qui peuvent être considérées comme offrant de bonnes prémisses à la construction d'un langage d'interrogation spécifique aux documents 3D.

3.1.1. Critères de recherche au niveau signal

La plupart des critères de recherche au niveau signal concernant les objets 3D sont généralement liée à la caractérisation de la surface des objets. Parmi les propositions de caractérisation des surfaces 3D on retrouve deux grandes familles qui reposent respectivement sur :

- les *projections* des surfaces 3D pour obtenir une image 2D – des techniques propres à l'analyse d'image sont appliquées par la suite pour statuer sur la similarité de deux surfaces 3D,

- la *construction* d'une structure (vecteur, graphe, …) contenant certaines des propriétés géométriques d'une surface 3D – ensuite pour mesurer la similarité entre deux surfaces 3D, on évalue la distance métrique entre les deux structures.

L'étude que nous présentons par la suite met également en évidence des combinaisons entre ces deux approches : l'analyse des propriétés géométriques sert à construire une image 2D qui est caractérisée à l'aide de descripteurs spécifiques aux images 2D. Ensuite, la mesure de similarité entre deux images est généralement calculée en extrayant un vecteur caractérisant les images (contours, densité, …) de certaines régions de l'image.

Nous avons aussi identifié des méthodes s'intéressant aux autres propriétés (d'apparence, topologiques) des surfaces 3D afin d'augmenter la pertinence du processus de recherche.

Dans la section suivante nous dressons un succinct état de l'art sur les divers types d'indexation des propriétés géométriques et des critères de recherche associés. Ensuite, nous considérons les apports de l'indexation de l'apparence et de la topologie des contenus 3D.

3.1.1.1. Indexation des propriétés géométriques des objets 3D

Le processus d'indexation des propriétés géométriques implique la caractérisation automatique par le biais d'outils d'analyse mathématique. Les travaux que nous présentons par la suite couvrent un large champ d'outils et de critères d'analyse.

Afin d'en donner un aperçu, nous avons retenu pour l'étude les méthodes s'intéressant à :

- la description spectrale de formes 3D [Zaharia *et al.*, 2001][Zaharia *et al.*, 2002],
- la caractérisation de projection 2D des formes [Johnson *et al.*, 1999],
- l'analyse de relations spatiales entre les régions d'une forme [Del Bimbo *et al.*, 1998],
- la caractérisation de projections sphériques d'une forme 3D [Kazhdan *et al.*, 2003].

Dans le cadre du processus de standardisation de l'indexation à base de contenu, dans la communauté MPEG-7, [Zaharia *et al.*, 2001] proposent l'utilisation d'un descripteur de forme *3D Shape Spectrum Descriptor* (3DSSD). Le descripteur 3DSSD caractérise de manière intrinsèque la forme d'un maillage 3D comme la distribution de l'index de forme [Koenderink, 1990] dans tous les points de la surface. L'index de forme est un attribut local de la géométrie d'une surface 3D, calculé à partir de l'angle entre les deux courbures principales de la surface dans un point précis de la surface.

La valeur de cet index varie entre 0 et 1 et n'est pas définie pour les surfaces planes. Suivant sa valeur, la forme de la surface approche un ombilic minimal (0.0), un creux (0.25), une selle (0.5), une crête (0.75) ou un ombilic sphérique (1.0) (voir Figure 3.1). Le spectre de surface est défini comme la distribution de l'index de forme dans tous les points du maillage. Les auteurs considèrent une discrétisation de la distribution en utilisant entre 10 et 100 classes dont les valeurs sont réparties uniformément dans l'intervalle [0,1]. La similarité entre deux surfaces est mesurée comme la différence entre les spectres de surface définis comme les diagrammes de distribution.

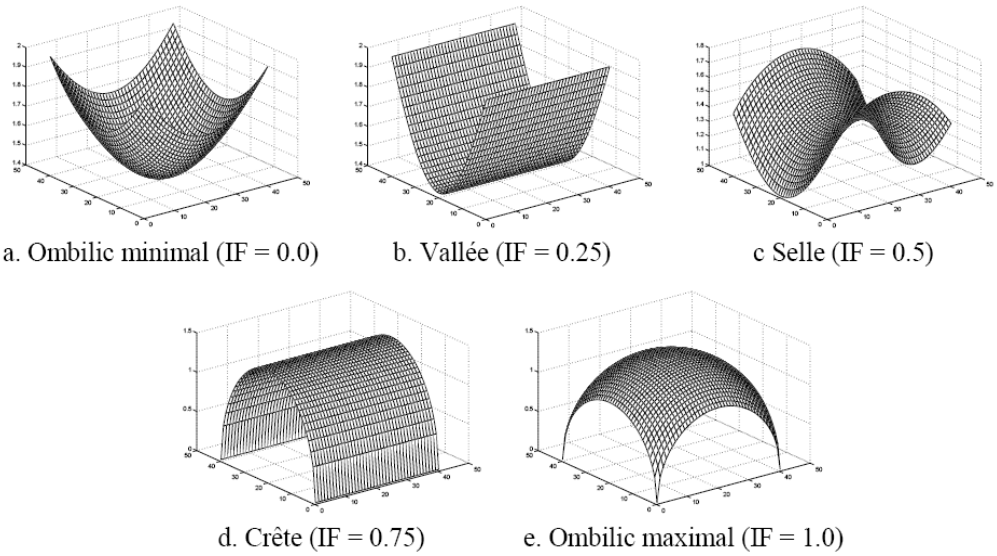


Figure 3.1 Valeurs représentatives de l'index de forme de [Koenderink, 1990].

Ce descripteur est stable vis-à-vis des transformations géométriques car le vecteur de courbures principales l'est également. Cependant le 3DSSD n'est pas stable vis-à-vis des multiples représentations topologiques d'un même objet. Un deuxième descripteur *3D Hough Transform Descriptor (3DHTD)* stable typologiquement a été dérivé de la *transformée 3D de Hough* [Hough, 1962]. La *transformée de Hough* s'appuie sur le principe d'accumulation des points dans une partition de l'espace. Cette méthode n'est pas stable aux transformations géométriques car la transformée dépend du repère choisi. Les auteurs l'ont rendu stable en l'associant à une procédure d'alignement spatial ce qui donne naissance au descripteur *Canonical 3D Hough Transform Descriptor (C3DHTD)* [Zaharia *et al.*, 2002]. L'alignement est réalisé en évaluant les directions principales de la surface.

[Johnson *et al.*, 1999] encodent les positions relatives des points définissant les surfaces d'un modèle 3D par rapport à l'un d'entre eux (voir Figure 3.2) en obtenant ce qu'ils appellent une *image de spin*. Pour chaque point on considère le plan tangent perpendiculaire à la normale du point par rapport à la surface du modèle. Sur ce plan, on effectue une projection de tous les autres points du modèle. Le nuage obtenu suite à la projection est discrétisé produisant une matrice 2D de fréquences. L'espace correspondant au nuage de points est découpé en cases de tailles identiques. Les nœuds obtenus en quadrillant la surface constituent l'*image de spin*. La couleur associée à chaque nœud dépend du nombre de points projetés dans les cases voisines. Lorsque la projection d'un point tombe à l'intérieur d'une case, l'ensemble des nœuds du quadrillage entourant la case voit leur niveau de fréquence augmenter d'un certain pourcentage (suivant la proximité de la projection). Cette technique offre de bons résultats car elle sauvegarde l'information de proximité. Si plusieurs points de vue sont considérés, la perte d'information observée lors de la projection 2D impacte moins sur la pertinence des résultats. Le modèle utilise une méthode de filtrage des points projetés sur le plan orienté afin de limiter les effets de bords dûs aux bruitages visuels comme par exemple, les occlusions. La similarité est mesurée en calculant le niveau de corrélation entre les *images de spin* indexées et l'*image de spin* obtenue pour le modèle en cours. Au minimum deux *images de spins* doivent être isolées afin de pouvoir calculer la transformation rigide appliquée (rotation et translation) à l'objet recherché pour s'apparier à l'objet inclus dans la scène. Les auteurs ont appliqué leur approche avec des résultats satisfaisants dans le domaine médical, en CAO, dans le domaine géographique, etc.

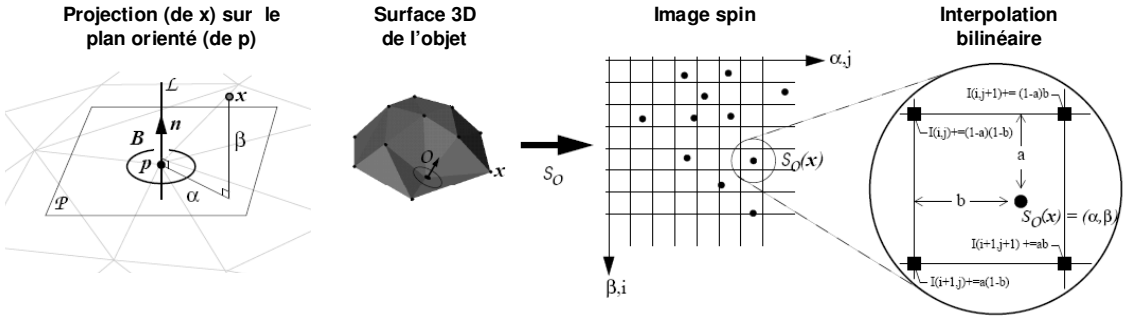


Figure 3.2 Processus d'inclusion d'un nouveau point dans la matrice de spin inspiré de [Johnson et al.*, 1999].*

Les *relations spatiales pondérées* [Del Bimbo *et al.*, 1998] assurent une caractérisation locale des objets 3D. Une relation spatiale pondérée entre deux régions correspond à une matrice 3x3 dont les éléments quantifient, sur une échelle de 0 à 1, les possibilités de déplacement de n'importe quel point de la région A à la région B. Les lignes correspondent aux trois types de déplacements relatifs sur l'axe des ordonnées (-1, 0, 1) et les colonnes aux déplacements relatifs sur l'axe des abscisses. Dans la Figure 3.3, nous illustrons trois relations spatiales pondérées. Les régions sont obtenues par segmentation de la surface 3D selon les valeurs de courbures uniformes. Par exemple, la première matrice indique que, si l'on part de A, le déplacement relatif (+1,+1) nous amène dans 100% des cas dans B. Elles sont organisées au sein d'un graphe dont les nœuds correspondent aux régions de la surface 3D. Les nœuds sont alors étiquetés avec l'information de courbure moyenne. Les nœuds sont reliés entre eux par des arêtes étiquetées indiquant la relation spatiale pondérée entretenue par deux régions. La similarité entre deux modèles est alors établie en estimant la possibilité de construire un isomorphisme partiel entre les deux modèles.

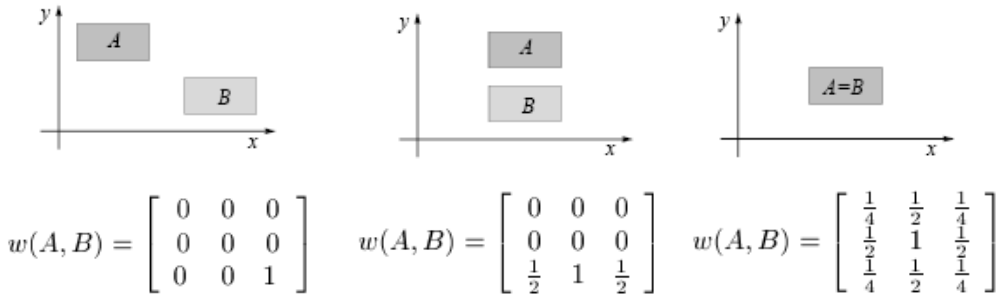


Figure 3.3 Relations spatiales pondérées caractérisées par les matrices de déplacements relatifs [Del Bimbo et al.*, 1998].*

Dans [Kazhdan *et al.*, 2003] les auteurs proposent un descripteur basé sur la caractérisation de la projection d'une forme sur un ensemble de sphères concentriques (*spherical harmonics*). Le calcul de ce descripteur repose sur la décomposition d'un modèle 3D dans une collection de fonctions 3D définies sur des sphères concentriques. Afin de rendre le processus de décomposition en sphères concentriques, une étape préliminaire de voxélisation du modèle 3D polygonal doit être réalisée. La voxélisation correspond à un processus d'indexation spatiale à base de volumes unitaires (cf. section 1.2.2). Chaque sphère est alors analysée afin d'obtenir les fonctions harmoniques qui la composent. Les fonctions harmoniques permettent d'identifier la signature de chaque sphère. Le descripteur de forme correspond à la compilation des amplitudes des fonctions harmoniques ainsi obtenues. Le processus de calcul du descripteur sphérique est

illustré dans la Figure 3.4. Le descripteur ainsi obtenu est utilisé pour indexer et par la suite mesurer la similarité entre deux objets 3D.

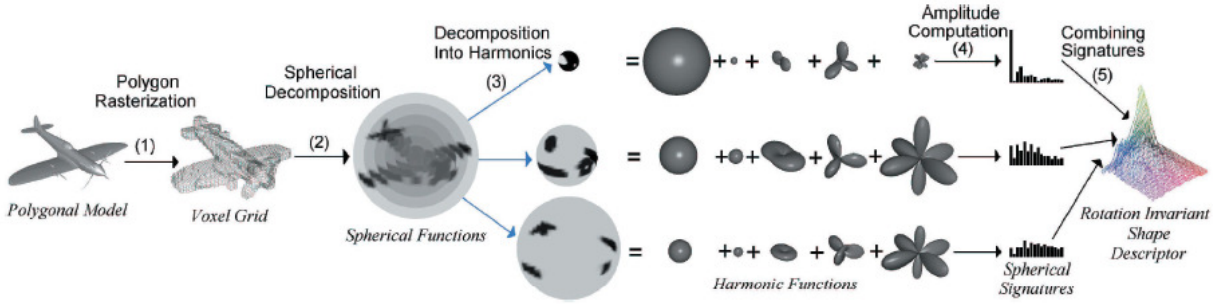


Figure 3.4 Calcul du descripteur sphérique de [Kazhdan et al.*, 2003].*

3.1.1.2. Indexation des objets 3D à base de vues 2D

Les approches que nous présentons dans cette section ont l'avantage d'essayer de réduire la complexité d'une recherche de motifs 3D à une recherche de motifs sur des images 2D. Lors d'une telle transformation on risque de perdre de l'information cependant les résultats montrent une efficacité et de faibles temps de calcul, ce qui justifie l'attrait des scientifiques pour de telles approches. De plus, généralement les auteurs proposent l'utilisation de plusieurs images 2D pour la caractérisation d'un même modèle ou scène 3D. Ainsi, on vise à combler, dans une certaine mesure, les lacunes issues du processus de passage de la 3D à la 2D. Cependant, il n'existe pas de méthodes robustes pour prédire le nombre et les directions des prises de vues. Ceci est un aspect important car le nombre de projections considérées détermine la complexité en termes de stockage et de temps moyen de réponse.

Dans [Mahmoudi *et al.*, 2002], les auteurs proposent une solution basée sur la construction de *vues caractéristiques* (*Characteristic Views*) pour la recherche de modèles 3D en s'appuyant sur une technique proche du monde 2D. Les modèles 3D concernés par cette proposition correspondent aux nuages de points 3D modélisés par la primitive *PointSet* de VRML [Web3D, 1997]. À partir d'un modèle 3D, les auteurs extraient un certain nombre de vues 2D obtenues en regardant l'objet 3D sous différents angles (voir Figure 3.5).

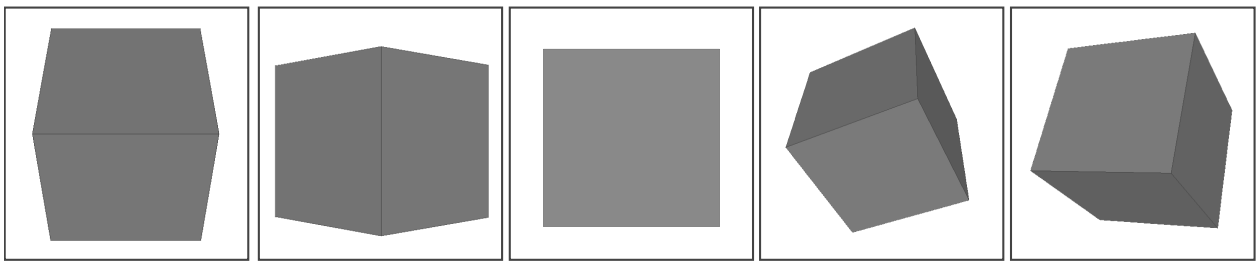


Figure 3.5 Cinq vues caractéristiques d'un cube.

Une méthode pour calculer le nombre de vues à prendre en compte est exposée dans [Mokhtarian *et al.*, 2000]. Ici, les auteurs considèrent trois vues principales et quatre vues secondaires. Les directions principales sont calculées à partir des trois valeurs principales de la matrice de covariance du modèle 3D. Les vues secondaires sont calculées à partir des orientations déduites des trois orientations principales. Pour chaque vue, les auteurs procèdent à l'extraction des contours de l'objet. Des mesures concernant la courbure de chaque contour sont effectuées. Par rapport à ces mesures, la description du contour est projetée dans un espace nommé *Curvature Scale Space* (CSS) [Mokhtarian *et al.*, 2003]. La similarité de deux objets 3D

est mesurée par rapport aux projections de contours respectifs dans l'espace CSS pour les sept vues considérées. Afin d'augmenter la vitesse de calcul des mesures de similarités, les projections CSS sont indexées à l'aide d'une structure *M-tree*. Dans une structure *M-tree* les éléments sont regroupés suivant leur distance à un élément repère du groupe. Cette structure permet d'optimiser les recherches car lorsque le centre d'un groupe est trop éloigné de l'élément recherché le groupe entier est élagué par le processus de recherche.

[Weiss *et al.*, 2001] proposent une méthode d'identification d'un objet 3D sur une image 2D. La méthode s'appuie sur l'extraction d'un ensemble d'éléments de contours (points et lignes) qui sont comparés aux descripteurs associés aux modèles 3D existants.

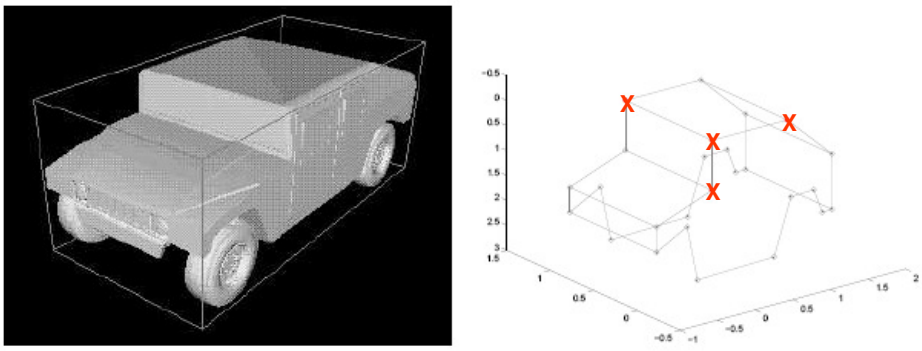


Figure 3.6 Projection d'un modèle 3D (à gauche) dans un espace d'invariants canoniques (à droite) [Weiss et al.*, 2001] – les points de base de l'espace sont marqués par un X.*

Afin de limiter le nombre de comparaisons et d'augmenter l'efficacité de la recherche tout en gardant des niveaux élevés de pertinence, les auteurs s'appuient sur le calcul d'invariants pour l'image et pour l'ensemble des modèles 3D. Les invariants 3D sont calculés à partir des propriétés géométriques des objets. Les auteurs s'intéressent à des propriétés du modèle qui se conservent lors de la projection 2D. Pour chaque couple de quatre points (X1, X2, X3, X4) du modèle, les auteurs construisent dans un repère canonique (voir Figure 3.6) les projections de tous les autres points du modèle.

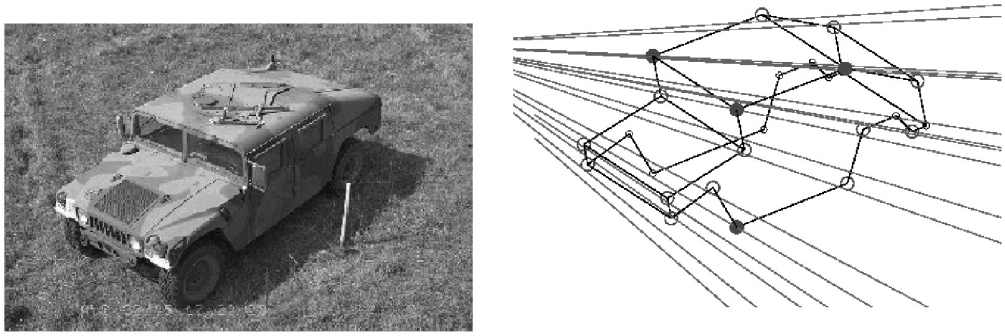


Figure 3.7 Intersection entre les lignes construites dans l'espace d'invariants 3D [Weiss et al., *2001].*

La reconnaissance d'un modèle dans une image se fait en quatre étapes. La première est consacrée à l'extraction des couples de cinq caractéristiques de l'image. En réalisant une analyse de contours sur l'image 2D, les auteurs retiennent uniquement les lignes parallèles aux directions principales. Ces lignes servent à sélectionner les ensembles de cinq points. Quatre points parmi les cinq sont sélectionnés de telle façon à ce qu'ils forment un angle 3D (c'est-à-dire qu'un point se trouve à l'intersection de trois segments et les trois autres points correspondent aux extrémités de ces segments). Le cinquième point est quelconque. La deuxième étape sert à calculer les deux

équations pour les trois invariants 3D et à construire une ligne dans l'espace 3D des invariants pour chaque couple de cinq points. Ensuite, il faut déterminer toutes les lignes ainsi construites qui approchent dans l'espace 3D des invariants, les points qui caractérisent un modèle 3D. Pour identifier un objet, on vérifie que toutes les lignes sont suffisamment près des points invariants du modèle (voir Figure 3.7). Une dernière étape est nécessaire à la validation du choix du modèle. On évalue la transformation correspondant à la projection du modèle 3D dans l'image 2D. Si l'erreur est trop importante l'appariement est invalidé.

L'originalité de cette méthode réside dans le fait qu'une seule prise de vue d'un modèle 3D peut aboutir à des résultats satisfaisants. Le bruit influence beaucoup l'efficacité de la méthode car il augmente le nombre de segments potentiellement intéressant et implicitement le nombre de lignes à construire dans l'espace des invariants 3D. Le nombre de caractéristiques considérées permet de compenser ces baisses d'efficacité. Cependant, l'augmentation du nombre de caractéristiques a un coût important sur les performances du système car cela augmente considérablement le nombre d'intersections à calculer.

L'indexeur 2D présenté dans [Funkhouser *et al.*, 2003] sert à caractériser les contours des objets 3D en considérant treize directions orthogonales pour les projections 2D. Cette approche d'indexation 2D des objets a été mise en place afin de supporter une recherche selon des dessins 2D fournis par les utilisateurs. Les auteurs appliquent une approche de caractérisation similaire aux sphères harmoniques (voir la Figure 3.8). Chaque forme 2D est indexée comme suit : 1) on construit la matrice des distances concernant le contour de la forme ; 2) on décompose la matrice en fonctions construites sur les cercles concentriques ; 3) et 4) la signature de chaque fonction est caractérisée par une somme de fonctions trigonométriques de base ; 5) la signature de la figure est construite en cumulant les caractéristiques de chaque fonction.

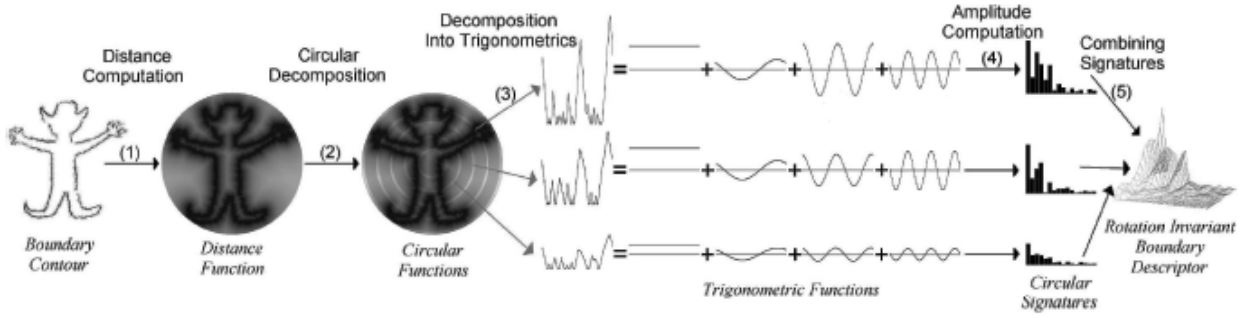


Figure 3.8 Calcul de descripteur 2D associé aux vues orthogonales d'après [Funkhouser et al.*, 2003].*

3.1.1.3. Critères de recherche liés à l'apparence

[Koubaroulis, 2001] propose un descripteur pour la caractérisation de l'apparence des images 2D. Même si les auteurs ne proposent pas explicitement une application à la recherche de contenus 3D, leur approche présente l'intérêt de considérer l'apparence des objets en tant que facteur discriminant. De plus, la robustesse de leur descripteur aux éventuelles variations de l'intensité lumineuse offre de bonnes prémisses pour une adaptation 3D. Le descripteur appelé MNS (*Multimodal Neighbourhoods Signature*) s'intéresse à la caractérisation locale (dans une région voisine) de la densité des couleurs à l'aide d'une fonction multimodale. Les régions traitées sont uniquement celles dont le nombre de modes est supérieur ou égal à deux. Pour faciliter les calculs, les auteurs considèrent des voisinages rectangulaires. Les modes de la densité des couleurs sont localisés à l'aide de l'algorithme *mean-shift* [Comaniciu *et al.*, 2002]. Les modes de moins de 5 pixels sont rejetés. Deux par deux, les modes restants sont organisés en *clusters* dans l'espace de couleurs RGB. Seul un vecteur est retenu par *cluster* afin de rendre le

processus de recherche plus efficace. Ainsi, on obtient un descripteur qui correspond à un ensemble de vecteurs RGB. Lorsque plusieurs images sont disponibles pour le même objet les descripteurs obtenus indépendamment peuvent être superposés, en gardant uniquement les vecteurs distincts. Le processus de construction du descripteur MNS est présenté dans la Figure 3.9. Du fait que le nombre de vecteurs entrant dans la composition d'un descripteur varie en fonction de la complexité des images utilisées lors de la mesure de similarité, les auteurs ont prévu le calcul d'une distance entre descripteurs à nombre variable de vecteurs. Leur méthode repose sur le calcul d'un sous-ensemble biparti de vecteurs mis en correspondance par rapport à un seuil donné. La dissimilarité entre deux descripteurs est alors construite comme la somme des différences entre les vecteurs mis en correspondance. MNS n'est pas tout aussi discriminatif que les méthodes qui s'intéressent à la forme et la topologie des régions, mais il se relève robuste par rapport aux changements de l'angle de prise de vue.

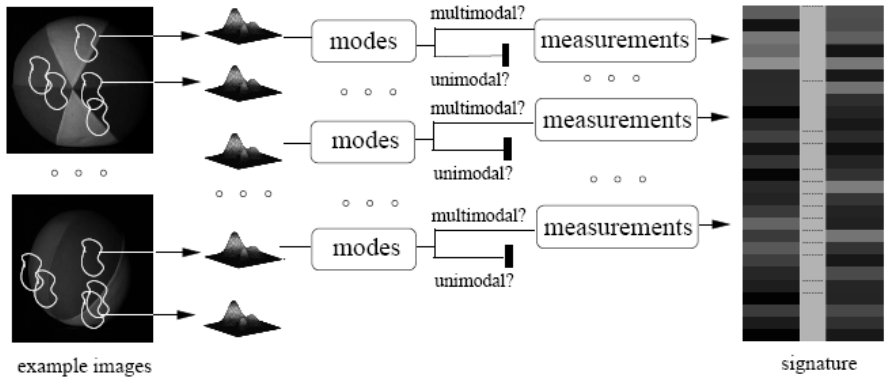


Figure 3.9 Calcul du descripteur MNS pour un objet à partir de plusieurs vues [Koubaroulis, 2001].

3.1.1.4. Critères de recherche topologiques

Dans [Hilaga *et al.*, 2001], les auteurs mesurent la similarité entre des modèles polyédriques en utilisant les caractéristiques topologiques des modèles. La méthode s'appelle *matching topologique*. Elle repose sur la comparaison des graphes multi-échelles de Reeb (MRG, *Multiresolutional Reeb Graphs*). Un graphe de Reeb (RG) [Reeb, 1946] contient des informations relatives à la structure et la topologie d'une forme 3D à différents niveaux de détail. Un nœud dans le RG représente une composante connexe dans une région particulière et les nœuds adjacents sont reliés par une arête si les composantes se touchent. La connexité est définie par rapport à une fonction continue choisie à l'avance. Un RG invariant aux transformations rigides, robuste aux changements de connectivité dûs aux simplifications, aux subdivisions ou bien aux bruits, ou à certaines déformations, peut être construit en considérant une fonction μ caractérisant la distribution de la distance géodésique [Lanthier *et al.*, 1997]. [Hilaga *et al.*, 2001] proposent une extension multi-échelle afin de pouvoir caractériser les modèles 3D à plusieurs niveaux de détail ce qui permet notamment de mesurer la similarité entre les sous-régions de modèles 3D. L'extension suppose le partitionnement de régions au sein d'une composante connexe. Des niveaux plus fins de partitionnement approximent plus précisément l'objet considéré. Des relations père-fils sont induites par ces partitionnements répétitifs entre les nœuds de deux niveaux successifs.

Afin de mesurer la similarité entre deux MRGs, on considère au début les graphes dans leurs représentations les moins détaillées. On ajoute dans une liste les nœuds de chaque graphe au niveau de détail actuel et on cherche des paires de nœuds issus de graphes distincts dont les caractéristiques et celles de leurs voisins sont similaires. La mesure de la similarité entre les caractéristiques de deux nœuds est une somme pondérée entre l'aire et la longueur relative des

triangles représentés par les deux nœuds. Les nœuds sont ensuite retirés de la liste et leurs fils respectifs y sont ajoutés. On répète le processus de recherche de paires de nœuds tant qu'il y a encore des nœuds dans la liste à explorer. Afin de respecter la consistance topologique, lors de la recherche de nouvelles paires, les nœuds sont sélectionnés uniquement s'ils appartiennent à la même région et si leurs parents ont également été appariés. Ceci évite de comparer des nœuds qui se trouvent sur des branches différentes des graphes. La mesure de similarité entre les MRG est effectuée en considérant toutes les paires qui ont été identifiées aux pas précédents.

3.1.2. Critères de recherche au niveau sémantique

La recherche des données 3D en utilisant des critères de niveau sémantique peut s'apparier à la recherche à base de mots-clés que nous rencontrons classiquement dans la recherche textuelle. Ceci est dû au fait que la sémantique est associée de manière explicite aux données 3D. Elle est représentée sous forme textuelle soit en utilisant un ensemble de termes décrivant l'entité, soit en utilisant une description plus complexe qui inclut l'utilisation des relations entre termes à la manière de RDF, OWL, MPEG-7 ou autre formalisme de représentation.

La principale barrière à la large adoption dans les systèmes de recherche de données 3D est la difficulté d'automatiser le processus d'extraction de la sémantique à partir des caractéristiques de bas niveau comme la géométrie, l'apparence ou la topologie de données 3D. Cependant, en dépit de la difficulté d'obtenir des informations sémantiques de manière automatique, la plupart des entrepôts de données 3D disponibles sur le Web, tels que 3DWarehouse[19], 3DCafe[20], mettent à disposition des utilisateurs des moyens de recherche par mots-clés. Ces systèmes encouragent les utilisateurs à décrire les données 3D qu'ils déposent dans les entrepôts. La description se fait suivant un certain nombre de critères sémantiques imposés par le gestionnaire de l'entrepôt. À défaut de fournir ces informations, les données 3D n'ont aucune visibilité et ne sortent jamais dans les résultats de recherche. Cette politique de publication de données 3D au moyen d'entrepôts accessibles depuis le Web, laisse entrevoir une croissance certaine de données 3D qui peuvent être exploitées en considérant les informations sémantiques qui leurs sont associées.

Les propositions de caractérisation sémantique que nous avons présentées à la fin du chapitre précédent stockent explicitement la sémantique dans des documents (RDF, MPEG-7, X3D). Pour interroger la sémantique, des solutions spécifiques à chaque type de document peuvent être implémentées afin de récupérer les objets 3D dont la sémantique satisfait la requête de l'utilisateur. Dans la suite de cette section, nous présentons les principaux langages d'interrogation relatifs à la recherche d'information dans les standards RDF et MPEG-7 sur lesquels la plupart des applications sémantiques reposent.

3.1.2.1. Langages d'interrogation pour RDF

La structuration des documents RDF enrichit les possibilités d'analyse et implicitement d'utilisation des descriptions faites en RDF. Un grand nombre de langages de requêtes qui ont été répertoriés dans [Haase *et al.*, 2004][Hutt, 2005], visent à exploiter les informations représentées en intégrant les concepts du modèle RDF.

XQuery [Boag *et al.*, 2007] étant par excellence le standard d'interrogation des documents XML, qui correspond à la sérialisation des graphes RDF, semble être une bonne

[19] http://sketchup.google.com/3dwarehouse/
[20] http://www.altairmodels.com/Free-3D.php

solution pour interroger les documents RDF. Cependant, il n'intègre pas des opérateurs de sélection qui font référence aux concepts utilisés dans RDF et notamment les triplets (sujet, prédicat, valeur). Certains langages, comme XsRQL [Katz, 2004], s'appuient sur l'encodage XML de graphes de descriptions RDF et proposent des langages d'interrogation sous la forme d'une extension de XQuery. D'autres (SPARQL [Prud'hommeaux *et al.*, 2007], RDQL [Seaborne, 2004], et RQL [Karvounarakis *et al.*, 2002]) sont construits autour des concepts et des structures utilisés en RDF. Nous poursuivons en présentant les langages issus de cette deuxième catégorie.

SPARQL [Prud'hommeaux *et al.*, 2007]

SPARQL [Prud'hommeaux *et al.*, 2007] est un langage d'interrogation RDF en cours de standardisation par le consortium W3C. Il peut être utilisé afin de formuler des requêtes portant sur des données provenant de plusieurs sources de données (fichiers RDF distincts, ou bien fichiers contenant des (méta)données exprimées en RDF). SPARQL supporte l'utilisation des patrons de graphes simples (tout triplet *terme RDF, prédicat RDF, terme RDF*) ou complexes qui incluent l'utilisation de filtres, de conjonctions, de disjonctions et de contraintes optionnelles dans la formulation de requêtes.

Chaque triplet de la requête peut contenir des variables et des valeurs RDF (termes ou URI). Un ensemble de contraintes peut être associé à la requête afin de contrôler les valeurs des variables. Le triplet suivant `( ?x, <http://lig.imag.fr/employe#type>, <http://lig.imag.fr/doctorant>)` représente une requête qui lie la variable `x` à tous les employés `doctorant` du LIG (Laboratoire d'Informatique de Grenoble). Le langage peut être également utilisé afin d'inférer de nouvelles connaissances en utilisant des requêtes telles que `CONSTRUCT`, `DESCRIBE`, `ASK` qui permettent respectivement de construire de nouveaux graphes RDF, d'ajouter des nouveaux descripteurs aux données, d'évaluer la validité de certaines expressions. Les requêtes sont formulées en utilisant une écriture de type SQL. Le résultat d'une requête SPARQL sérialisable en XML peut correspondre à un ensemble de valeurs ou bien de graphes RDF.

SPARQL est pour l'instant dans l'état de proposition de standardisation par le consortium W3C. En termes d'implémentation on peut compter sur le serveur Web Joseki[21] pour la plate-forme Jena[22] dont les bases scientifiques ont été posées dans [Carroll *et al.*, 2004]. D'autres variantes moins complexes (*RDF Data Query Language* – RDQL) [Seaborne, 2004], Squish [Miller, 2001], Versa [Olson *et al.*, 2004]) mais suffisamment puissantes ont été implémentées dans des systèmes de gestion de données RDF. Ces librairies sont distribuées en tant que projets libre source (Jena, RDFStore[23], Sesame[24], PHP XML Classes[25], 3Store[26], RDF Api for PHP[27]). Compte tenu de la grande ressemblance au sein de cette famille de langages, nous présentons ci-dessous uniquement RDQL [Seaborne, 2004] qui semble jouir d'un grand support étant notamment inclus dans les plates-formes Jena, PHP XML Classes.

RDQL [Seaborne, 2004]

RDQL [Seaborne, 2004] est un langage qui utilise une description de requêtes similaire au langage SQL. RDQL est utilisé exclusivement pour l'extraction de l'information contenue dans les graphes RDF. Une requête RDQL correspond à un patron de graphe exprimé comme

[21] http://www.joseki.org
[22] http://jena.sourceforge.net/
[23] http://rdfstore.sourceforge.net/
[24] http://sesame.sourceforge.net/
[25] http://rdfxmlclasses.sourceforge.net/
[26] http://3store.sourceforge.net/
[27] http://rdfapi-php.sourceforge.net/

une liste de triplets. Dans la Figure 3.10, nous donnons l'équivalent de la requête formulée plus haut en SPARQL, par l'intermédiaire d'un triplet, pour retrouver les doctorants du LIG.

```
SELECT ?x
WHERE (?x,
     <http://lig.imag.fr/employe#type>,
     <http://lig.imag.fr/doctorant>)
```

Figure 3.10 Requête RDQL pour retrouver les doctorants du LIG.

La puissance d'expression est moindre qu'en SPARQL mais les nombreuses implémentations démontrent qu'il est suffisamment puissant. Cette implémentation est disponible également sous forme d'une API Java ce qui rend possible son utilisation directement au sein de logiciels ou d'environnements supportant l'exécution dynamique de scripts.

RQL [Karvounarakis *et al.*, 2002]

RQL [Karvounarakis *et al.*, 2002] est un autre langage d'interrogation RDF qui suit une approche fonctionnelle. RQL supporte l'utilisation de chemins génériques en associant des variables aux termes (sujets, propriétés) ainsi qu'aux prédicats. RQL repose sur un modèle formel de graphe qui inclut les primitives de description RDF et permet l'interprétation de ressources externes par utilisation de schémas supplémentaires. Ce choix différencie le plus ce langage des autres langages d'interrogation RDF qui considèrent une approche basée sur des triplets. L'originalité de RQL réside dans sa capacité à combiner l'interrogation des schémas et des données, l'exploitation de taxonomies de termes et de multiples classifications. Dans la Figure 3.11, nous donnons l'équivalent de la requête formulée précédemment pour retrouver les doctorants du LIG.

```
SELECT x
FROM {x}type{y}
WHERE y=&http://lig.imag.fr/doctorant
```

Figure 3.11 Requête RQL pour retrouver les doctorants du LIG.

XsRQL [Katz, 2004]

XsRQL (*XQuery-style RDF Query Language*) [Katz, 2004] est un langage d'interrogation RDF qui s'est fortement inspiré de la syntaxe et du style de requêtes XQuery. XsRQL réutilise tant un grand nombre de concepts (le modèle de données, l'approche fonctionnelle) que la syntaxe proposés dans XQuery. Globalement, les prédicats XPath[Clark *et al.*, 1999] sont remplacés par des critères de sélection portant sur les relations décrites dans RDF. Cette approche permet aux utilisateurs familiers avec les langages XML et XQuery de facilement formuler des requêtes sur les descriptions RDF. Dans la Figure 3.12 nous présentons la syntaxe XsRQL équivalente aux requêtes précédemment présentées dans cette section.

```
declare prefix lig: = <http://lig.imag.fr/>;
declare datasource membresLIG  = <http://lig.imag.fr/users.rdf>;
for $person in membresLIG//*[@lig:employe#type/lig:doctorant ]
            return     $person
```

Figure 3.12 Requête XsRQL pour retrouver les doctorants du LIG.

3.1.2.2. Interrogation de données MPEG-7

Des langages d'interrogation spécifiques pour MPEG-7 existent [Graves *et al.,* 2002][Liu *et al.,* 2002][Fatemi *et al.,* 2003] mais ils ciblent plutôt la dimension audio-visuelle du contenu multimédia et, à notre connaissance, ils ne sont généralement pas extensibles. Dans [Hammiche *et al.,* 2007], toutefois, les auteurs proposent une solution d'interrogation extensible qui s'appuie sur l'utilisation conjointe d'ontologies de domaine et du standard MPEG-7. Nous présentons d'abord la proposition de [Fatemi *et al.,* 2003], car même si elle reste fortement ancrée dans le domaine de l'audio-visuel, les principes et les choix qui ont mené à sa mise en place semblent pertinents et transposables aux données 3D. Ensuite, nous analysons la proposition de [Hammiche *et al.,* 2007].

MPEG-7 est encodé en XML, ce qui fait du langage XQuery un candidat sérieux pour l'interrogation de documents MPEG-7. Toutefois, si un utilisateur veut exploiter directement ce langage, il doit bien connaître la façon dont les informations attributaires, les descripteurs audio-visuels et sémantiques sont organisés au sein du schéma MPEG-7. Afin de faciliter la récupération des informations enfouies dans les documents MPEG-7, il est nécessaire que les requêtes formulées s'appuient sur un modèle abstrait de données qui respecte les besoins des utilisateurs potentiels.

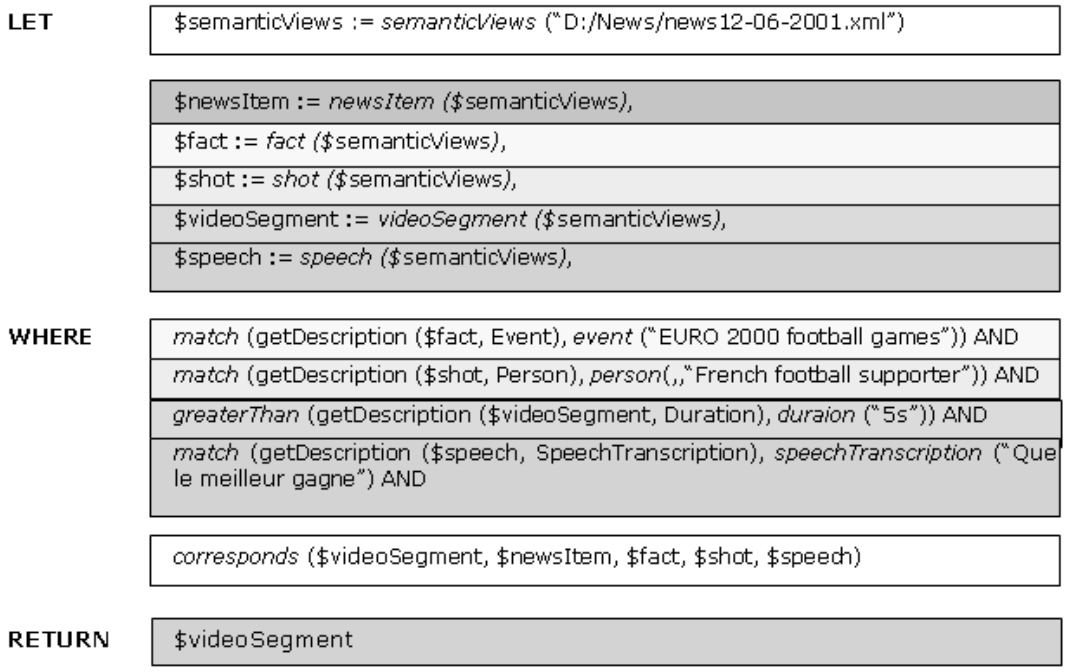


Figure 3.13 Un exemple de requête SVQL d'après [Fatemi *et al.*, 2003].

Dans le cadre d'une étude des besoins professionnels identifiés auprès du personnel de la société Radio Suisse Romande (RSR), [Fatemi *et al.*, 2003] proposent *Semantic View Query Language (SVQL)*, une extension du langage XQuery qui contient des fonctions et prédicats spécifiques qui permettent aux utilisateurs d'extraire l'information en s'affranchissant de la structure complexe des documents MPEG-7. Les prédicats et les fonctions assurent l'exploitation de l'information contenue dans les documents MPEG-7 considérés sous différentes vues (thématique, visuelle, auditive, …). Chaque vue est décrite en utilisant cinq éléments : les entités de base, les descriptions, les intra-relations, les inter-relations et les opérateurs. Ces éléments sont à la base de toute requête SVQL. Comme le langage SVQL suit la syntaxe de XQuery (voir la Figure 3.13), les opérateurs propres à SVQL sont considérés comme des fonctions XQuery qui

traduisent les concepts abstraits des vues en expressions XPATH pointant sur les informations associées dans la structure du document MPEG-7.

Dans la Figure 3.13 nous présentons un exemple de requête SVQL qui recherche des vidéo concernant une intervention d'un supporter français de football en rapport avec le championnat européen EURO 2000. Les auteurs introduisent une série de fonctions spécifiques à l'organisation de leurs descripteurs (*fact()*, *shot()*, *videoSegment(), speech(), event(), person(), getDescription()*) qui cachent la complexité de l'organisation des descripteurs au sein de documents MPEG-7. Une fonction qui contrôle les critères de jointures induites par l'évaluation des fonctions présentées ci-dessus est également implémentée : *corresponds*.

Dans [Hammiche *et al.,* 2007], les auteurs s'intéressent à l'interrogation des informations sémantiques stockées dans les documents MPEG-7 en utilisant des opérateurs sémantiques qui portent sur les données MPEG-7. Les opérateurs sémantiques sont définis au moyen de règles sémantiques qui exploitent les opérateurs de base liés aux concepts présents dans les documents MPEG-7. Les requêtes comportant des opérateurs autres que ceux de base sont réécrites en utilisant les règles de définition de ces opérateurs. En répétant plusieurs fois le processus de réécriture, la requête est décrite uniquement en utilisant des opérateurs de base, ce qui permet de générer une requête XQuery qui extrait directement les informations sémantiques recherchées. L'intérêt de cette proposition est qu'elle est extensible puisque de nouveaux opérateurs peuvent être construits en s'appuyant sur ceux existants.

Cette section relative à la recherche de contenu 3D nous a permis d'analyser les différents moyens de caractérisation (géométrique, d'apparence, topologique, sémantique) des données 3D en vue de leur réutilisation. Dans la section suivante, nous insistons sur l'exploitation de certains types de critères au sein des applications de recherche de contenus 3D.

3.1.3. Applications pour la recherche de contenus 3D

Dans cette section nous présentons trois applications (*Search Engine 3D* [Funkhouser *et al.*, 2003], *GIDeS* [Fonseca *et al.*, 2004] et *Digital Shape Workbench* (*DSW*) [Albertoni *et al.*, 2005]) issues du monde académique qui visent l'exploitation d'un ou plusieurs types de critères pour la recherche de contenus 3D. *Search Engine 3D* combine des critères sémantiques (mots-clés) avec des critères géométriques. GIDeS s'appuient sur les critères géométriques et topologiques. *DSW* privilégie l'utilisation de critères sémantiques.

3.1.3.1. 3D Search Engine [Funkhouser et al.*, 2003]*

À l'Université de Princeton, les travaux à l'initiative de [Funkhouser *et al.*, 2003] et poursuivis par [Min, 2004] s'intéressent à la construction d'un moteur de recherche d'objets 3D dont les requêtes sont basées sur une réplique qui reprend les principaux traits des modèles recherchés. Les requêtes peuvent être enrichies en utilisant des mots-clés afin de lever l'ambiguïté possible entre les objets distincts de formes similaires. Les auteurs optent pour une mixité entre les recherches à base de contenus et les approches classiques à base de mots-clés. Les approches classiques de recherche par mots-clés se montrent insuffisantes pour le contenu 3D. Les auteurs reprochent aux méthodes à base de mots-clés le manque de robustesse qui apparaît lorsque les objets (et les fichiers qui les contiennent) ne comportent pas d'information textuelle auto-descriptive, ou lorsque bien que présente, elle s'avère insuffisamment descriptive.

En même temps, les auteurs s'accordent sur le fait qu'une recherche uniquement basée sur le contenu (au moyen de descripteurs de forme) montre certains manques : des objets ayant quasiment la même forme peuvent représenter des choses différentes. Pour pallier les lacunes des

approches présentées, un système capable de combiner les deux approches (textuelle et à base d'exemples) a été mis au point. L'organisation du système est présentée dans la Figure 3.14. La partie *off-line* du système assure l'indexation textuelle et l'indexation de la forme (2D et/ou 3D) des objets 3D.

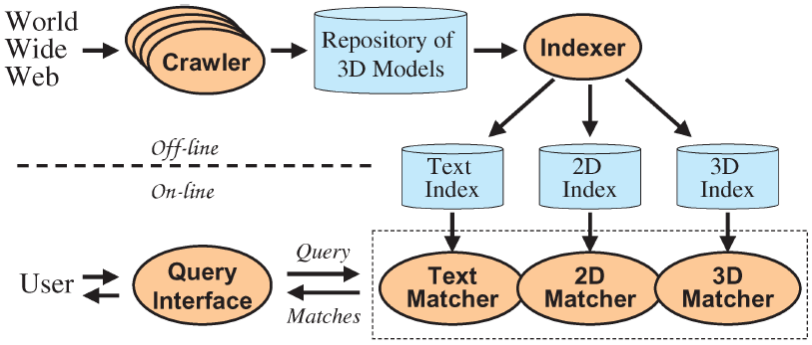


Figure 3.14 Organisation du système de recherche proposé par [Funkhouser et al.*, 2003].*

L'indexation textuelle correspond à l'extraction des noms présents dans le code modélisant les objets 3D. Par exemple, lorsque l'on manipule des documents VRML, l'*Indexer* extrait le contenu des attributs `DEF` ou des éléments `WorldInfo`. La sélection des termes à indexer est réalisée en utilisant l'approche TF-IDF/Rocchio [Rocchio, 1971]. La partie géométrique des objets est transformée en polygones afin de pouvoir être plus facilement voxélisable. Ensuite, chaque objet ainsi voxélisé (en utilisant une grille de 64x64x64 voxels) est analysé afin de calculer le descripteur de surface sphérique (en utilisant 32 sphères) tel qu'indiqué dans la section 3.1.1.1.

L'indexeur 2D sert à caractériser les contours des objets 3D en considérant treize directions orthogonales pour les projections 2D. Cette approche d'indexation 2D des objets a été mise en place afin de supporter une recherche selon des dessins 2D fournis par les utilisateurs.

Afin de valider leur prototype, les auteurs ont réalisé trois types de tests : i) tests pour évaluer la pertinence de l'utilisation de la technique des sphères harmoniques ; ii) tests pour quantifier la plus-value obtenue suite à l'enrichissement d'une méthode de recherche ; et iii) tests pour caractériser l'utilisation de l'outil faite par des utilisateurs sur le Web. Pour la collection de modèles considérés (1890 modèles de meubles et diverses fournitures mises à disposition par Viewpoint[28]) la courbe rappel/précision de leur méthode est meilleure que d'autres techniques implémentées pour l'occasion qui s'appuient sur les moments [Divarkan, 2004], les images de Gauss [Horn, 1984] ou encore la distribution de la forme [Osada *et al.*, 2002]. La deuxième série d'expériences a démontré l'intérêt et la complémentarité des recherches multimodales. Dans la troisième série, les utilisateurs ont la possibilité de reformuler leurs requêtes plusieurs fois afin de lever certaines ambiguïtés lorsqu'ils utilisent uniquement une des trois méthodes de recherche. Les mesures réalisées montrent que l'utilisation des itérations qui s'appuient sur la similarité de formes est plus efficace que les itérations s'appuyant uniquement sur la similarité textuelle.

Des critiques peuvent être apportées à ces travaux et notamment en ce qui concerne l'utilisation des index 2D. Lors de l'indexation, on ne considère que les contours extérieurs des objets, toute information liée à la texture ou aux contours intérieurs est ignorée. En ce qui concerne l'indexation 3D, les auteurs traitent les objets comme un tout, il n'est pas possible de rechercher et récupérer uniquement les parties d'un modèle. De plus, l'indexation ne traite que

[28] http://www.viewpoint.com/

de la forme de l'objet. Des attributs tels que la structure, la couleur, la texture ne sont pas pris en compte.

3.1.3.2. GIDeS [Fonseca et al., 2004]

[Fonseca *et al.*, 2004] proposent GIDeS, un système pour la création de modèles géométriques 3D en utilisant une technique de *sketch* combinée à des processus de recherche et de suggestion. Cette proposition a été développée dans l'esprit de favoriser la réutilisation des données dans le domaine de la CAO.

Le processus de recherche est mis à contribution dans le but d'inclure des objets déjà créés au sein de nouvelles scènes 3D. L'inclusion/récupération d'un objet se fait de manière interactive. À tout moment le système suggère à l'utilisateur un ensemble d'objets 3D de la collection, similaire à ce que l'on est en train de dessiner/construire. Par exemple, l'utilisateur commence par dessiner un contour 2D. Le système lui propose un ensemble de modèles 3D qui sont similaires au contour 2D sous la forme d'une liste de suggestions sélectionnables. L'utilisateur choisit l'objet le plus proche de ce qu'il cherche parmi les suggestions faites par le système. Cet objet peut être ensuite complété par de nouveaux éléments géométriques fournis par l'utilisateur. Par la suite, le système calcule un nouvel ensemble d'*objets suggestions*. Ainsi, après quelques itérations les utilisateurs arrivent à retrouver l'objet recherché. Lors des mesures, les auteurs ont observé au plus quatre itérations pour aboutir à l'identification de l'objet.

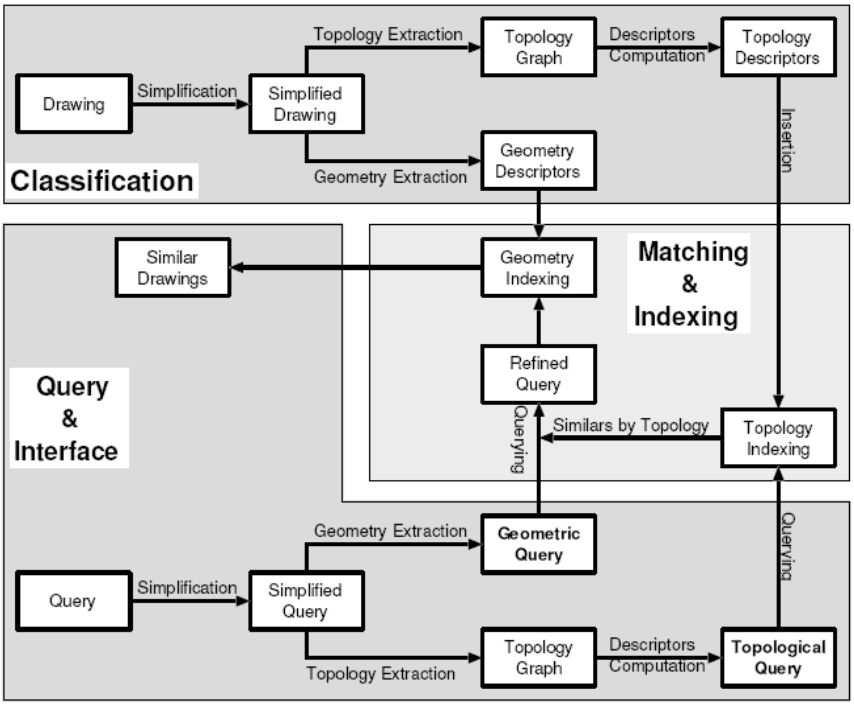


Figure 3.15 Architecture de base pour la recherche d'objets dans GIDeS d'après [Fonseca et al.*, 2004].*

Le système GIDeS s'appuie sur une architecture initialement imaginée pour la recherche à base de *sketch* d'images complexes. L'architecture est présentée dans la Figure 3.15. Elle est découpée en trois modules qui gèrent respectivement : la classification, la mise en correspondance et l'indexation, l'interface et la requête. La classification correspond au processus de caractérisation d'objets 3D préexistants dans le système. Chaque objet est considéré d'un point de vue géométrique et topologique. La signature topologique concerne les relations spatiales entre les facettes des objets 3D et la signature géométrique concerne les arêtes des

objets 3D. Les objets sont tous définis suivant le paradigme B-rep (voir section 1.2.2). Pour chaque objet 3D on considère deux graphes décrivant respectivement les liens entre les facettes et les arêtes des objets 3D. Chaque graphe est ensuite analysé et on calcule des descripteurs qui s'appuient sur les valeurs principales de la matrice d'adjacence des graphes [Cvetkovic *et al.,* 1997]. Les descripteurs sont ensuite insérés dans deux structures d'indexation de type *NB-Tree* [Fonseca *et al.,* 2003] adéquates pour l'indexation de données multidimensionnelles. Pour construire une structure *NB-Tree*, on projette les données dans un espace 1D en considérant la norme euclidienne de chaque donnée. Ensuite, les points sont triés en utilisant les méthodes *B+tree*. Lorsqu'une requête est émise par un utilisateur, les descripteurs associés aux graphes des arêtes et des facettes de la surface requête sont calculés. Les descripteurs obtenus sont comparés à l'ensemble des descripteurs indexés dans les structures *NB-Tree* existantes. La mesure de similarité reflète la distance entre les vecteurs multidimensionnels correspondant aux descripteurs de la requête et de l'objet candidat.

L'approche proposée par les auteurs a le mérite de s'intéresser à une nouvelle approche d'utilisation à base de requêtes de données 3D dans un environnement de conception. Cependant, les critères utilisés par les auteurs ne semblent pas les plus opportuns. Les descripteurs ne sont pas robustes par rapport aux multiples représentations (versions plus ou moins détaillées) d'un même objet.

3.1.3.3. Digital Shape Workbench [Albertoni et al.*, 2005]*

Le Réseau d'Excellence AIM@SHAPE[29] propose dans [Albertoni *et al.*, 2005] une solution pour la recherche de contenu en utilisant uniquement des critères de haut niveau. Présentés dans le chapitre précédent (voir section 2.4.5), ces critères couvrent une large palette d'information concernant : le processus d'acquisition d'une forme 3D, les caractéristiques de la forme en tant que fragment média et la description des futures utilisations de la forme. En plus de ces outils formels de description de formes, les auteurs proposent une architecture logicielle nommée DSW (*Digital Shape Workbench*) pour la recherche de l'information au sein d'un entrepôt de formes (voir la Figure 3.16).

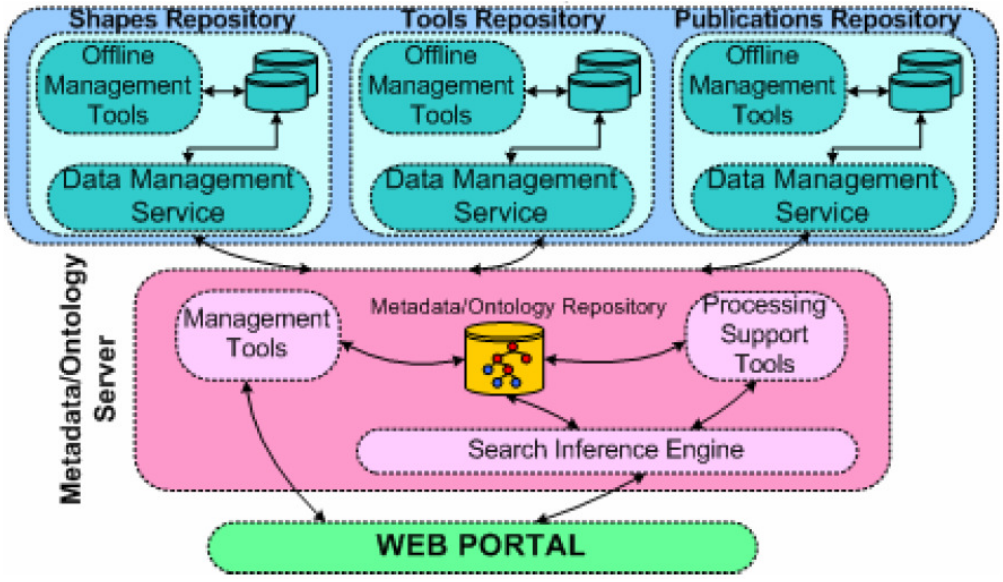


Figure 3.16 L'architecture générale de Digital Shape Workbench [Albertoni *et al.*, 2005].

[29] http://www.aimatshape.net

Cette architecture formelle vise à établir les bases d'une plate-forme de recherche pour la modélisation, le stockage et l'exploitation de formes par des outils logiciels spécifiques. Au cœur de l'architecture se trouve le serveur responsable de la représentation des connaissances organisées en ontologies relatives aux formes. L'architecture est munie d'une série de services permettant d'inférer des connaissances et de rechercher des formes 3D.

Les modèles et les outils logiciels sont répartis de manière distribuée sur le réseau et sont accessibles à travers une API commune. Le moteur d'inférence et de recherche (*Search Inference Engine*) s'attache à offrir une bonne qualité de service lors de la recherche des formes dans la base de connaissances. Les auteurs misent sur l'utilisation des ontologies modélisées en OWL [McGuinness *et al.*, 2004] afin de pouvoir mettre en œuvre des raisonnements déductifs à l'aide de moteurs d'inférence spécifique OWL tels que Racer[30]. Les mécanismes d'inférence permettent d'éviter les problèmes inhérents aux processus de recherche classique (synonymie, utilisation de termes différents, annotation incomplète, etc.). Le moteur de recherche propose une interface unifiée pour l'accès aux sémantiques relatives aux formes (*Shapes Repository*), aux outils d'analyse et traitement (*Tools Repository*) et aux publications scientifiques réalisées en rapport avec la caractérisation de formes et des outils associés (*Publications Repository*).

Les outils du module *Management Tools* sont employés pour la création, l'édition, la validation, le téléchargement, la navigation et la visualisation des ontologies et des sémantiques associées aux formes, aux outils et aux publications. Une première expérimentation d'un outil issu de cette catégorie, permettant d'annoter et transformer quelques formes 3D basiques (maillages triangulaires), est présentée dans [Danovaro *et al.*, 2007].

L'intégration au sein de l'entrepôt de connaissances des informations relatives aux outils laissent les auteurs envisager dans le futur la formulation de requêtes qui ne concernent pas uniquement une sélection de formes selon un critère sémantique, mais également de mettre en place des traitements en vue d'une transformation de l'état initial d'une forme. Ainsi des requêtes telles que : « transformer une forme d'une représentation 3D à une autre », ou « mesurer la similarité entre deux formes », peuvent être évaluées par les outils dans le module *Processing Support Tools* car les informations nécessaires à l'utilisation automatique des outils de traitement sont stockées dans l'entrepôt *Tool Repository*.

Le processus qui a mené à établir l'architecture DSW constitue une première étape dans la quête des auteurs vers le développement d'une plate-forme à large échelle pour la formalisation, le traitement et le partage des connaissances sémantiques relatives aux formes 3D et à leurs futures applications.

Cette première partie consacrée à l'analyse de critères relatifs à la recherche de données 3D montre une large variété de solutions concernant l'analyse des propriétés de bas niveau de la géométrie, de l'apparence ou de la topologie des données 3D. Cependant, la dimension sémantique des données 3D reste assez peu explorée.

Dans la section suivante nous nous intéressons aux systèmes qui, en plus de la possibilité de retrouver telle ou telle donnée 3D, permettent de les intégrer au sein de nouvelles scènes 3D.

[30] http://www.racer-systems.com

3.2. Réutilisation

La réutilisation est associée à un processus (semi)automatique d'intégration d'une donnée au sein d'un nouveau contenu autre que celui d'origine. La réutilisation de contenus 3D est primordiale car la construction des objets 3D requiert beaucoup de temps et un investissement conséquent. Les concepteurs disposent aujourd'hui d'un large choix d'environnements tels que *Maya* par *Alias|wavefront*[31] ou *3D Studio Max* par *Discreet*[32] qui permettent de créer des objets 3D à un haut niveau de réalisme. Cependant, les possibilités de réutilisation de scènes 3D au sein de différentes applications restent à ce jour limitées, et ce en raison d'une non dissociation quasi systématique de la sémantique et de la géométrie.

3.2.1. Typologie de réutilisation

Dans notre vision la réutilisation d'une donnée 3D correspond à un processus décomposable en deux étapes :

- la formulation d'une requête pour rechercher les donnée en question, et
- l'intégration de ces données au sein d'une nouvelle scène 3D.

Suivant la manière dont l'intégration de données se réalise au sein de nouvelles scènes nous pouvons dégager trois types de réutilisation :

- **réutilisation basique de l'objet** – l'objet est inclus tel quel (avec l'ensemble de ses caractéristiques géométriques, d'apparence et éventuellement sémantique) dans la nouvelle scène

  La réutilisation correspond au processus qui extrait un objet 3D d'une scène existante et l'inclut tel quel dans une nouvelle. Par exemple, le même objet 3D modélisant un bâtiment peut être utilisé dans plusieurs scènes. Il s'agit du type de réutilisation le plus répandu. Les critères sur lesquels se basent la recherche et la réutilisation sont généralement simples et l'acte de réutilisation consiste à référencer l'adresse physique du fichier contenant l'objet réutilisable. Ce type de réutilisation est celui qu'on rencontre le plus souvent dans les travaux s'intéressant à la réutilisation de contenus 3D. Les travaux que nous présentons dans cette section s'inscrivent dans cette catégorie.

- **réutilisation avec apport sémantique** – la géométrie de l'objet est enrichie avec des informations sémantiques relatives au domaine d'application de la nouvelle scène

  Ce type de réutilisation concerne les transformations d'ordre sémantique subies par un même modèle 3D utilisé dans différentes applications, ou tout du moins, à des fins différentes. Selon le domaine d'application ou, plus généralement le besoin, des informations sémantiques différentes peuvent être associées – en tant que métadonnées – au même objet 3D. Ainsi, alors que dans une application d'aménagement urbain on associe à un objet 3D représentant un bâtiment le nombre de personnes pouvant y vivre, dans une application d'évaluation du risque sismique, ce même objet doit se voir associer des informations sur sa vulnérabilité. Au niveau du langage X3D, il existe certains outils – que nous détaillerons dans la suite de cette section – qui permettent d'envisager la réutilisation avec apport sémantique. Cependant, à ce jour, nous n'avons pas

[31] http://www.alias.com
[32] http://www.autodesk.com

rencontré de systèmes visant à associer de manière automatique plusieurs profils sémantiques aux objets 3D en fonction d'un domaine d'application donnée.

- **réutilisation par adaptation** – la géométrie et l'apparence de l'objet sont modifiées afin de permettre un déploiement de la scène en correspondance avec les contraintes matérielles et les besoins informationnels du public ciblé.

  Ce type de réutilisation nécessite généralement une série d'adaptations portant sur le contenu de l'objet 3D et il est traité de manière approfondie dans le chapitre suivant.

Dans la suite de cette section nous nous intéressons aux supports de réutilisation existants notamment dans le cadre du standard de description X3D. Le standard offre un nombre d'outils qui facilite la réutilisation du contenu 3D. La mise en œuvre de tels outils de réutilisation est illustrée à travers plusieurs systèmes de génération automatique de scènes 3D.

3.2.2. Les supports de réutilisation dans X3D

Au niveau du langage X3D, deux mécanismes assurent la mise en œuvre de la réutilisation de données 3D ou de leurs caractéristiques : la réplication d'un même élément ou d'une caractéristique (apparence, géométrie, sémantique) à plusieurs endroits dans la scène (`DEF/USE`) et l'instanciation à base de prototypes (`PROTO/EXTERN PROTO`).

3.2.2.1. Le mécanisme DEF/USE

Le premier type de réutilisation s'appuie sur la décomposition sous forme d'une arborescence cyclique d'une scène 3D. Il consiste dans la réutilisation d'un même nœud dans le graphe de scène. Le nœud n'est pas recopié mais relié à plusieurs nœuds parents. Ainsi, le contenu représenté par le nœud en question apparaît à plusieurs endroits dans la scène. L'attribut `DEF` définit un nom qui identifie de manière unique le nœud dans son contexte. L'utilisation de l'attribut `USE` dans un nœud signifie que lors de la construction du graphe de scène le nœud en question est remplacé par celui dont le nom correspond à la valeur de l'attribut `USE`.

Dans la Figure 3.17, nous illustrons cette technique de réutilisation au sein d'une scène représentant des parcelles de terrains peuplés d'arbres. Afin d'optimiser la réutilisation, nous considérons une décomposition hiérarchique en trois niveaux : la parcelle d'arbres, la rangée d'arbres et l'arbre lui-même. Dans le document X3D de la Figure 3.17, nous traduisons cette organisation en imbriquant les définitions de différentes entités (parcelle – lignes 3-31, rangée – lignes 4-26, arbre – lignes 5-21). La définition d'une rangée implique la représentation de plusieurs arbres. En utilisant le mécanisme `DEF/USE` nous définissons uniquement le premier arbre de la rangée (lignes 5-21), les autres sont introduits en réutilisant le premier, en indiquant son identifiant dans l'attribut `USE` des éléments leur correspondant (lignes 22-24). Le même procédé de description est utilisé pour définir une parcelle en réutilisant autant de fois que nécessaire la première rangée (voir lignes 3-31 pour la définition de la première parcelle, et lignes 27-29 pour la réutilisation). Il nous est également possible de réutiliser la définition de la parcelle à plusieurs endroits dans le document (voir lignes 32-34).

Cette technique présente l'avantage que, dans le graphe de scène, une seule instance de nœud est construite. Chaque élément introduit par un `USE` n'est pas introduit dans le graphe, en revanche cela se traduit par l'introduction d'un lien entre le père de l'élément et le nœud référencé par l'attribut `USE`. Dans l'exemple décrit ici, en adoptant le mécanisme de réutilisation `DEF/USE`, nous utilisons uniquement 3 nœuds à la place des 32 nœuds qu'on aurait eu si chaque arbre était introduit indépendamment. Le mécanisme de réutilisation présente également une

limitation d'ordre technique qui rend impossible l'utilisation de ce mécanisme à travers plusieurs documents X3D. Afin de remédier à cette limitation une alternative de réutilisation est proposée au moyen du concept de *prototype*.

```
01: <X3D version="3.0" profile="Interchange">
02:   <Scene>
03:     <Group DEF='PARCELLE'>
04:       <Group DEF='RANGEE'>
05:         <Group DEF='ARBRE'>
06:           <Transform scale = '0.26 0.26 0.26'>
07:             <Transform>
08:               <Shape>
09:                 <Appearance><Material diffuseColor='.5 .25 0'/></Appearance>
10:                 <Cylinder height='1.25' radius='.25'/>
11:               </Shape>
12:             </Transform>
13:             <Transform translation ='0 1.25 0'>
14:               <Shape>
15:                 <Appearance><Material diffuseColor='0 .7 0'/></Appearance>
16:                 <Cone bottomRadius='1.25' height='2.0'/>
17:               </Shape>
18:             </Transform>
19:             …
20:           </Transform>
21:         </Group>
22:         <Transform translation='2.0 0.0 0.0' >
23:           <Group USE='ARBRE'/>
24:         </Transform>
25:         …
26:       </Group>
27:       <Transform translation='1 0 1'>
28:         <Group USE='RANGEE'/>
29:       </Transform>
30:       …
31:     </Group>
32:     <Transform translation="0 0 7">
33:       <Group USE="PARCELLE"/>
34:     </Transform>
35:   </Scene>
36: </X3D>
```

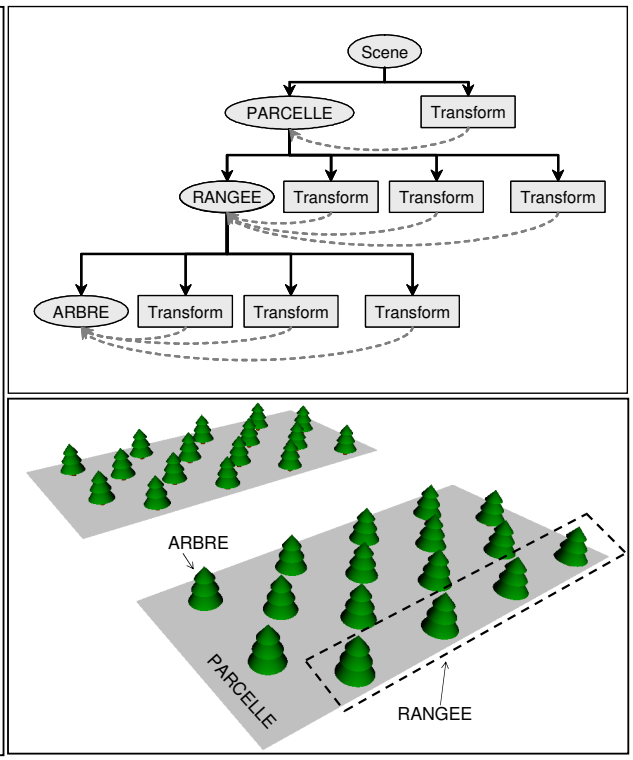


Figure 3.17 Réutilisation des éléments géométriques à l'aide du mécanisme DEF/USE de X3D.

3.2.2.2. Instanciation de prototype

La création et l'utilisation de prototypes permettent de mieux contrôler la duplication du contenu. Chaque prototype peut définir un ensemble de champs d'entrée et sortie qui permet de paramétrer les instances et d'interagir avec les autres composants d'une scène 3D de façon homogène.

Par rapport au mécanisme `DEF/USE`, un prototype peut être réutilisé à travers différents documents et il permet également que l'on personnalise chaque nouvelle instance du contenu. En contrepartie, l'utilisation au sein d'une scène d'un prototype est moins efficace. Ceci est dû au fait que pour chaque instance, un nœud conservant les propriétés de celle-ci est inclu dans le graphe de scène. Néanmoins, il est possible de combiner les deux techniques afin d'obtenir de meilleurs résultats. Si on doit instancier plusieurs fois le même prototype avec des paramètres identiques (comme c'est le cas des arbres dans l'exemple que nous avons présenté précédemment), il suffit d'introduire la première instance à travers l'instanciation de prototype et les autres instances peuvent être introduites à l'aide des éléments dont l'attribut `USE` indique le nom de la première instance. Nous illustrons cette technique dans la Figure 3.18. À gauche, nous avons la définition de la scène modifiée et à droite, la définition du prototype d'arbre. La nouvelle version de la scène utilise le prototype `Arbre_proto` (dont la définition est donnée dans le fichier `abre.x3d`) afin d'introduire un arbre dans la scène. L'utilisation d'un prototype suppose deux étapes : la déclaration du prototype (lignes 3-5 du fichier `parcelle.x3d` dans la partie gauche de la Figure 3.18) et l'instanciation du prototype (ligne 8 du fichier `parcelle.x3d`). Nous associons à cette instance l'identifiant `ARBRE` (ligne 8 `DEF="ARBRE"`) ce qui nous permet de la

référencer par la suite dans la construction de rangées (voir par exemple ligne 10 du fichier `parcelle.x3d - USE="Arbre"`).

La notion de prototype 3D est fondamentale dans le processus de réutilisation de données 3D. Par rapport à la réutilisation à base d'objet, la réutilisation à base de prototype présente l'avantage de pouvoir personnaliser les paramètres de chaque instance réutilisée.

Dans la section suivante, nous présentons des approches de réutilisation de données 3D qui s'appuient notamment sur la notion de prototype, même si, cette notion n'est pas toujours représentée en utilisant les éléments du standard X3D. Dans certaines propositions que nous présentons par la suite, des modules spécifiques instancient et associent des représentations prototypées aux éléments constituant une scène 3D en vue d'une réutilisation à base de génération dynamique de scènes.

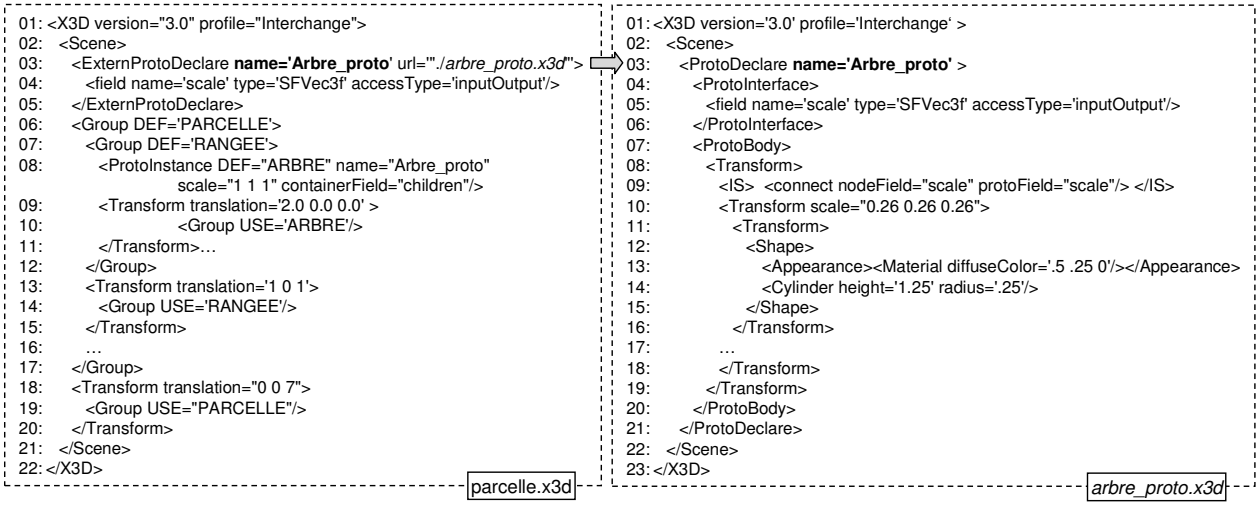


Figure 3.18 Technique de réutilisation combinant les prototypes aux mécanismes DEF/USE en X3D.

3.2.3. Systèmes de réutilisation à base de génération dynamique des données 3D

Les travaux que nous présentons ici portent sur la génération automatique de scènes 3D par réutilisation de contenus. Ces contenus sont obtenus à partir de descriptions de haut niveau (*i.e.* reposant sur des concepts et des propriétés de domaine) et faisant abstraction de la dimension géométrique des univers 3D.

Nous nous intéressons à ce type de travaux car l'étude des mécanismes liant une représentation sémantique à une représentation 3D laisse envisager des solutions de réutilisation accessibles à un large public. En effet, en s'appuyant sur des critères sémantiques et en utilisant des *prototypes* pour traduire les concepts sémantiques en données 3D, ces solutions ne nécessitent quasiment d'aucune connaissance relative aux standards et aux outils de construction 3D. Des experts de domaine peuvent ainsi créer des nouvelles scènes en réutilisant les modèles d'objets 3D existants.

3.2.3.1. ISAS [Oliverio et al.*, 2007]*

Dans le cadre du projet ISAS [Oliverio *et al.*, 2007], déjà présenté dans le chapitre précédent, les auteurs considèrent la génération de scènes 3D en partant d'une ontologie de domaine (décrivant le campus de l'Université de Floride) dont les concepts représentent les objets de la scène. Cette ontologie du domaine est associée à une ontologie dédiée à la

représentation qui contient des concepts géométriques simples ou complexes construits à partir de primitives géométriques présentes dans VRML ou X3D.

Cette séparation de la sémantique et de la représentation est exploitée par les auteurs afin de générer dynamiquement des scènes 3D en sélectionnant à la volée les objets à présenter. Le moteur de génération repose sur le système Lyra [Beck, 2005] de gestion d'ontologies *OWL-DL*.

Une particularité de cette proposition réside dans la dimension géospatiale des attributs des concepts de l'ontologie du domaine. Une carte géoréférencie les bâtiments sur le campus. Ces informations sont incluses dans la scène à travers des attributs spatiaux associés aux transformations rigides (translations, rotations) appliquées aux primitives 3D matérialisant les concepts de l'ontologie dans la scène.

Cependant, dans les travaux publiés, les auteurs ne précisent pas si l'association entre les concepts de deux ontologies (de domaine et de représentation) est fixe ou est décrite par des règles de mise en relation et donc flexible et évolutive. De même, l'intégration des informations géographiques dans le processus de génération n'est pas formalisée.

Même si cette proposition est orientée vers l'illustration de l'intégration de données issues de plusieurs sources de données (ontologie de domaine, représentation 3D, données géographiques), elle n'offre pas un cadre générique pour la réutilisation à base de génération dynamique de scènes 3D. Elle ne propose pas de formalismes transférables et applicables à d'autres domaines d'application.

3.2.3.2. Curriculum visualization in 3D [Sommaruga et al.*, 2007]*

Le processus de génération d'un environnement 3D pour la visualisation des informations liées aux activités d'enseignement dispensées à l'Université de Sciences Appliquées du Sud de la Suisse (SUPSI) est présenté dans [Sommaruga *et al.*, 2007]. Les curricula du département et les modules proposés aux étudiants sont présentés accompagnés d'un ensemble d'information (nombre d'heures, nombre de crédits, etc.) présenté sous une forme graphique, ce qui permet, selon les auteurs, une vue intuitive, simple et accessible aux étudiants ayant un besoin informationnel précis. L'utilisation d'un espace 3D donne aux étudiants la possibilité d'avoir une vision immédiate, claire et complète des différentes possibilités de formation au sein de SUPSI.

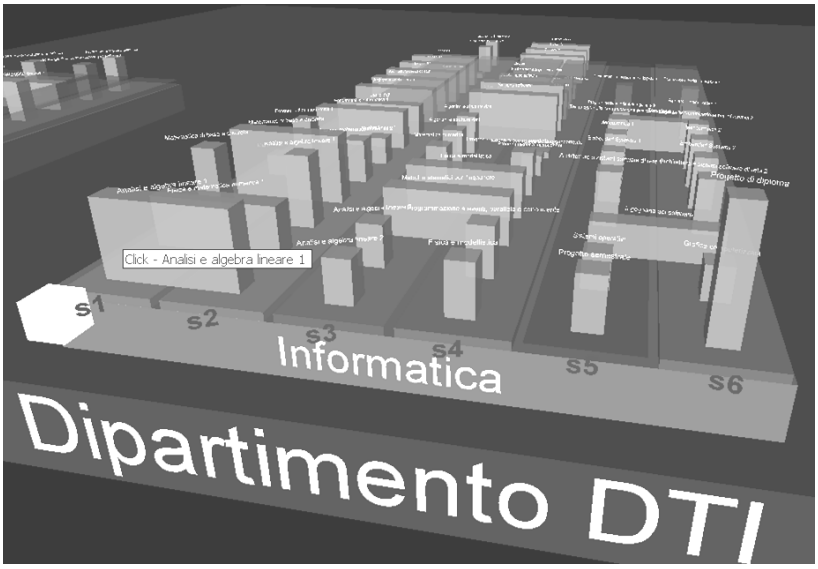


Figure 3.19 Aperçu de l'environnement SUPSI 3D [Sommaruga et al., 2007].

Les modules sont visualisés dans un environnement virtuel grâce à une métaphore géométrique qui est en fait une simplification de la métaphore de la ville [Dieberger *et al.*, 1998]. Chaque département est un quartier (voir Figure 3.19). Chaque curriculum d'un département correspond à un pâté de maisons divisé en 6 régions (un par semestre). Chaque module est une bâtisse dans le pâté de maison placé dans la région spécifique suivant le semestre d'enseignement, et munie d'un libellé indiquant le nom du module. La hauteur de la bâtisse indique le nombre de crédits, tandis que sa largeur concerne le nombre d'heure. Un complément d'information est fourni dès lors que l'utilisateur interagit avec l'environnement au moyen de cliques de souris.

Au centre du système de visualisation se trouve le standard X3D. Ainsi, la génération de l'environnement 3D s'appuie sur l'encodage XML du standard X3D. D'une part, les informations sur le curriculum et les modules sont extraites depuis une base de données en format XML. D'autre part, une feuille de transformation décrite selon le standard W3C XSLT est appliquée aux données issues de la base de données. L'utilisation d'une feuille de style XSLT est fortement inspirée de l'approche proposée par [Polys, 2003]. Cette dernière concerne la visualisation 3D de la molécule du cholestérol décrite initialement dans le standard *Chemistry Markup Language* (CML)[33] et transformée grâce à une feuille de style générique pour la transformation de documents CML en documents X3D.

Ce travail utilise le standard XSLT pour déterminer la manière dont la nouvelle scène est modélisée et organisée. Ceci demande à ce que les utilisateurs qui veulent faire de la réutilisation de donnés doivent disposer des connaissances suffisamment développées du standard XSLT afin de pouvoir définir des règles de traduction entre les éléments du curriculum et une représentation 3D. Toutefois, le travail illustre la possibilité de mettre en œuvre des techniques de réutilisation en s'appuyant uniquement sur des technologies Web standardisées : XML, XSLT et X3D.

3.2.3.3. OntoSphere3D [Bosca et al.*, 2007]*

Dans [Bosca *et al.*, 2007], les auteurs s'intéressent à la génération de scènes 3D illustrant les concepts et les relations existantes au sein d'une ontologie. Un composant logiciel réutilisable a été réalisé afin de permettre aux utilisateurs d'explorer et interagir dans un espace 3D avec les éléments (concepts, relations, instances) d'une ontologie. Une approche 3D favorise plus la visualisation d'ontologies que les approches 2D car elle permet une meilleure projection de la multitude d'informations véhiculées par une ontologie. En effet, plusieurs avantages indéniables sont mis en avant : une visualisation similaire à celle rencontrée dans le monde réel, une dimension supplémentaire pour convoyer plus d'information, une mise en évidence de certains objets importants en les plaçant au plus près de l'utilisateur en jouant sur la dimension spatiale...

Les ontologies supportées sont décrites en OWL-DL. Les auteurs ont associé à chaque construction du langage OWL-DL (classes, propriétés, restrictions, cardinalités, disjonctions, …) une représentation 3D adéquate, ainsi qu'un ensemble d'animations 3D prédéfinies pour faciliter la navigation de l'utilisateur au sein de la scène. Une attention spéciale a été accordée au grand nombre d'éléments que comporte une ontologie et à leur organisation spatiale. La complétude et la lisibilité des informations présentées sont deux aspects importants directement liés à l'efficacité et l'utilisabilité de cette nouvelle méthode de visualisation. Afin de trouver un bon compromis entre ces deux aspects, les auteurs s'appuient sur l'utilisation d'indices visuels (taille, couleur, niveau de transparence) ainsi que sur une stratification des éléments du langage OWL-DL couplée à un mécanisme d'accès progressif à l'information.

[33] http://www.xml-cml.org

Par exemple, dans une hiérarchie *isA* chaque niveau de la hiérarchie est associé à une couleur. L'élément qui se trouve au centre de la visualisation est noir, les éléments *non expansés* sont blancs et leur taille est directement proportionnelle au nombre d'éléments qu'ils représentent. À chaque moment OntoSphere3D présente uniquement trois niveaux de la hiérarchie afin d'éviter une surcharge visuelle et cognitive. Ces faits sont illustrés dans la Figure 3.20 où sont illustrés les concepts pour décrire la composition d'une pizza.

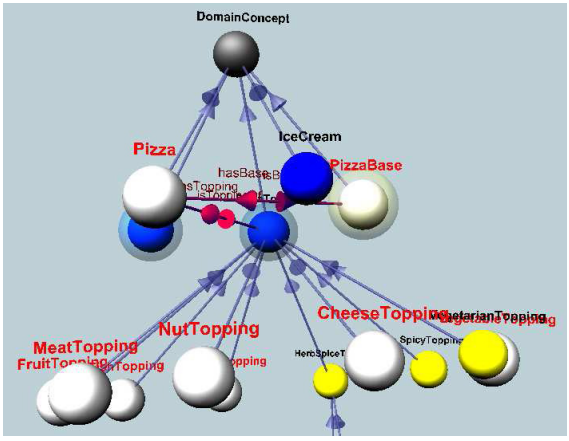


Figure 3.20 Visualisation d'une hiérarchie isA dans une ontologie en OntoSphere3D [Bosca et al.*, 2007].*

La sélection dynamique des éléments à inclure dans une scène est une partie importante du processus de visualisation. OntoSphere3D emploie plusieurs modèles de scène prédéfinis qui présentent et organisent l'information sur l'écran, chacun associé à un niveau de détail bien défini. Chaque modèle de visualisation est en effet associé à une tâche de visualisation (visualisation d'une classe, visualisation d'une instance, etc.). Il y a autant de tâches de visualisation que de méta-concepts dans l'ontologie (classe, instance, relation, propriété, …).

OntoSphere3D est implémenté (voir Figure 3.21) en utilisant une représentation interne d'un modèle (*Internal Model*) qui dépend du niveau d'abstraction d'une ontologie (ses méta-concepts – *Ontology Abstraction*) et un ensemble de modèles de scènes (*Scene3D*) qui présentent les informations contenues et la structure de l'ontologie à différents niveaux de détail. Les scènes sont automatiquement générées en interprétant le modèle interne et en appliquant la mise en correspondance entre concepts (*Concept*) et relations (*Branches*), définis par le modèle de scène choisi pour un élément de l'ontologie et un niveau de détail donné.

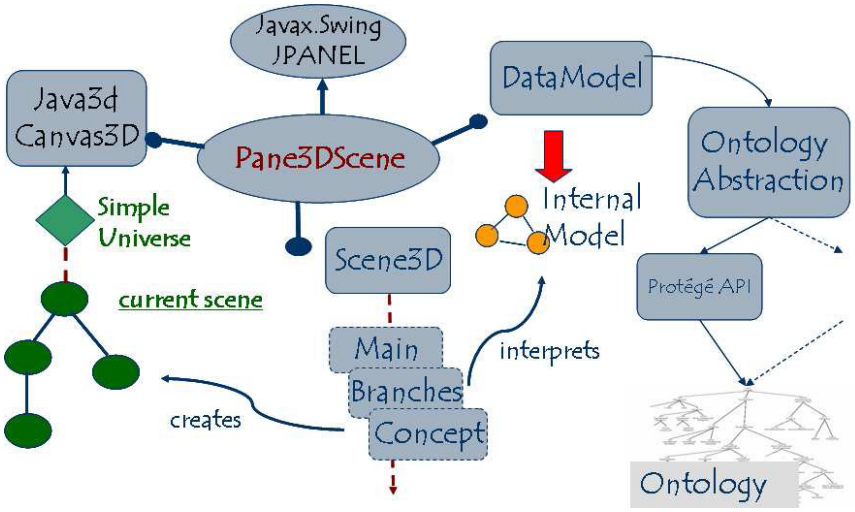


Figure 3.21 L'architecture du système OntoSphere3D [Bosca et al.*, 2007].*

L'intérêt de cette approche réside dans le fait qu'il est possible de mettre en relation un ensemble de concepts et de relations totalement indépendantes d'une représentation 3D et un ensemble de primitives géométriques et d'outils de navigation 3D. Cependant, il est regrettable qu'il n'y ait pas de formalisme permettant de décrire comment la mise en correspondance entre les concepts du modèle interne et la représentation graphique est réalisée. De même, les auteurs ne précisent pas comment des représentations à base de modèles de scènes prédéfinies s'emboîtent à différents niveaux de détail. Par exemple, il n'est pas clairement expliqué si un modèle de visualisation utilisé lorsque le focus de la visualisation est une instance (*MV_INST*) réutilise un autre modèle utilisé lorsque dans le focus se trouve une propriété (*MV_PROP*). Un autre reproche que nous pouvons faire est l'utilisation d'une composante logicielle qui réalise directement le rendu visuel. Ainsi, il est impossible de reproduire et diffuser largement sur le Web les scènes générées.

3.2.3.4. OntoWorld [Kleinermann et al.*, 2005]*

OntoWorld [Kleinermann *et al.*, 2005], créé par le groupe de recherche *VR-WISE* de *Vrije Universiteit Brussel*, s'intéresse à l'implication des experts de domaine dans la construction des environnements virtuels. L'approche utilise les ontologies comme un moyen intuitif et orienté domaine pour la génération de mondes virtuels.

OntoWorld se démarque d'une approche classique de création d'un monde virtuel car il permet d'accumuler des connaissances relatives aux domaines, de décrire les mondes virtuels plus en termes de concepts de domaines que de constructions géométriques et de faciliter la génération de mondes virtuels. Le processus de génération est rendu possible en dérivant un certain nombre de propriétés pour chaque objet à partir de l'ontologie du domaine. Par exemple, il est possible de décrire un lit dans l'ontologie de domaine selon ses dimensions réelles : hauteur, largeur, profondeur. Ces informations permettre de décider des propriétés d'une primitive géométrique de type boîte qui peut être utilisée pour représenter le lit. De plus, toutes ces informations sur les objets et leurs relations peuvent être stockées et utilisées a posteriori lors des futures utilisations de la scène (interrogation, filtrage, etc.).

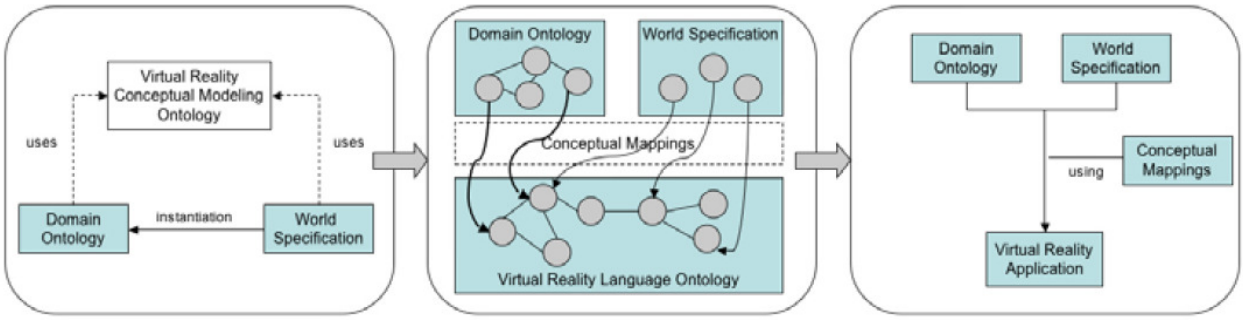


Figure 3.22 Aperçu de l'architecture d'OntoWorld [Kleinermann et al.*, 2005].*

La création d'un monde virtuel en OntoWorld suit un processus en trois étapes : l'étape de *spécification*, l'étape de *mise en correspondance*, l'étape de *génération*. Les différentes étapes sont illustrées dans la Figure 3.22. Dans l'étape de *spécification*, le monde est spécifié au niveau conceptuel en utilisant les ontologies de domaine. Le monde est défini en instanciant les concepts de l'ontologie et en décrivant les relations entre elles, leur disposition spatiale et la manière dont elles interagissent. L'étape de *mise en correspondance* indique comment les concepts et les instances de domaine sont représentés dans le monde virtuel. Ceci est réalisé en associant aux concepts et aux instances des primitives géométriques, des contraintes de

placement, etc. OntoWorld permet de définir la position d'un objet dans la scène soit en indiquant les coordonnées absolues, soit en indiquant un placement relatif (*RightOf*, *Posterior*, *Anterior*, *Above*, *Bellow*, …) à un autre objet de la scène (comme illustré dans la Figure 2.14 présentée dans le chapitre précédent).

Cette solution est d'un niveau de généricité assez important, car elle permet de séparer la sémantique de la représentation géométrique et la manière dont un concept est traduit en objet 3D. Cependant, elle ne traite pas de l'organisation de la scène dans son ensemble. L'ontologie de *représentation 3D* associe des coordonnées absolues aux différents objets disponibles (cubes, sphères, …). Ainsi, dans la scène 3D, tous les objets se retrouvent au même niveau hiérarchique et ne sont pas organisés convenablement dans le graphe de scène afin d'optimiser la structuration et, lors du déploiement, la visualisation efficace de la scène.

3.3. Synthèse

La première partie de ce chapitre dédié à la catégorisation et l'analyse de différents critères de recherche de données 3D, nous a permis de mettre en exergue certaines difficultés inhérentes au processus de caractérisation de données 3D.

Les critères proches du niveau signal (critères géométriques, critères d'apparence, critères topologiques) sont relativement faciles à évaluer, cependant ils ne peuvent pas servir directement pour caractériser d'un point de vue sémantique l'objet 3D. Ceci est dû au fait que de tels critères s'appuient sur une caractérisation numérique des dimensions d'un objet 3D. L'appariement de deux objets est décidé en fonction de la distance entre les descripteurs numériques leur étant associés. En revanche, si l'un des deux objets qui sont appariés dispose d'un certain nombre d'information sémantique qui lui est associé on peut envisager de propager cette connaissance sémantique à l'autre objet. Toutefois, ce que nous retenons de cette première partie est le fait, qu'à ce jour, les besoins en termes d'informations sémantiques associées aux données 3D ne peuvent pas être satisfaits sans l'intervention plus ou moins intrusive de l'utilisateur.

Du fait que certaines propositions matérialisent les caractéristiques sémantiques des données en utilisant des langages de descriptions tels que RDF ou MPEG-7, il est possible de faire appel à des langages d'interrogation spécifiques pour effectuer des recherches sur les données 3D. Néanmoins, nous regrettons **l'absence de propositions offrant un cadre formel pour l'interrogation de connaissances sémantiques associées à différents niveaux de description d'une scène 3D : la géométrie, l'apparence, la sémantique locale** – associée à l'objet 3D, **et la sémantique générale** – associée à l'entité représentée par l'objet 3D.

La deuxième partie de ce chapitre a été consacrée à l'étude des approches de réutilisation. Après la présentation des moyens de réutilisation présents nativement au niveau du standard X3D, autour duquel nous construisons notre étude et notre future proposition, nous analysons quatre propositions qui explorent la génération dynamique de contenu comme moyen de réutilisation.

Ce que nous observons comme trait commun à toutes les approches présentées est la méthodologie suivie dans le processus de génération.

- le système extrait les informations qui sont organisées suivant une structure qui correspond au modèle abstrait de la scène ;

- à chaque élément du modèle abstrait on applique une métaphore de visualisation (l'élément est transformé suivant les besoins applicatifs dans un ensemble de primitives 3D) en obtenant une scène 3D.

Cependant, dans l'ensemble de propositions nous avons noté **un manque des moyens spécifiques facilitant la formulation des requêtes qui indiquent la liste des objets à réutiliser** dans le modèle abstrait de la scène. De même, à l'exception de OntoWorld [Kleinermann *et al.*, 2005], les propositions présentées ici, **ne formalisent pas au niveau documentaire la mise en correspondance entre les concepts de la scène abstraite et les objets de la scène 3D**.

Les propositions mettent un accent important sur la visualisation en négligeant les aspects relatifs à l'enrichissement sémantique de la scène générée avec les informations sémantiques dont on dispose au niveau abstrait. En plus de ces choix de visualisation il faut prêter une attention particulière aux autres facteurs (la quantité d'information, la disposition spatiale, le niveau de détail, la qualité des textures) dont dépend l'efficacité de la visualisation afin d'éviter la surcharge cognitive vis-à-vis de l'utilisateur et la surcharge de calcul vis-à-vis du dispositif de visualisation. Ces points peuvent être considérés dans le processus de réutilisation au moment de la génération, ou appliqués *a posteriori* sur la scène générée, en fonction du dispositif d'accès ou du profil de l'utilisateur visualisant la scène.

Le chapitre suivant est consacré entièrement au recensement des différentes techniques et solutions logicielles associées, qui œuvrent pour la génération et/ou la diffusion adaptée de la scène 3D en rapport avec les attentes de l'utilisateur et les contraintes matérielles du contexte de diffusion.

4. L'adaptation de données 3D

Les données 3D sont caractérisées par une grande complexité, autant d'un point de vue géométrique que spatial ou encore multimédia (textures, images ou vidéo). Une même donnée 3D peut ainsi se révéler difficile à déployer dans sa forme initiale sur l'ensemble des supports capables de visualiser à présent de l'information 3D (ordinateurs, TV, téléphones, etc.). Une donnée 3D, dans sa version initiale, peut également contenir un nombre important d'informations qui ne sont pertinentes que pour une partie des utilisateurs. Dans ces cas, une transformation de la scène est souhaitée afin de rendre le processus de déploiement le moins contraignant possible. Ces constats nous encouragent à considérer le processus d'adaptation parmi les opérations de base qu'une plate-forme de gestion de données 3D doit offrir.

L'adaptation d'une donnée 3D correspond à l'ensemble des transformations subies par la donnée initiale afin que ses caractéristiques (présentation, taille mémoire, contenu informatif) soit en accord avec un ensemble de contraintes imposées par les besoins de visualisation et de diffusion, comme par exemple les attentes de l'utilisateur, les caractéristiques de son dispositif d'accès, le débit du réseau de diffusion, etc.

Ainsi, lorsque l'on s'intéresse au processus d'adaptation, plusieurs notions doivent être prises en compte :

- le sujet de l'adaptation – la géométrie, l'apparence et la structure logique des données 3D ;
- les éléments déclencheurs de l'adaptation – une ou plusieurs contraintes matérielles, logicielles ou relatives aux points d'intérêts, et les préférences de l'utilisateur ;
- les paramètres du processus – la technique d'adaptation, le coût de l'adaptation, la perte tolérée, etc.

Dans ce chapitre, nous nous intéressons plus particulièrement aux techniques d'adaptation existantes et à la manière dont elles peuvent être appliquées aux différentes parties d'une scène 3D afin d'obtenir une version de celle-ci plus allégée et qui cible au mieux les besoins des utilisateurs. Dans cet état de l'art, nous n'abordons pas la question de la représentation du contexte ni celle de son acquisition qui viennent en amont du processus

d'adaptation. Indépendamment d'une organisation ou d'une représentation spécifique du contexte [Schilit *et al.,* 1994][Pascoe, 1998][Dey *et al.,* 2001][Lemlouma, 2004][Kirsch-Pinheiro *et al.,* 2005][Bucur *et al.,* 2005][Pittarello *et al.,* 2005][Schwinger *et al.,* 2005][Vetro *et al.,* 2005][Hernandez *et al.,* 2007] nous considérons ici que le processus d'adaptation dispose entre autres d'une liste de propriétés descriptives du contexte servant de paramètres guidant les choix d'adaptation.

Dans ce chapitre, nous analysons les techniques d'adaptation multimédia présentant des caractéristiques intéressantes pour une éventuelle extension au domaine 3D. Ensuite, nous discutons des types de méthodes d'adaptation spécifiques aux données 3D en nous intéressant aux propositions considérant la géométrie et l'apparence des données 3D. Nous présentons également quelques propositions s'intéressant à l'adaptation des scènes 3D dans leur ensemble (modification de la disposition spatiale des données au sein d'une scène). Tout au long de ce chapitre nous essayons de souligner les apports de l'existant et les verrous qui empêchent d'aboutir à une architecture générique qui supporte la mise en œuvre transparente des adaptations concernant les données 3D.

4.1. Travaux d'adaptation multimédia

Nombreux sont les travaux d'adaptation qui ont vu le jour depuis l'avènement des documents multimédia sur le Web [Bulterman, 1998][Boll *et al.*, 1999][Villard *et al.*, 2000][Ossenbruggen *et al.*, 2001][Libsie *et al.*, 2002][Bilasco *et al.*, 2005a][Benbernou et *al.*, 2005]. Cette effervescence autour des documents multimédia, gourmands en ressource, a été intensifiée lors de l'ouverture du Web aux dispositifs d'accès caractérisés par des ressources limitées en termes de capacités de calcul, taille d'affichage et d'autonomie [Mohan *et al.*, 1999][Hoi *et al.*, 2003][Lauf *et al.*, 2005][Raento *et al.*, 2005].

Ces propositions offrent des informations pertinentes sur les besoins relatifs à l'émergence d'une solution d'adaptation générique, valable indépendamment du type de support spécifique 2D ou 3D. Parmi celles-ci, nous notons : la prise en compte du profil utilisateur et du contexte de diffusion dans l'élaboration d'une stratégie d'adaptation, l'importance de la sémantique et le besoin de l'utilisateur, en termes d'information visuelle ou autres, etc.

Dans cet état de l'art, consacré plus spécifiquement aux documents 3D, nous présentons uniquement deux propositions issues du domaine 2D qui mettent en avant des plates-formes d'adaptation facilement transposables à l'adaptation de scènes 3D.

4.1.1. NAC [Lemlouma, 2004]

L'architecture *Negociation and Adaptation Core* (NAC) [Lemlouma, 2004] vise le déploiement adapté de documents multimédia, en s'appuyant sur des proxies capables de transformer le document afin qu'ils satisfassent les contraintes de diffusion. L'architecture s'appuie sur une modélisation du contexte de diffusion au moyen de descriptions CC/PP. L'organisation sémantique des contraintes de l'environnent, en utilisant des modèles de description tels que CC/PP, permet d'automatiser le traitement des contraintes et, par conséquent, d'automatiser l'exécution des règles et des processus d'adaptation qui dépendent des caractéristiques du client et de son contexte global.

L'architecture générale de NAC est illustrée dans la Figure 4.1. Cette architecture est composée de cinq entités qui coopèrent. Le *proxy de communication* assure la communication entre le client et le serveur. Les réponses du serveur doivent être adaptées ou modifiées avant

qu'elles soient délivrées à leur destinataire. Pour cela, le *proxy de communication* assure également la communication orientée négociation avec un module côté client (le module UCM). Le module UCM (*User Context Model*) renseigne l'architecture sur les capacités du terminal et le profil de l'utilisateur, ce qui permet de mettre en œuvre des négociations en vue d'obtenir un contenu accommodant les spécificités du terminal. *Le module d'adaptation et de négociation* (ANM) assure l'adaptation et la négociation du contenu. Cela est effectué grâce à l'application d'un ensemble de méthodes de transformation structurelle et d'adaptation de contenu. L'adaptation est dans la majorité des cas dynamique et dépend des valeurs que prennent les dimensions du contexte (l'application cliente utilisée, les formats acceptés, la taille d'écran, etc.). Ce module communique avec les autres entités de l'architecture afin de prendre la meilleure décision de négociation : choix de version de contenu, choix de méthode d'adaptation, choix de méthode de transmission du contenu, etc. Le *protocole de négociation* définit un modèle d'interaction entre le module UCM et le processus de négociation du module ANM. Le *système de gestion de profils* assure l'analyse et la gestion des descriptions du client, du contenu et des méthodes d'adaptation supportées, au profit de l'adaptation de la réponse du serveur.

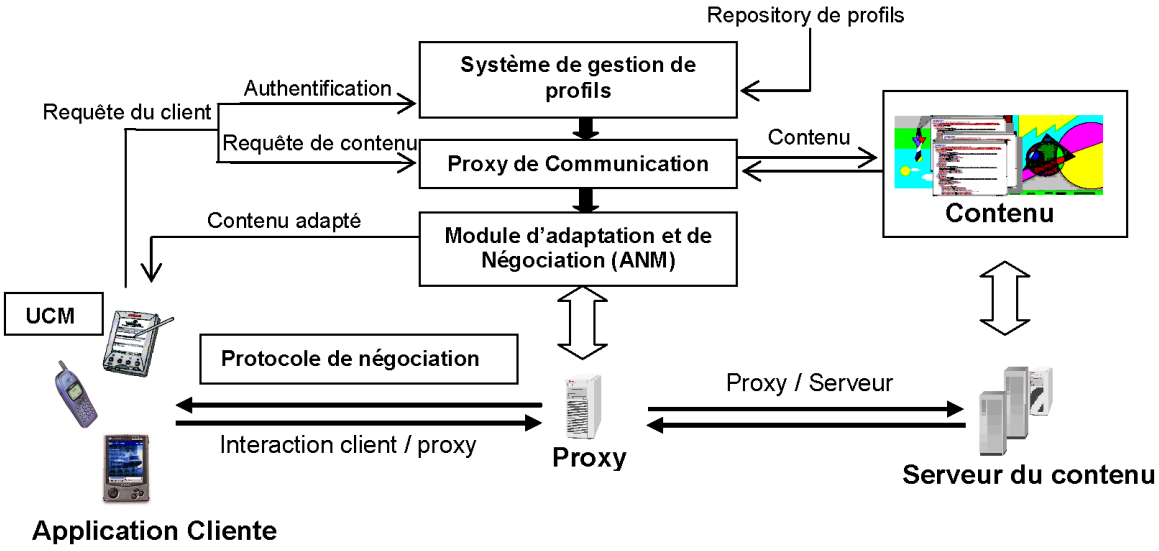


Figure 4.1 Architecture NAC d'après [Lemlouma, 2004].

L'objectif des mécanismes de négociation est de trouver un consensus entre l'utilisateur et le serveur concernant le contenu demandé. L'approche de négociation de l'architecture NAC évite que l'application de tels mécanismes soit de la responsabilité de l'application cliente, qui n'a qu'une vision partielle du contexte de diffusion. La stratégie de négociation de NAC s'appuie sur l'ensemble des déclarations de profils des différents éléments de l'environnement et sur les méthodes d'adaptation disponibles.

Parmi les techniques d'adaptation mises en œuvre par le module ANM nous notons :

- la transformation de contenu SMIL,
- le filtrage des documents SMIL,
- la substitution de variants SMIL,
- **la conversion de format image** (JPEG, WMP),
- **la compression des images** (JPEG),

- **la réduction de la taille d'une image**,
- **la réduction du nombre de couleurs**,
- **la réduction du débit des vidéo MPEG**,
- l'adaptation de protocole de communication (HTML, WAP).

L'intérêt de cette architecture est qu'elle présente une grande capacité d'adaptation aux différents types de documents multimédia. La stratégie de négociation de NAC peut être appliquée dans n'importe quel cadre incluant : un ensemble de méthodes d'adaptation qui peuvent transformer le contenu ; un processeur d'adaptation qui permet à tout moment d'interpréter les règles d'adaptation ; une description en termes de profil du contenu du serveur afin d'assurer une bonne qualité d'adaptation, la description des caractéristiques et des préférences relatives au client. De plus, les techniques d'adaptation que nous avons marquées en gras dans la liste ci-dessus, peuvent être directement utilisées dans une solution visant l'adaptation des textures des objets dans une scène 3D.

4.1.2. La plate-forme MPEG-21 Digital Item Adaptation

L'initiative MPEG-21 *Multimedia Framework* [Bormans *et al.*, 2003][Burnett *et al.*, 2003] propose des solutions pour l'utilisation de ressources multimédia à travers une large variété de réseaux et de dispositifs d'accès. Afin d'aboutir à cela, MPEG-21 s'attaque à l'uniformisation de la gestion de contenus, de la réutilisation de contenu dans de nouveaux contextes d'utilisation, de la protection des droits, de la protection de la vie privée des consommateurs et des fournisseurs de contenus multimédia, etc.

MPEG-21 est construit au-dessus de la famille des standards MPEG (MPEG-1, MPEG-2, MPEG-4, MPEG-7) en s'appuyant plus particulièrement sur MPEG-4 en ce qui concerne la partie diffusion de l'information et sur MPEG-7 pour la description sémantique des contenus.

MPEG-21 est organisé en plusieurs parties pouvant évoluer indépendamment. Chaque partie couvre un des aspects liés à la gestion et la diffusion d'information multimédia. À ce jour, il y a douze parties qui composent la plate-forme MPEG-21, parmi lesquelles on retient *:*

- *Digital Item Declaration (DID)*[Burnett *et al.*, 2005] – munit la plate-forme d'une notation homogène et flexible pour la déclaration de ressources média.
- *Digital Item Identification (DII)* [Burnett *et al.*, 2005] – permet d'identifier une entité au sein de la plate-forme indépendamment de sa nature, son type, son niveau de granularité.
- *Digital Item Adaptation (DIA)* [Vetro *et al.*, 2005] – définit des outils pour la description de l'environnement d'utilisation et des propriétés des ressources média qui peuvent influer sur le processus de diffusion (notamment les terminaux, les réseaux, les utilisateurs et l'environnement naturel que l'utilisateur et le dispositif d'accès partagent).
- *Digital Item Processing (DIP)* [De Keukelaere *et al.*, 2005] – définit des mécanismes assurant la standardisation et l'interopérabilité des processus d'exploitation des informations caractérisant les ressources média.

Les concepts de base de MPEG-21 mettent en relation les ressources média et le public auquel elles s'adressent (utilisateurs, créateurs, communautés, etc.). Les ressources média sont introduites à travers le concept de *Digital Item* (DI). Un DI peut être composé de plusieurs

ressources média : des images, des vidéo, des textes, des objets 3D, etc. Le DI est l'unité fondamentale pour la diffusion et les opérations au sein de la plate-forme MPEG-21.

MPEG-21 DIA [Vetro *et al.*, 2005] définit les outils pour l'adaptation des DI. Un des buts visé par MPEG-21 est de rendre le processus d'adaptation des DI transparent vis-à-vis des utilisateurs, des dispositifs d'accès et des canaux de communication. Comme indiqué dans l'architecture conceptuelle présentée dans la Figure 4.2, un DI peut être assujetti à plusieurs types d'adaptation :

- adaptation qui vise une ressource du DI,
- adaptation qui vise un descripteur spécifique du DI,
- adaptation qui vise le DI lui-même, ce qui corrspond à la génération d'un DI adapté.

Du point de vue de l'adaptation, il est nécessaire de disposer, en plus du contenu à adapter, des descriptions relatives à son format et à l'environnement de diffusion afin de trouver la meilleure solution de transformation.

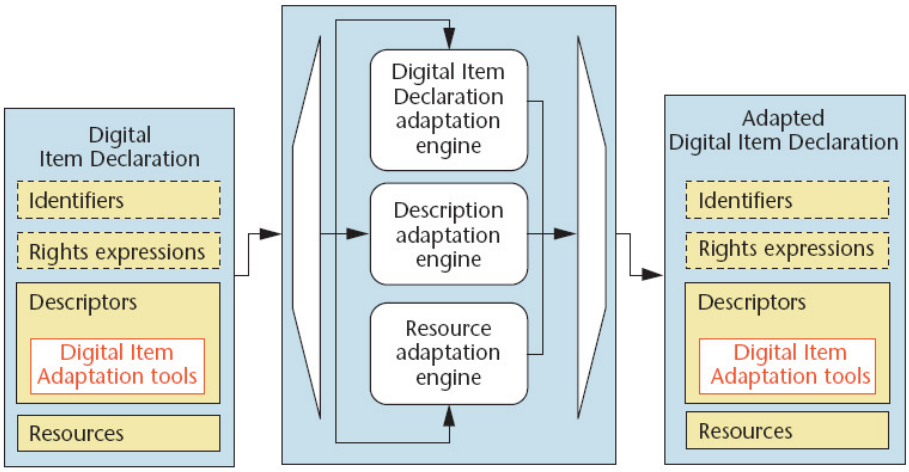


Figure 4.2 Architecture du processus d'adaptation d'un DI en MPEG-21 d'après [Burnett et al.*, 2005].*

DIA inclut également des concepts permettant de caractériser le processus et les différents acteurs intervenant dans la diffusion de l'information. L'environnement de diffusion est caractérisé par :

- les caractéristiques des dispositifs d'accès (*Terminal Capabilities* – caractéristiques matérielles : vitesse du processeur, mémoire disponible, taille de l'écran ; et logicielles : formats supportés, système d'opérations, etc.),
- les caractéristiques des canaux de communication (*Physical network condition* – bande passante, variations de la bande passante, etc.),
- les caractéristiques de diffusion (*Delivery Capabilities* – types de protocoles impliqués dans la diffusion MPEG-2, TCP/IP, RTP, etc.),
- les préférences utilisateur (*User preferences* – préférences de navigation, préférences de filtrage, âge, sexe, etc.),
- les caractéristiques de l'environnement physique dans lequel la transaction a lieu (*Natural environment characteristics* – localisation, environnement intérieur/extérieur, vitesse de déplacement, etc.) et

- les caractéristiques de service (*Service Capabilities* – rôle de l'utilisateur, type de service, négociation des droits de diffusion, etc.).

La plate-forme d'adaptation impose un codage unique de description des contenus d'un DI nommée *Bitstream Syntax Description* (BSD). BSD se présente comme un format XML qui décrit le flux associé à une ressource média. L'encodage XML de la description permet d'envisager facilement la mise en place des adaptations en utilisant des langages de transformation XML tels que XSLT [Clark, 1999] ou STX [Cimprich *et al.*, 2007]. Des illustrations dans ce sens sont disponibles dans [Panis *et al.*, 2003], [Kim *et al.*, 2006] et [Devillers *et al.*, 2005]. Dans la Figure 4.3 est illustrée l'architecture d'adaptation introduite par [Vetro *et al.*, 2005]. En filtrant ou en modifiant la description BSD associée à un DI lors de la diffusion, la construction du flux résultat se fait sur la base de la nouvelle description BSD. Afin de maîtriser la construction de descriptions BSD et de s'assurer de la validité des opérations subies par une description BSD, un langage nommée *Bitstream Syntax Description Language* (BSDL) est spécifié dans la partie DIA du MPEG-21. BSDL est construit au-dessus de *XML Schema* et définit les futures restrictions et/ou extensions qui serviront à décrire les différents flux multimédia.

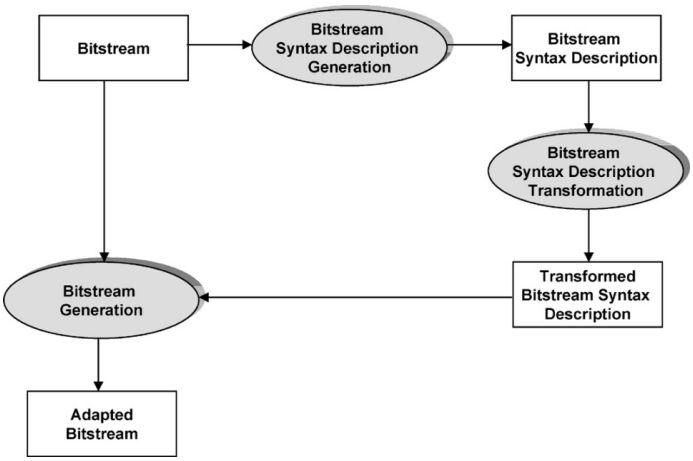


Figure 4.3 Architecture d'adaptation basée sur BSD d'après [Vetro et al.*, 2005].*

Après avoir passé en revue ces deux propositions auxquelles nous attachons une attention spéciale vis-à-vis de leur généricité, nous présentons dans la section suivante quelques notions spécifiques à l'adaptation portant sur les données 3D.

4.2. Aperçu des techniques d'adaptation 3D

Afin de dégager les spécificités des adaptations relatives aux données 3D, nous donnons un aperçu des techniques d'adaptation visant la modification des différentes dimensions de la scène (géométrie, apparence, …), ainsi que des dimensions du document sous-jacent (structure logique, type d'encodage, etc.).

Le processus d'adaptation peut affecter une ou plusieurs dimensions spécifiques d'une donnée 3D. Nous avons identifié plusieurs types de méthode d'adaptation : les méthodes qui s'attachent à la simplification de la géométrie de l'objet 3D, les méthodes qui portent sur la simplification de l'apparence de l'objet 3D, les méthodes qui s'intéressent à une reconsidération de la structure logique de la donnée afin d'optimiser la diffusion, et les méthodes qui visent un réaménagement spatial de la scène afin de mieux présenter l'information qu'elle contient. Généralement, les propositions rencontrées dans la communauté 3D concernent plusieurs

dimensions en même temps. Par exemple, les adaptations qui portent sur la géométrie de la donnée doivent s'intéresser également aux transformations que cela implique sur les textures des objets. Dans ce cas, nous prenons en compte la dimension visée en premier lieu par l'adaptation. Pour l'exemple discuté précédemment, nous identifions l'adaptation comme une adaptation de type géométrique car c'est la géométrie qui prime dans le processus d'adaptation même si d'autres dimensions sont altérées par la suite.

Les adaptations qui s'appliquent à une donnée, ou plus généralement à une scène 3D, sont mises en place pour satisfaire les contraintes imposées par le contexte de diffusion. L'accommodation de contraintes matérielles se traduit le plus souvent par une réduction de la complexité de la scène ou de la taille physique du fichier contenant la scène. Lorsque l'on réduit la complexité de la scène, nous parlons de dégradation de scène. La dégradation d'une scène peut s'effectuer à deux niveaux. Tout d'abord, au niveau des objets atomiques nous retenons : la simplification de la géométrie [Hoppe, 1996][Certain *et al.*, 1996][Xia *et al.*, 1997][Cohen *et al.*, 1997][To *et al.*, 1999] et de l'apparence [Lewis *et al.*, 1992][Barni *et al.*, 2001]. En ce qui concerne les objets composites on peut filtrer ou substituer les objets composant de moindre importance pour la bonne visualisation de la scène [Funkhouser *et al.*, 1993][Chittaro *et al.*, 2002][Dachselt *et al.*, 2006][Marvie *et al.*, 2004][Mulloni *et al.*, 2007]. L'importance d'un élément peut être évaluée par rapport au contexte de diffusion, à la sémantique de l'objet, à l'importance attachée à l'objet d'un point de vue utilisateur.

Les techniques visant la réduction de la taille de la scène en proposant des encodages plus efficaces (encodage binaire, compression de données, transformation de la représentation géométrique) sont orthogonales à celles discutées précédemment. Nous distinguons les méthodes assurant une compression sans perte de qualité (encodage binaire *XML Binary* [Goldman *et al.*, 2005] pour les documents X3D) et les méthodes qui n'assurent pas une reconstruction parfaite de la scène initiale. En effet, les méthodes de compression, autres que celles qui explorent uniquement la compression de la structure du document, supposent une dégradation ou une substitution des moyens de représentation de la géométrie (par exemple, l'utilisation de NURBS [Piegl, 1991] à la place de polygones [Versprille, 2005]), à l'instar des techniques de compressions d'images 2D (par exemple, à base d'ondelettes [Lewis *et al.*, 1992][Barni *et al.*, 2001] qui restent importantes pour la compression des textures des objets dans les scènes 3D.

Un autre point important à prendre en compte lorsque l'on caractérise les différentes techniques d'adaptation est le moment auquel elle intervient dans le processus de diffusions d'une scène 3D. Le processus de diffusion des scènes 3D se fait en deux phases :

a) la transmission de la scène (où des contraintes de bande passante doivent être respectées) et

b) la visualisation de la scène sur le dispositif d'accès (où des contraintes relatives à la configuration logicielle et matérielle sont à observer).

Les techniques de compression permettent de faciliter la transmission de la scène, cependant, le plus souvent, la visualisation ne peut pas démarrer avant que le document soit téléchargé en intégralité et décompressé par la suite. Afin d'accélérer la mise à disposition d'une visualisation même partielle d'une scène, des techniques d'encodage et de transmission progressive ont vu le jour [Yin *et al.*, 1997][Lafruit *et al.*, 2000][Hosseini *et al.*, 2001][Celakovski *et al.*, 2005][Cai *et al.*, 2007]. Dans ces propositions, la scène 3D est vue comme un média *scalable*[34] décrit par une couche de base, qui correspond à la représentation de base de la scène, enrichie par des couches supplémentaires qui rapprochent la scène de sa

[34] média qui peut être chargé progressivement

représentation initiale. L'intérêt de ces approches est qu'elles sont sensibles aux variations (de bande passante ou de capacités de traitement du dispositif d'accès) du contexte de diffusion pendant la diffusion même de la scène. En conséquence, en réduisant le nombre de couches d'information, ces propositions peuvent adapter le niveau de détail de la visualisation de manière automatique.

En plus de ces techniques qui affectent uniquement l'encodage ou le contenu de la scène 3D, il existe des techniques de *transmodage*[35] qui changent la modalité de représentation média des données. Ainsi, à la place de diffuser la scène en tant que documents 3D, certains auteurs [Di Giacomo *et al.*, 2004][Lamberti *et al.*, 2003] proposent la diffusion à distance d'une séquence d'images correspondant au rendu visuel de la scène en question. Ce type d'approche se prête bien aux situations où les dispositifs d'accès ne supportent pas les données 3D ou n'offrent que peu de possibilités d'interaction avec la scène. Ainsi, l'utilisateur perd son statut d'acteur dans la scène et se transforme en simple spectateur. [Lamberti *et al.*, 2003] essaient de répondre à cette problématique en proposant également des moyens d'interaction avec la scène à travers un client léger déployé sur le dispositif d'accès.

Dans la suite de ce chapitre, nous concentrons notre attention sur les méthodes d'adaptation visant uniquement des modifications directes d'une des dimensions (géométrie, apparence, structure logique) du document associé à une scène 3D. Nous faisons ce choix car dans cette thèse, nous nous intéressons plus précisément à la dimension document d'une scène 3D et de la manière dont les modifications affectant le document sous-jacent à la scène sont réalisées. Nous considérons les techniques d'encodage, de compression ou de transmodage comme étant orthogonales à la modification de la scène. Ces techniques qui visent principalement l'optimisation de la transmission des données 3D, peuvent s'appliquer en fin de processus d'adaptation de la scène afin d'accélérer la diffusion et l'accès à l'information.

Pour analyser les apports de chaque proposition relative à l'adaptation des objets et des scènes 3D nous proposons considérons les critères suivants :

- les types d'adaptation implémentés : adaptation de la géométrie, adaptation de l'apparence, modification de la structure logique, substitution,filtrage ;
- critères pour la mise en œuvre de l'adaptation : contraintes matérielles, préférences utilisateur ;
- paramètres du processus d'adaptation ;
- formalisation de l'approche d'adaptation.

4.3. Typologie d'adaptation 3D

4.3.1. Dégradation de la géométrie

Les propositions s'intéressant à la modification de la géométrie s'appuient sur des méthodes de simplification. Les transformations s'appliquent aux éléments géométriques simples (points, lignes, triangles) et elles ciblent en priorité les objets définis comme des maillages 3D. Ainsi, on rencontre des solutions visant à limiter le nombre de sommets, d'arêtes et/ou de triangles composant la surface. L'élimination d'un élément se fait de façon à nuire le moins possible la forme initiale de la surface. Lorsque l'on veut appliquer les techniques que nous présentons ci-dessous à une scène quelconque, une transformation préalable de celle-ci est

[35] changement de modalité (audio->texte, vidéo->image, 3D->vidéo, etc.)

nécessaire. Ainsi, toutes les primitives complexes (cubes, sphères, etc.) doivent être remplacées par des maillages équivalents. Cette transformation se traduit dans une augmentation importante de la taille du fichier encodant la scène. À la place d'une seule primitive définissant une sphère, on est amené à introduire un nombre important de triangles simulant une sphère. Cependant, cette augmentation de la taille est compensée par un meilleur contrôle du niveau de détail et une amélioration des performances lors du rendu visuel du nouveau maillage. Ainsi, la plupart des solutions s'attachant à l'adaptation géométrique ont été proposées dans le cadre de la transmission progressive des surfaces 3D [Hoppe, 1996][Certain *et al.*, 1996][Xia *et al.*, 1997][Cohen *et al.*, 1997][To *et al.*, 1999][Kim *et al.*, 2006].

[Hoppe, 1996] se trouve parmi les premiers auteurs à introduire le terme de *progressive mesh* (en français maillage progressif). Les *maillages progressifs* répondent à plusieurs problèmes inhérents à la gestion de données 3D : simplifier les surfaces, calculer les niveaux de détail, limiter le stockage, faciliter la diffusion, etc. Un maillage progressif est caractérisé par une version basique et un certain nombre d'informations qui peut être utilisé petit à petit afin de reconstruire la version complète de la surface. [Certain *et al.*, 1996] s'attaquent à la diffusion progressive et à la visualisation interactive en s'appuyant sur la caractérisation à plusieurs échelles des surfaces, en analysant séparément la forme et la couleur. [Xia *et al.*, 1997] proposent une solution pour une simplification appliquée de manière hétérogène à une surface. Sur une surface, il peut y avoir des régions qui sont mieux préservées par le processus de simplification que d'autres. Les auteurs se servent de cette approche afin d'optimiser les régions qui sont dans le champ de vision de l'utilisateur. Ainsi, les informations permettant de parfaire le niveau de détail de triangles se trouvant au centre du champ de vision, sont transmises en premier. Dans la suite, nous discutons plus en détails des travaux de [Cohen *et al.*, 1997], [To *et al.*, 1999] et [Kim *et al.*, 2006] afin de mieux illustrer les grandes lignes du processus de simplification.

Dans [Cohen *et al.*, 1997], un autre travail issu de la transmission progressive de surface, les auteurs étudient la simplification successive d'une surface 3D en observant plusieurs niveaux de détail. Les auteurs souhaitent que le passage entre deux niveaux de détail successifs soit déterminé afin de mieux contrôler le processus de dégradation de la surface. Chaque niveau de détail est déterminé par une fonction linéaire qui quantifie la déviation du nouveau niveau de détail de la surface par rapport au précédent et par rapport à la surface initiale. Ces fonctions sont également utilisées afin de calculer les modifications à apporter aux coordonnées de la texture pour conserver au mieux l'apparence de la surface. L'idée générale de l'algorithme proposé est de supprimer les arêtes dans l'ordre inverse de leur importance. L'importance, ou le coût de suppression d'une arête, est donnée par la courbure de la surface dans son voisinage. La sélection des arêtes à supprimer au passage à un niveau de détail inférieur, se fait à l'aide d'une fonction minimisant la différence entre la déviation souhaitée et les coûts de suppression des arêtes sélectionnées. La suppression d'une arête est réalisée en déplaçant ses extrémités dans un point qui devient un nouveau sommet de la surface. Le choix de ce point est capital dans le processus de simplification car la déviation introduite par un point mal choisi peut être plus importante que celle prévue. Afin de minimiser cette déviation, les auteurs utilisent une méthode qui projette tous les triangles voisins sur un plan déterminé afin qu'il n'y ait pas d'intersection entre les projections des triangles. Le choix du point se fait en 2D de telle façon que les arêtes des nouveaux triangles, induits par l'introduction du nouveau sommet, ne se chevauchent pas. À ce point dans le plan de projection, on associe une droite dans l'espace 3D initial. La position de ce point sur la droite est ensuite calculée afin de minimiser la distance qu'il peut y avoir entre le nouveau sommet et les anciens sommets de l'arête qui a été supprimée.

Dans [To *et al.*, 1999], les auteurs s'intéressent à la mise en œuvre d'un algorithme permettant la transformation d'un maillage, en maillage progressif, en veillant à limiter la

quantité d'informations supplémentaires permettant l'évolution du maillage basique (le moins détaillé) à sa version initiale (la plus détaillée). Le mécanisme mis en œuvre pour obtenir la surface basique tout en gardant les aspects visuels importants (creux, pics, …), comme toute simplification opérée pour tenir compte des conditions matérielles, correspond à un processus d'adaptation. Il s'appuie sur la simplification de la surface en enlevant les points de la surface dont l'importance est moindre. L'importance d'un point est définie par rapport à la courbure locale de la surface. Plus le voisinage du sommet est accidenté, plus l'importance du sommet est grande, car si le sommet est supprimé on perd beaucoup d'information liée au profil du terrain. Lorsqu'un sommet est sélectionné pour exclusion, les arêtes dont il est point de départ sont considérées afin de trouver l'arête dont l'importance est moindre. L'importance d'une arête est fonction de l'importance des sommets la composant ainsi que de sa longueur. L'extrémité de moindre importance est alors absorbée par l'autre extrémité et l'arête est supprimée de la surface. Les arêtes incidentes au sommet absorbé se cumulent avec les arêtes incidentes au sommet absorbant. Le processus est illustré dans la Figure 4.4. Le graphe (sommet et arêtes) correspondant à la structure interne de la surface initiale est présenté en haut à gauche. Initialement, le nombre de sommets absorbés (présenté au centre du chaque sommet) est zéro. L'élimination du sommet v_4 et son absorption par le sommet v_6, ainsi que l'ajout des arêtes incidentes (v_6,v_1), (v_6,v_5), (v_6,v_3), (v_6,v_7), (v_6,v_2) sont illustrées par la mise à jour des étiquettes du sommet v6 et des arêtes en question. Avant de continuer le processus d'élimination, il faut recalculer l'importance des sommets affectés par les opérations effectuées. Le processus continue jusqu'à l'exclusion de tout sommet dont le niveau d'importance est inférieur à un seuil donné. Cette méthode est moins complexe que celle présentée précédemment [Cohen *et al.*, 1997] car elle implique moins d'opérations géométriques et d'optimisation. Néanmoins, elle ne présente pas les avantages des précédentes car cette méthode ne permet pas de générer des niveaux de détail de manière automatique et de dégrader de manière contrôlée la surface.

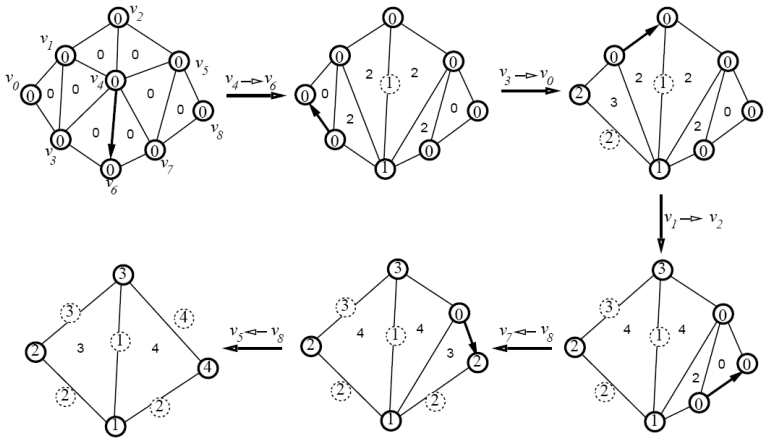


Figure 4.4 Simplification par élimination de sommets et d'arêtes d'après [To et al.*, 1999].*

[Kim *et al.*, 2006] proposent une méthode d'adaptation générique pour les maillages 3D en utilisant les outils mis à disposition par la plate-forme MPEG-21que nous avons présentée dans une section précédente. [Kim *et al.*, 2006] reprochent aux méthodes existantes un manque de maîtrise du processus d'adaptation. Les auteurs proposent un cadre théorique qui permet de mesurer le niveau d'adaptation requis en fonction des contraintes matérielles, de la complexité de la surface et d'autres préférences utilisateur. Chaque surface est associée à un item digital (DI) et est encodée grâce aux descriptions gBSD. Ces descriptions, faites en XML, sont transformées et filtrées grâce aux feuilles de transformation XSLT afin de correspondre au contexte de diffusion (utilisateur, dispositif d'accès). Le contexte lui aussi est représenté en

utilisant une autre brique de la technologie MPEG-21 XDI (*Context digital item*). L'encodage du contenu au sein du flux gBSD est structuré en plusieurs couches de contenu : une couche de base et des couches additionnelles. La superposition de toutes les couches correspond à la version originale de la surface. Les auteurs utilisent une méthode de simplification pour la construction de ces différents niveaux. Le choix de la méthode est indépendant de l'architecture générale du système. Une algèbre d'ensembles portant sur les sommets et les arêtes permet de construire les couches du contenu de manière indépendante. En ce qui concerne le choix de la dernière couche de contenu à envoyer lors de la diffusion, les auteurs proposent une méthode déterministe qui s'appuie sur des estimations de performance et de qualité. Ainsi, ils ont mesuré le nombre d'images visualisées par seconde lorsque tout le modèle est chargé sur le poste client, les capacités de calcul du dispositif en mode normal de fonctionnement et les capacités de calcul du dispositif au moment de la diffusion. À partir de ces données, suivant le nombre d'images par seconde souhaitées, un ratio d'adaptation à appliquer au modèle peut être calculé. Si ce taux est inférieur à un, une adaptation est mise en place.

Les méthodes présentées ci-dessus ne peuvent pas s'appliquer directement aux objets ayant une structure complexe (des bâtiments, des voitures, etc.) ou ceux construits à l'aide des primitives géométriques. Une étape de transformation de tels objets en surfaces simples est nécessaire. Nous considérons que cette mise à plat peut se révéler assez néfaste car lorsque l'on supprime des arêtes et sommets appartenant à des parties différentes de l'objet, on risque de perdre la cohérence de la forme. Par exemple, on ne devrait pas fusionner des sommets situés dans des parties distinctes de l'objet (un sommet de la roue et un sommet de la carrosserie). Ce problème ne se pose pas, dès lors que les parties connexes d'un objet forment des angles relativement grands. Ainsi puisque l'importance des sommets situés sur les lignes de séparation est grande, le processus de simplification ne traite pas ces sommets. Cependant, garder l'information concernant la fragmentation initiale de l'objet est probablement nécessaire afin d'éviter ce genre de soucis dans un cadre générique.

4.3.2. Dégradation de l'apparence

Les techniques d'adaptation visant l'adaptation de l'apparence des objets 3D se rapprochent fortement de celles proposées dans le domaine des images car, pour la plupart, l'apparence est définie au moyen de simples images ou bien de vidéo. Des problèmes plus complexes peuvent se poser concernant les textures des surfaces géométriques définies à partir d'un ensemble de points ou des triangles, cas dans lesquelles des techniques similaires à celles décrites dans la section précédente sont envisageables. Dans la suite, nous présentons brièvement quelques travaux visant l'adaptation d'images, issus du domaine multimédia [Williams, 1983][Fox *et al.*, 1998][Smith *et al.*, 1998][Lee *et al.*, 2001][Chen *et al.*, 2003] ou spécifique au domaine 3D [Inatsuka *et al.*, 2005].

[Williams, 1983] propose une technique qui s'appuie sur une décomposition pyramidale de la texture afin de contrôler le niveau de compression de la texture tout en garantissant une dégradation qui nuit le moins possible la qualité du processus de visualisation. Pour une texture, Williams propose la construction d'une structure qui est construite en utilisant les composantes (R, G, B) des couleurs de la texture initiale dans chaque point (voir le carré *N0* de la Figure 4.5). Le niveau suivant (*N1*) est obtenu en conservant une valeur sur quatre pour chaque région, et ainsi de suite.

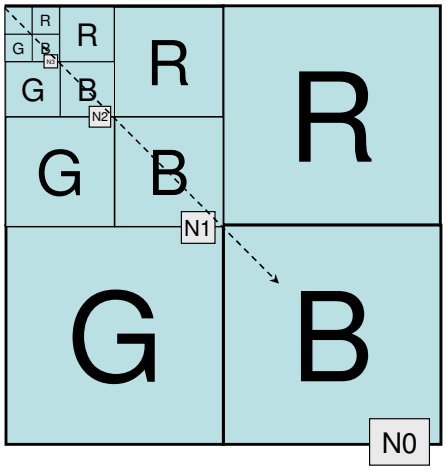


Figure 4.5 Structure pyramidale pour la mise en œuvre des techniques de mip-mapping [Williams, 1983].

Cette structuration de la texture à plusieurs niveaux permet d'envisager la mise en place de techniques d'interpolation à inter ou intra niveaux. Au-dessus de ces types d'interpolation, Williams met en place des méthodes de compression de textures de type *mip-mapping* ('multum in parvo' – beaucoup d'objets dans un endroit restreint). Les techniques de *mip-mapping* enrichissent l'interpolation bilinéaire classique qui sert à évaluer les valeurs de pixels, en considérant l'interpolation entre différentes versions (N3, N2, N1, N0) de la texture initiale. Ainsi il est possible d'avoir des régions d'images compressées de manière plus ou moins importante suivant l'importance de la région. Cette technique a été adoptée avec succès dans de nombreuses applications visant la compression ou la construction de niveau de détail d'une texture. Nous retenons [Abasolo *et al.*, 2006] et [Bordersen, 2005] qui s'intéressent à la compression de textures associées aux modèles de terrains de grandes tailles.

[Fox *et al.*, 1998] proposent une architecture à base de proxy pour l'application des adaptations spécifiques aux différents types de contenus. En particulier, les auteurs considèrent la compression d'images afin de réduire le temps de transmission. Des paramètres, tels que la taille du fichier, le nombre de couleurs, et le format, influent sur le niveau de compression appliqué à l'image initiale.

[Smith *et al.*, 1998] présentent un système de transcodage d'images intégré dans des pages Web. Le système repose sur la classification de contenu de l'image, en considérant le type d'image et l'intention de l'auteur. La classification est réalisée de manière automatique en explorant le texte entourant l'image et le contexte dans lequel le document Web est placé. L'analyse de cette classification guide les choix de transcodage du système. Le système choisit entre plusieurs fonctions, celle qui modifie les images en affectant la taille, la qualité de l'encodage, les niveaux de couleur ou qui remplace l'image avec le texte (transmodage) qui la caractérise.

[Lee *et al.*, 2001] explorent la mise en place d'adaptation au-dessus de la notion de région d'intérêt [Christopoulos *et al.*, 2000]. Une région d'intérêt correspond à une zone de l'image qui présente de l'information importante par rapport au sens donné à l'image. Ainsi, au lieu de traiter l'image comme un tout, les régions d'intérêts jouissent d'un traitement spécial qui les préserve des transformations (compression) appliquées aux régions moins importantes de l'image.

[Chen *et al.*, 2003] proposent une technique d'adaptation qui modifie le contenu de l'image afin que celle-ci s'accommode le mieux possible à la taille de l'écran du dispositif d'accès. Ils s'appuient pour cela, sur une modélisation de l'attention de l'utilisateur en introduisant la notion d'*objet attentionnel*. Un *objet attentionnel* correspond à une région de

l'image dont le contenu est censé attirer l'attention de l'utilisateur (par exemple, une légende sur une image). Le plus souvent, un objet attentionnel correspond à un objet sémantique tel qu'un visage, une fleur, une voiture, une proposition, etc. Trois attributs sont associés à un objet attentionnel : la localisation de l'objet dans l'image à l'aide d'une région d'intérêt (le plus souvent par un rectangle ou une sphère), la valeur attentionnelle de l'objet (en relation avec la quantité d'information apportée), et la dimension minimale de perception (au-delà de laquelle l'objet ne peut être perçu correctement). Ces informations sont ensuite exploitées afin d'identifier les régions de l'image à préserver. Les auteurs s'appuient également sur des techniques de découpage qui réduisent la taille d'une image en retenant uniquement les parties intéressantes de celle-ci.

Cette sous-section consacrée aux adaptations de l'apparence se termine par l'étude d'une proposition issue du domaine géographique. Au sein de tout système géospatial 3D, l'apparence complète la description géométrique du relief avec des indices visuels qui aide à synthétiser l'information que le système veut diffuser (plan d'occupation des sols, formes de reliefs, altitudes, etc.). Ainsi, les textures dans un tel système occupent un rôle primordial et souvent, elles dépassent, en taille et en complexité, les données géométriques. Des techniques de découpage de textures en plusieurs régions (*tiles*) permettent de faciliter la diffusion des textures de grande échelle. Cependant, lorsque les contraintes matérielles l'exigent, il est nécessaire d'appliquer des opérations de simplification de chaque région de la texture.

[Inatsuka *et al.*, 2005] proposent une méthode de dégradation d'une texture représentant une carte 3D d'une zone urbaine. La dégradation se fait en proposant plusieurs niveaux de détail de la résolution de la texture. Les auteurs contrôlent a priori, à la diffusion, la génération des niveaux de détail en fonction de la taille de dispositif d'affichage et de l'importance de la texture. Ils font ce choix de construction a priori, en réduisant les temps de traitement pendant la diffusion de la texture. Le processus suppose la définition d'un certain nombre de points de vue (*viewpoints*) représentatifs au sein de la scène. C'est par rapport à ces *viewpoints* et à la complexité de la texture (contours, densités de couleurs) que l'importance d'une texture est mesurée. Plus la texture est éloignée du point de vue, moins elle est importante et plus l'on peut lui associer un niveau de détail moindre. Plus la texture présente des contours nets et une grande variété de couleurs, plus les auteurs la considèrent importante et lui associent des niveaux de détail fins. Ces niveaux de détail concernant les textures sont associés aux *viewpoints* depuis lesquels les textures en question sont visibles. Ainsi, lors de la visualisation on ne charge en mémoire que le niveau de détail spécifique de la texture pour chaque *viewpoint*.

4.3.3. Modification de la structure logique de la scène

Dans cette section, nous présentons quelques travaux [Brooks *et al.*, 1986][Airey *et al.*, 1990][Funkhouser *et al.*, 2003][Marvie *et al.*, 2004][Mulloni *et al.*, 2007] qui s'attachent à la modification de la structure logique prédéfinie d'une scène 3D afin de faciliter sa visualisation et d'augmenter la vitesse de rendu en temps réels en réduisant la charge en mémoire et le temps de traitement d'une scène. Ces méthodes s'apparentent à celles présentées dans la section consacrée aux adaptations géométriques et notamment à la transmission progressive des maillages. La ressemblance entre les deux approches réside dans la transmission progressive des éléments de la scène au poste client. Cependant, les éléments ne sont pas dégradés comme dans le cas des maillages progressifs mais présentés dans leurs états initiaux respectifs. Ces méthodes sont propres aux scènes représentant des environnements similaires aux espaces clos, tels que les intérieurs des bâtiments. Ces méthodes viennent compléter les solutions apportées notamment par la transmission progressive des maillages dans les scènes représentants des environnements ouverts similaires aux larges espaces, tels qu'un modèle de terrain ou une ville virtuelle.

Nous observons dans quelle mesure les différentes solutions sont implémentées au niveau du document décrivant la scène 3D, ou bien si elles sont spécifiques à un certain moteur de visualisation. Les solutions implémentées au niveau document présentent plus d'intérêt car elles sont reproductibles indépendamment d'une solution logicielle spécifique. Le comportement est donc reproductible sur tous les navigateurs supportant le standard de description choisi.

[Funkhouser *et al.*, 1992]

Les premiers travaux importants portant sur la navigation efficace au sein de scènes de taille conséquente ont été entrepris par [Brooks *et al.*, 1986] et [Airey *et al.*, 1990]. Ensuite, dans les années 90, à l'Université de Californie de Berkeley, les travaux initiés par [Teller, 1992], [Funkhouser *et al.*, 1992] et [Funkhouser *et al.*, 1993] se concrétisent par un prototype qui permet la navigation, à des taux de rafraîchissement élevés de l'image (*framerate*), au sein d'une scène représentant de façon très détaillée un bâtiment (Soda Hall) du campus de Berkeley. La scène comporte plus de dix millions de polygones couvrant les éléments architecturaux du bâtiment ainsi que le mobilier et les autres éléments du bâtiment (fenêtres, portes, lumières, etc.).

Le prototype de navigation jouit d'un système de gestion de la mémoire efficace et d'un moteur de sélection et de mise à jour de la liste des objets visibles lorsqu'un utilisateur se déplace à l'intérieur du bâtiment. Ces deux composants logiciels reposent sur :

a) une organisation de données au sein d'une base de données qui décrit d'une manière hiérarchique le bâtiment avec ses éléments comportant plusieurs niveaux de détail et

b) une division de l'espace en cellules qui facilite le calcul de la liste des objets visibles en s'appuyant sur une analyse de visibilité à deux niveaux cellule-cellule, cellule-objet.

La division spatiale se fait par rapport aux plans opaques de la scène (murs, portes, plafonds) et par soucis d'efficacité, les cellules sont organisées dans une structure de type *k-D tree* [Bentley, 1975]. L'analyse de visibilité cellule-cellule repose sur l'identification de portails (zone de visibilité vers l'extérieur) pour chaque cellule. Un graphe d'adjacence qui rend compte des relations de visibilité entre cellules est construit. Les nœuds représentent les cellules et des arêtes sont créées entre les nœuds qui sont réciproquement visibles (voir la partie gauche de la Figure 4.6).

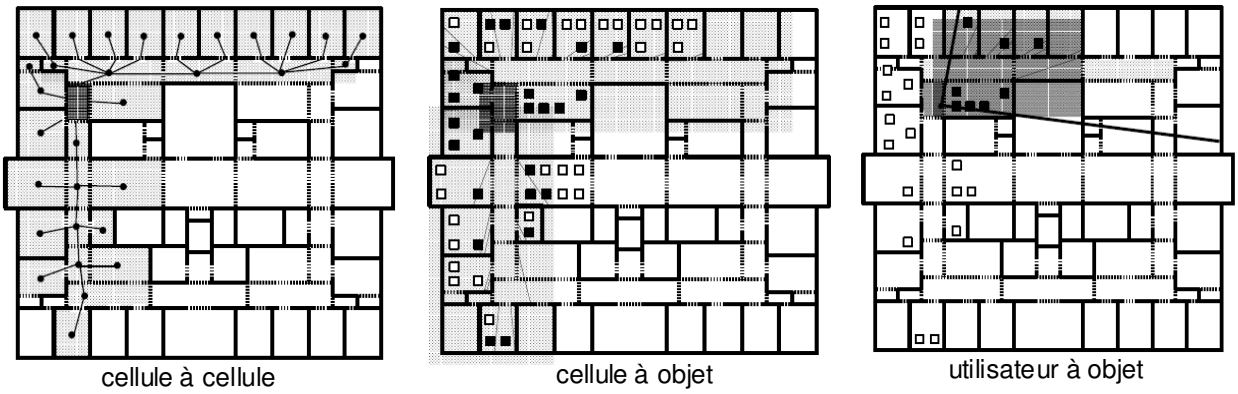


Figure 4.6 Vue partielle sur les différentes relations de visibilité d'après [Funkhouser et al.*, 1992].*

Cette structure sert à optimiser le calcul de relations de visibilité cellule-objet (voir la partie centrale de la Figure 4.6 ; les objets visibles sont dessinés en noir). Seuls les objets se trouvant dans les cellules adjacentes dans le graphe de visibilité cellule-cellule sont analysés. Les relations cellule-cellule ou cellule-objet sont calculées une seule fois lors du pré-traitement de la

scène. Le dernier type de relation de visibilité utilisateur-objet (voir la partie droite de la Figure 4.6 ; les objets visibles sont dessinés en noir), est évalué à chaque fois que l'utilisateur change sa position ou son orientation au sein de la scène. Les objets se trouvant dans le champ de vision de l'utilisateur sont choisis parmi ceux qui sont visibles depuis la cellule où se trouve l'utilisateur. Les résultats obtenus avec cette technique ont été très encourageants, cependant la proposition est restée très liée au prototype construit. Beaucoup d'informations supplémentaires sont stockées dans une base de données externe afin de sauvegarder les relations de visibilité pré-calculées. De même, deux modules logiciels (gestion de la mémoire, sélection des objets sur la base de relations de visibilité) sont étroitement liés à l'organisation particulière de ces données. Cependant, le point important de cette proposition est l'idée que lorsque l'on est en présence de grands volumes de données 3D il est nécessaire, en vue d'une diffusion et d'une visualisation efficace, de s'intéresser à un découpage de la scène qui privilégie le calcul des relations de visibilité.

D'autres propositions plus récentes, telles que [Marvie *et al.*, 2004][Mulloni *et al.*, 2007], s'intéressent à ce problème en considérant des solutions qui essaient d'inclure une plus grande partie d'informations additionnelles relatives à la visibilité au niveau du document 3D. [Cohen-Or *et al.*, 2003] recueillent les plus importantes propositions antérieures à 2002.

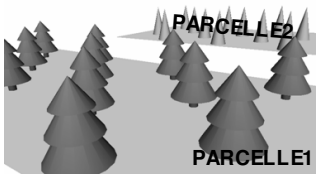
Les niveaux de détail dans X3D [Web3D, 2004]

Ce problème concernant l'optimisation de la vitesse de rendu, inhérent à la visualisation de scènes 3D de tailles conséquentes, a été reconnu en tant que préoccupation fondamentale de la communauté 3D. Dans le standard X3D [Web3D, 2004], des outils basiques permettant d'optimiser le rendu visuel sont proposés. Ainsi, on note la prise en compte des *niveaux de détail* à travers les éléments `LOD`. Les *niveaux de détail* visent l'optimisation du rendu visuel de la scène en laissant la possibilité aux utilisateurs de proposer plusieurs représentations en rapport avec la distance à laquelle l'utilisateur se trouve. Le fait que, lorsque l'on regarde de loin un objet on ne perçoit pas tous les détails de sa géométrie, est à la base de cette technique d'optimisation. Ainsi, à la place de demander le rendu d'une représentation complexe d'un objet dont on ne voit pas les détails, le langage X3D donne la possibilité aux concepteurs de proposer des représentations alternatives moins détaillées. Les niveaux de détail sont spécifiés du plus détaillé (L_0) ou moins détaillé (L_N). Le critère de passage d'un niveau à l'autre est défini comme une fonction de distance. Le concepteur doit indiquer N-1 distances de sauts (d_1, …, d_{N-1}). Le chargement en mémoire du niveau de détail correspondant à une certaine distance, se fait de manière transparente pour l'utilisateur. Le client logiciel assurant le rendu détermine la classe dans laquelle la distance effective se positionne par rapport aux distances de sauts.

Afin d'illustrer ces propos, nous présentons un exemple basique dans la Figure 4.7. La scène représente deux parcelles contenant plusieurs rangées d'arbres. À chaque arbre nous avons associé une représentation à deux niveaux (voir lignes 8-16 pour les arbres de la `Parcelle1`, respectivement lignes 25-33 pour les arbres de la `Parcelle2`). Le premier niveau correspond à une modélisation externe de l'arbre introduite à travers un nœud `Inline` (voir ligne 9, respectivement ligne 26). Cette représentation suppose l'utilisation de plusieurs primitives (cônes et cylindres). Une représentation moins détaillée, à base d'un seul cône, est introduite comme un deuxième niveau de détail (voir lignes 10-15, respectivement lignes 27-32). Comme il y a seulement deux niveaux de détail proposés, le concepteur indique uniquement une distance de saut (voir l'attribut `range` de la ligne 8, respectivement 25). Dans la partie gauche de la figure, nous illustrons la situation où la distance entre le point de vue sur la scène et les arbres de la `Parcelle1` est inférieure à la distance de saut, et supérieure à la distance de saut pour les arbres de la `Parcelle2`. Ainsi, la `Parcelle1` est peuplée d'arbres en représentation détaillée, tandis que

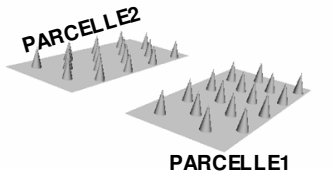
les arbres de la `Parcelle2` sont présentés sous leur forme basique. Dans la partie droite, la distance est supérieure aux distances de sauts, ainsi les arbres des deux parcelles sont présentés en utilisant le niveau de détail moindre. Nous préconisons l'utilisation des nœuds `Inline` qui correspondent à l'importation des scènes externes afin de garder la taille du fichier comportant les `LODs` raisonnables. Si on inclut systématiquement de manière explicite l'ensemble des représentations au sein du document, la taille de celui-ci risque d'exploser et ainsi nuire à la visualisation de la scène.

```
01:<X3D version="3.0" profile="Immersive">
02:  <Scene>
03:    <Background skyColor="0 0.8 1"/>
04:    …
05:  <Group DEF='PARCELLE1'>
06:  …
07:    <Group DEF='RANGEE1'>
08:      <LOD DEF="ARBRE" range='5'>
09:        <Inline url="arbre.x3d"/>
10:        <Transform translation="0 0.40 0">
11:          <Shape>
12:            <Cylinder radius="0.23" height="1"/>
13:            …
14:          </Shape>
15:        </Transform>
16:      </LOD>
17:      …
18:    </Group>
19:    …
20:  </Group>
```



Dist(Parcelle1,VP) < Range
Range < Dist(Parcelle2,VP)<+∞

```
21:  <Transform translation="0 0 7">
22:    <Group DEF="PARCELLE2">
23:      …
24:      <Group DEF='RANGEE2'>
25:        <LOD DEF="ARBRE2" range='5'>
26:          <Inline url="arbre.x3d"/>
27:          <Transform translation="0 0.40 0">
28:            <Shape>
29:              <Cylinder radius="0.23" height="1"/>
30:              …
31:            </Shape>
32:          </Transform>
33:        </LOD>
34:        …
35:      </Group>
36:      …
37:      </Group>
38:    </Transform>
39:  </Scene>
40:</X3D>
```



Range < Dist(Parcelle1,VP) <+∞
Range < Dist(Parcelle2,VP) <+∞

Figure 4.7 Exemple d'utilisation des niveaux de détail dans les scènes X3D

Le langage X3D offre uniquement le support pour la mise en œuvre de telles optimisations, cependant elles doivent être définies de manière explicite par le concepteur de la scène [Visintini *et al.*, 2007]. Une piste à explorer peut être d'associer ce mécanisme avec les propositions de dégradation de la géométrie afin de définir des niveaux de détail de manière automatique pour tout objet dont la représentation est complexe.

[Marvie *et al.*, 2004]

Dans [Marvie *et al.*, 2004] les auteurs décrivent une extension VRML-X3D qui permet d'inclure des relations de visibilité au sein d'une scène 3D. Ces informations trouvent toute leur importance dans le contexte de la transmission progressive et de la visualisation en temps réel de grandes scènes 3D. L'extension supporte la description des relations de visibilité cellule-cellule, cellule-object et hybrides. Les relations de visibilité hybrides combinent les deux types de relations précédentes afin d'éviter la redondance lorsqu'un ensemble important d'objets dans une sous-région de cellule est visible depuis des cellules voisines.

Les auteurs construisent un prototype VRML/X3D nommé *ConvexCell* pour représenter le concept de cellule convexe. Les auteurs ne s'intéressent qu'aux cellules ayant une enveloppe convexe afin de bénéficier d'avantage de certaines propriétés géométriques permettant d'optimiser les calculs de visibilité et la navigation. En plus des informations définissant la géométrie (coordonnées, facettes, etc.) et les identifiants (ID, URL), le prototype contient une liste d'objets se trouvant à l'intérieur de la cellule, une liste de cellules adjacentes, une liste de cellules potentiellement visibles depuis la cellule actuelle (relations cellule-cellule), une liste d'objets potentiellement visibles depuis la cellule actuelle (relations cellule-objet). Les relations de visibilité hybrides sont stockées dans la liste des cellules adjacentes et la liste des objets

potentiellement visibles. La matérialisation de ces relations au sein du document 3D est, à notre avis, une importante avancée car c'est un premier pas vers une large adoption des techniques d'optimisation et de navigation dans le monde 3D. D'un autre côté, la construction de ces cellules devrait être rendue automatique et transparente pour les utilisateurs afin de ne pas rendre encore plus compliquée la création de scènes 3D. Une politique de description indépendante du point de vue documentaire qui se traduirait par la construction d'un nouveau fichier VRML/X3D contenant uniquement cette organisation en cellules, nous semble un bon compromis.

[Mulloni *et al.*, 2007]

[Mulloni *et al.*, 2007] s'intéressent à l'optimisation des processus de visualisation et de navigation au sein de scènes 3D sur dispositifs mobiles. Les auteurs ciblent les scènes modélisant des environnements clos similaires aux intérieurs de bâtiments et notamment les applications de type guide touristique de musées. L'approche est légèrement différente des précédentes car les auteurs ne matérialisent pas explicitement les relations de visibilité. Cependant, les auteurs proposent une description topologique de la scène qui est ensuite utilisée dans le processus de sélection d'objets potentiellement visibles. Cette description externe aux documents X3D matérialisant la scène, consiste en une liste de cellules et de portails les reliant, similaire au graphe d'adjacence proposé par [Funkhouser *et al.*, 1993]. Chaque cellule est représentée par un fichier X3D distinct. Les auteurs préconisent cette solution car l'alternative de maintenir toute la scène dans un seul fichier, enrichie d'annotations sémantiques délimitant les cellules et décrivant les portails, ne semble pas bien adaptée pour les dispositifs de type PDA. Dans la description topologique, les cellules sont identifiées par leur URL. On retient également les coordonnées de chaque frontière de la cellule afin de pouvoir, au moment du rendu, connaître la cellule dans laquelle se trouve l'utilisateur. En plus de l'évaluation des relations de type cellule-cellule, cellule-objet, les auteurs incluent un critère de distance pour augmenter les performances de visualisation. Ils partent de la supposition que, sur des dispositifs mobiles, la taille plus petite de l'écran fait que des objets distants deviennent peu visibles dans la scène et que, de plus, il est plus probable que l'utilisateur s'intéresse aux objets les plus près.

En synthèse, nous pouvons remarquer que la caractéristique générale de ces systèmes est le souci de dessiner le moins de polygones possibles lors de la visualisation d'une scène afin d'utiliser le moins possible le processeur et la carte graphique du système. La sélection de ces polygones (plus généralement des triangles) joue un rôle central dans l'efficacité des méthodes d'optimisation de rendu. Dans la plupart des méthodes, seuls divers critères de visibilité sont employés, sans prêter beaucoup d'attention à l'information sémantique caractérisant les différentes parties de la scène.

Dans la section suivante, nous discutons quelques propositions s'intéressant à l'adaptation d'une scène 3D en considérant la génération dynamique du contenu.

4.3.4. Génération adaptée du contenu

Les travaux qui s'inscrivent dans cette catégorie [Chittaro *et al.*, 2002][Chittaro *et al.*, 2004][Estalayo *et al.*, 2004][Dachselt *et al.*, 2006], considèrent généralement l'ensemble des caractéristiques des objets 3D (géométrie, apparence, organisation spatiale) afin de trouver le meilleur compromis entre les capacités matérielles et les besoins de l'utilisateur. Ces approches présentent une certaine similitude avec les adaptations multimédia mises en œuvre dans les documents 2D. Dans ce cas de figure, les objets de la scène sont bien identifiés et considérés comme des unités de présentation à part entière (comme les objets 2D sur une page Web) et non

plus comme un ensemble hétérogène de points, arêtes ou polygones 3D. Deux grandes familles d'approches se dégagent dans les communautés 2D/3D.

La première consiste en la description exhaustive de l'ensemble de situations envisageables a priori. Des standards, tels que SMIL [Bulterman, 2005] pour le 2D ou X3D [Web3D, 2004] pour le 3D, contiennent un élément appelé `Switch` permettant de décrire les alternatives. L'intérêt de ce premier type d'approche est que, tant que l'on est en présence d'une situation prévue, on obtient un rendu très soigné car décrit à l'avance par le concepteur de la scène. En contrepartie, ce type d'approche demande la création d'autant de variants que de contextes à prendre en compte et par conséquent montre des limites lorsque l'on rencontre une situation qui n'a pas été prévue a priori. Nous considérons ce type d'approche comme une approche de génération *pseudo* dynamique car, pour un contexte donné, les choix d'adaptation sont figés dès la conception de la scène. De plus, nous pensons que ce type d'approche peut s'appliquer avec succès aux applications qui mettent en œuvre une solution locale où le contexte de diffusion varie peu et de manière prévisible. Lorsque l'on vise des applications de type Web, des solutions plus génériques doivent être mises en place.

Le deuxième type de solution modifie dynamiquement une scène afin de faire respecter les contraintes imposées par le contexte de diffusion. Le choix des éléments à modifier n'est pas indiqué à l'avance par le concepteur de la scène. Les informations sur l'organisation de la scène et sur les objets contenus dans cette même scène, guident le processus d'adaptation dans les choix d'actions à mettre en œuvre. Si l'évolution des préférences et des intérêts des utilisateurs par rapport à une scène est prise en compte au niveau du système mettant en œuvre les adaptations dynamique, on parle de systèmes adaptatifs [De Bra *et al.*, 2003].

Dans la suite de cette sous-section nous présentons des travaux issus des deux catégories d'approches, en insistant sur les formalismes permettant de décrire les adaptations à mettre en place.

Adaptative Web 3D [Chittaro *et al.*, 2002]

Dans [Chittaro *et al.*, 2002], les auteurs abordent la conception d'une architecture générique pour la génération adaptée de scènes 3D nommée *Adaptative Web 3D* (AWE3D). L'architecture a aussi une dimension adaptative car elle suit les actions des utilisateurs afin de modifier les paramètres des adaptations mises en place. Les auteurs s'intéressent plus particulièrement au standard VRML, cependant leur proposition peut s'appliquer à toute approche de modélisation 3D dès lors que les deux conditions suivantes sont remplies : a) il est possible de suivre les faits et gestes de l'utilisateur au sein de l'espace 3D et b) la génération de contenu peut se faire à partir de prototypes paramétrables. Ces deux conditions sont respectées au sein de la famille VRML/X3D, car il est possible de tracer l'utilisateur, grâce à une large palette de capteurs (`TouchSensor`, `VisibilitySensor`, …) et il est possible de définir et instancier des prototypes (`PROTO`).

Afin de maîtriser la complexité d'une telle approche, les auteurs identifient trois grandes tâches :

- la tâche d'*acquisition* des informations relatives à l'utilisateur et à l'usage que l'utilisateur fait de la scène,
- la tâche de *représentation* des informations caractérisant l'utilisateur et les règles d'adaptation,
- la tâche de *production* du contenu adapté.

La tâche d'acquisition est celle qui permet de suivre les actions des utilisateurs et qui apporte ainsi la dimension adaptative au système. Cette tâche est rendue possible grâce aux capteurs qui peuvent être éparpillés dans la scène. Chaque objet instancié dans la scène est muni d'un capteur de contact (`TouchSensor`), d'un capteur de visibilité (`VisibilitySensor`) et d'un capteur de proximité (`ProximitySensor`). Ce qui nous intéresse plus particulièrement dans cette proposition est la formalisation des règles décrivant les adaptations et le processus de génération qui sont couverts par les deux dernières tâches.

L'architecture générique proposée est illustrée dans la Figure 4.8. La tâche de représentation est principalement assurée par le module de *Personalization*. Ce module est décomposé en deux sous-modules. Le sous-module *User Module Update Rules* s'intéresse à l'inférence de nouvelles règles et connaissances relatives à l'utilisateur à partir des informations stockées dans le modèle utilisateur (*User Model Database*). Le sous-module *Web3D Personalization Rules* s'attache à choisir les règles de personnalisation qui doivent s'appliquer aux scènes à générer, suivant les informations fournies pour caractériser l'utilisateur. Les résultats sont ensuite stockés dans la base de données relative à l'utilisateur en tant que préférences de présentation.

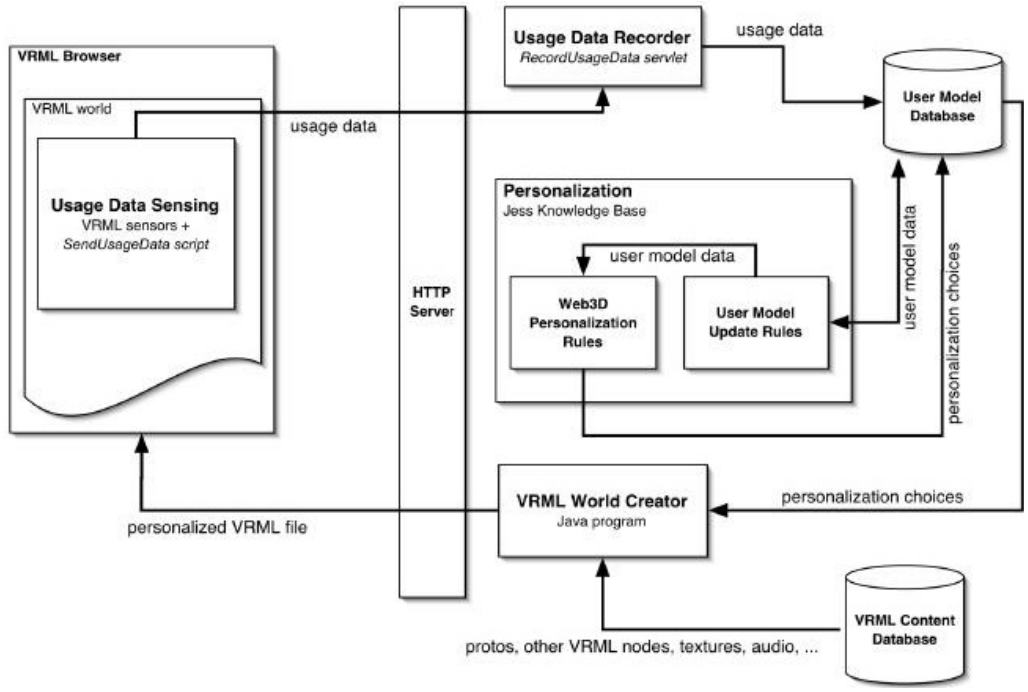


Figure 4.8 Architecture AWE3D pour des systèmes 3D adaptatifs d'après [Chittaro et al.*, 2002].*

La tâche de *génération* est implémentée par le module *VRML World Creator*. Ce module instancie la scène suivant les choix de personnalisation sauvegardés dans la base de données utilisateur concernant l'utilisateur ciblé. La tâche de génération adaptée repose sur l'utilisation de prototypes d'objets 3D paramétrables. Certains paramètres d'entrée du prototype sont mis en correspondance avec les attributs de personnalisation (taille, couleur, etc.) spécifiques à chaque utilisateur, et calculés suite à l'application des règles d'adaptation.

Ce travail propose une démarche et quelques outils de suivi pour la génération adaptative de contenus 3D. Des prototypes englobant la définition des capteurs pour suivre l'évolution de l'utilisateur au sein d'une scène 3D sont fournis. Cependant la description de la scène ainsi que la définition de règles de personnalisation ne se font pas au niveau déclaratif. Des classes Java créées de façon *ad hoc* doivent décrire pour une application spécifique la scène, les attributs à

personnaliser, et les règles à appliquer. En effet, chaque application doit implémenter son propre module de génération de contenu (afin de refléter une organisation spécifique de la scène, ainsi que les objets qui s'y trouvent), son propre module de suivi de l'utilisateur et de mise à jour des préférences de personnalisation.

SLIM-VRT [Estalayo *et al.*, 2004]

La réflexion autour d'une architecture client-serveur pour fournir de l'information 3D adaptée est menée également par [Estalayo *et al.*, 2004] dans le cadre d'une application d'enseignement à distance (*e-learning*) pour la marine. En particulier, le scénario choisi concerne une application dans VRML qui place l'élève à l'intérieur d'un bâtiment de marine virtuel offrant un éventail de possibilités éducatives pour différents profils d'élèves. Les utilisateurs peuvent, au fur et à mesure, accéder à une reconstruction virtuelle du bateau pour suivre un cursus prédéfini. Dans cette application, la description de la scène comporte, en plus de la géométrie, une description de l'information associée aux modèles 3D et la manière de la présenter aux élèves. L'utilisation d'une scène 3D dans le processus d'enseignement offre un cadre propice à la mise en situation des élèves, et les connaissances sont transmises par un ensemble d'éléments multimédia (textes, images, vidéo). Dans cette application, l'adaptation porte principalement sur ces éléments et la manière dont ces éléments sont mis à jour de manière interactive pendant la session. Afin d'atteindre ces buts, les auteurs proposent un découplage entre l'information associée aux modèles et la description VRML de la géométrie de la scène.

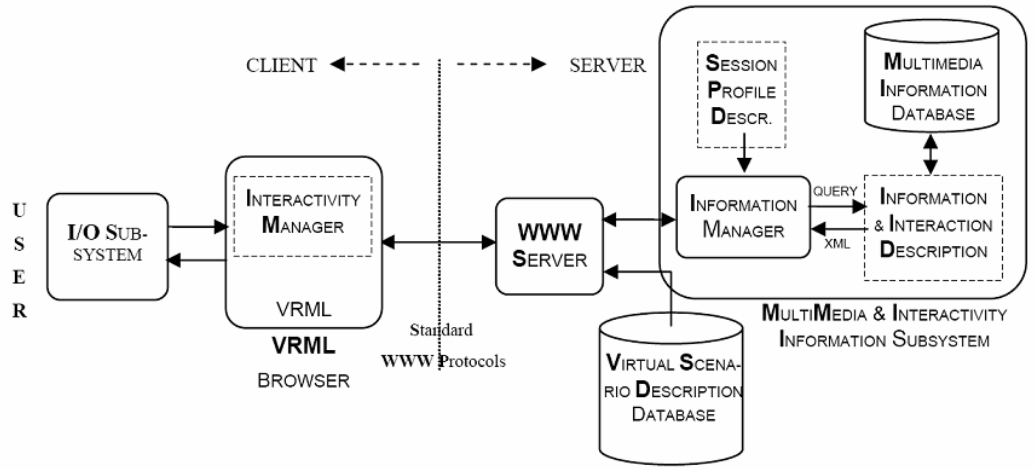


Figure 4.9 Architecture client-serveur pour la diffusion personnalisée dans le cadre d'applications VRML adaptatives d'après [Estalayo et al.*, 2004].*

L'architecture proposée par les auteurs afin de permettre la personnalisation dynamique de l'information multimédia incluse dans les scènes 3D est illustrée dans la Figure 4.9. Trois sous systèmes assurent les tâches de serveurs : le sous-système responsable de la communication avec le client (*WWWS*), la base de données contenant les définitions des scénarios virtuels (*VSDd*) et le sous-système qui concerne la gestion des informations multimédia et la définition de l'interactivité (*MMII*) au sein de l'application. La base de données *VSDd* contient, en plus des éléments géométriques et des textures définissant l'espace d'apprentissage dans lequel l'élève évolue, un ensemble de modèles 3D constituant les éléments qui présentent les notions à acquérir et les éléments 3D introduisant les divers niveaux d'interactivité supportés par le système.

Le sous-système MMIIs, qui a le rôle le plus important dans le processus de personnalisation, est construit autour de quatre composants ayant des responsabilités bien identifiées. L'information associée aux différents éléments d'interactivité dans la scène, est

organisée selon la nature (image, vidéo, audio, texte) et le type d'encodage (HTML, JPEG, GIF, PNG, MPG, etc.) de l'information, au sein d'une base de données (système) destinée à faciliter le processus d'adaptation. La base de données *Information and Interaction description* (IId) retient les informations sur les fichiers contenant les données multimédia associées aux éléments de la scène virtuelle. Sa fonctionnalité première est d'identifier quels sont les données et les types d'interactions supportés pour chaque modèle 3D en fonction des paramètres de la session. Ces descriptions sont réalisées en utilisant XML. Pour chaque modèle 3D adaptable, un fichier décrit le modèle en précisant dans le système, les références aux fichiers contenant les différentes versions de données multimédia à associer avec le modèle en question. Les types d'information qui peuvent être diffusés au sein d'un modèle VRML définissant une partie de l'espace de navigation, sont également définis en utilisant XML.

La description de profils de session *Session Profile description* (SPd) contient des renseignements sur les élèves, et les canaux de communication disponibles. Toutes ces informations réparties entre les trois sources de données, concernant respectivement les données (système), leurs caractéristiques (IId) et le contexte de diffusion (SPd), sont exploitées par le module *Information manager* dans le processus de personnalisation. Son activité se concentre sur deux tâches :

a) la sélection des données multimédia qui sont disponibles pour la session courante de l'utilisateur en concordance avec les valeurs extraites de SPd et IId et

b) l'établissement d'une communication directe avec l'élève à travers le navigateur et le gestionnaire d'interactivité disponibles côté client.

À chaque fois que le module observe des évolutions dans le profil de la session (SPd), les deux tâches s'exécutent en séquence automatiquement de manière transparente, sans que le module ait besoin de recharger la scène dans son ensemble.

Les points importants de cette architecture sont la description indépendante des données 3D qui décrivent l'espace dans lequel l'utilisateur évolue, et l'adaptation de la scène, qui se fait de manière complètement transparente pour l'utilisateur. Cette séparation permet de donner plus de flexibilité au processus d'adaptation et de mieux le maîtriser en permettant la mise en œuvre d'adaptations spécifiques à chaque catégorie d'objets. La transparence des adaptations est acquise en proposant une solution basée sur les outils de contrôle (capteurs et scripts) mis à disposition par le standard VRML. Cependant, les auteurs ne montrent pas comment cette solution, qui vise la personnalisation des éléments d'information de la scène, pourrait compléter les solutions s'intéressant à l'adaptation de l'espace 3D lui-même.

AMACONT [Dachselt *et al.*, 2006]

[Dachselt *et al.*, 2006] présentent une solution pour la personnalisation de contenus et d'applications 3D en considérant à la fois l'utilisateur ainsi que l'hétérogénéité des dispositifs d'accès. L'approche générique proposée par les auteurs s'appuie sur une transposition d'AMACONT [Hinz *et al.*, 2004], une architecture hypermédia adaptative, au monde 3D en utilisant l'approche de modélisation 3D à base de composants Contriga [Dachselt *et al.*, 2002]. AMACONT traite de la génération dynamique de présentations Web adaptées aux besoins des utilisateurs, aux capacités des dispositifs d'accès et aux données contextuelles telles que la localisation des utilisateurs. En étendant cette architecture initialement proposée pour le domaine du multimédia 2D, les auteurs visent principalement la mise en place d'adaptations 3D jouant sur les paramètres de divers objets (*adaptations paramétriques*), et sur l'organisation spatiale de la scène (*adaptations structurelles*).

L'architecture générale du système est présentée dans la Figure 4.10. Les modules *Context Modeling* and *Context Model* gèrent la représentation et la mise à jour des informations caractérisant l'utilisateur et son contexte suite au suivi de l'utilisateur au sein du système. Différents types de capteurs, qui surveillent l'utilisateur et son contexte, sont utilisés afin d'alimenter le profil utilisateur, le profil dispositif et le profil localisation. La représentation de chaque profil est construite en utilisant le standard CC/PP [Klyne *et al.*, 2004] et une grammaire RDF pour décrire les caractéristiques des dispositifs et les préférences utilisateur.

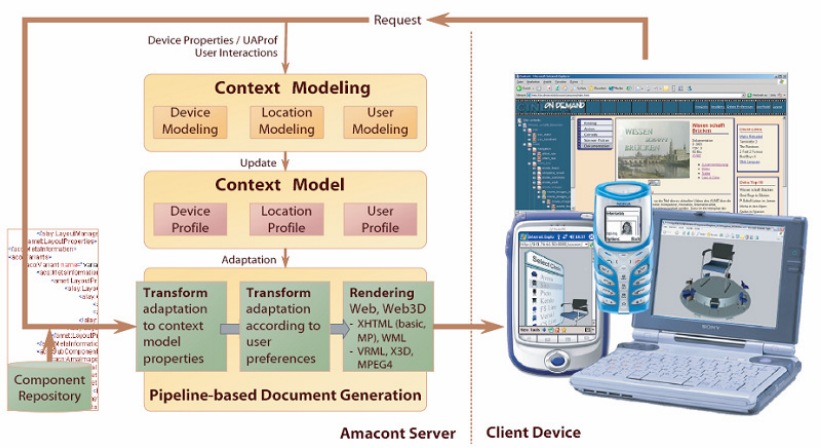


Figure 4.10 L'architecture AMACONT adaptée à la diffusion de scènes 3D [Dachselt et al.*, 2006].*

Le module qui nous intéresse plus particulièrement dans cet état de l'art est celui assurant la génération de documents adaptés (*Document Generation*). Ce module met en œuvre un processus séquentiel d'adaptation en accentuant l'importance des informations contextuelles vis-à-vis des préférences de présentation de l'utilisateur. En effet, les adaptations à base de variants portant sur la géométrie et l'apparence des objets individuels sont les premières à intervenir dans le processus. Ensuite, une étape de personnalisation permet de mettre la scène au goût de l'utilisateur en tenant compte de ses préférences. L'ultime étape du processus concerne le choix du type de sérialisation (VRML, X3D, MPEG4, etc.) le mieux adapté pour le dispositif d'accès de l'utilisateur. La chaîne d'adaptation dans son ensemble est implémentée en utilisant des technologies XML, telles que XSLT et Cocoon. Le choix de ce standard de représentation est dû à la grande flexibilité et la facilité de mise en place des transformations au-dessus d'un format semi structuré tel que XML. Ainsi, l'entrée du processus correspond à un document XML complexe qui encapsule toutes les versions possibles concernant le contenu, la disposition spatiale et la structure logique du document final. Les versions correspondent toutes à une description Contriga [Dachselt *et al.*, 2006] de la scène. En résumé, une scène Contriga est définie comme une instance qui contient la définition de l'environnement de la scène (lumières, points de vues, etc.) et une référence vers un composant contenant la description des objets de la scène.

Dans Contriga, toute composante est décrite par une interface qui contient la liste des paramètres de la composante et une implémentation qui s'apparente à la description d'un graphe de scène. L'interface et l'implémentation sont toutes deux décrites en XML. Cette description à deux niveaux permet de mettre en œuvre des adaptations complexes de manière relativement simple.

Les adaptations paramétriques ou structurelles sont définies en utilisant les variants et les outils de sélection (opérateurs logiques) mis à disposition par la grammaire *AmaAdaptation* définie en tant que schéma XML au sein de l'architecture AMACONT. Un variant correspond à

une occurrence d'une unité structurelle spécifique à un certain contexte de diffusion. Cette occurrence spécifique peut être obtenue en utilisant des paramètres associés à l'unité structurelle. Le choix des valeurs de paramètres (variants paramétriques) en fonction du contexte de diffusion correspond à une adaptation paramétrique. Lorsque les variants portent sur une composante on parle d'adaptation structurelle.

Dans la Figure 4.11 sont illustrées : une description de la structure générale d'une description à base de variants (en haut à droite), une description d'une adaptation paramétrique (à gauche) et une description d'une adaptation structurelle (en bas à droite). La structure générale d'une description à base de variants comporte une première partie dédiée à la description de la manière dont on choisit le variant le plus approprié et une deuxième partie contenant la liste de variants. Chaque variant est identifié par un nom unique et le contenu qui remplace la construction au moment de la génération. Dans la partie gauche de la Figure 4.11, la logique de sélection est illustrée dans le cadre d'une adaptation qui porte sur le paramètre décrivant la couleur des coussins de chaises (`CushionColor`). Une construction de type *if-then-else* permet d'indiquer le choix d'un variant à l'aide de l'élément `aada:ChooseVariant` en fonction de la préférence de l'utilisateur par rapport à la couleur favorite (`<aada:UserParam>Favorite Color</aada:UserParam>`). Lors de la génération de la présentation finale, suivant la couleur favorite renseignée dans le profil utilisateur, le paramètre `CushionColor` vaut `Models/blueCushion.wrl` si bleu est la couleur favorite, `Models/redCushion.wrl` sinon. Ces fichiers `wrl` contiennent la définition de la géométrie et de l'apparence de l'objet. De la même manière, en bas à droite de la Figure 4.11, on définit une adaptation structurelle. Suivant la largeur de l'écran du dispositif d'accès de l'utilisateur, un composant de type menu circulaire ou un composant menu flottant, est proposé à l'utilisateur.

Figure 4.11 Description d'adaptations dans AMACONT [Dachselt et al.*, 2006].*

D'autres types d'adaptation concernant des documents Web comportant plusieurs types d'objets média (SVG, XHTML, X3D, etc.) peuvent être décrits en utilisant cette approche. La proposition d'un moyen de décrire la logique des adaptations et leurs alternatives est un pas en avant vers la large adoption des outils d'adaptation.

Cependant, nous considérons que cette solution présente l'inconvénient que toutes les futures situations auxquelles le contenu peut faire face, doivent être décrites à l'avance. Cela se traduit par des documents très complexes contenant en somme toutes les combinaisons de

variants envisageables pour déployer un certain type de contenu dans une variété de contextes. En contrepartie, la description exhaustive des adaptations garantit que la scène générée ne comporte pas d'anomalies qui peuvent apparaître lors de l'application des méthodes d'adaptation déclenchées sans une maîtrise totale du processus.

Une autre limitation que nous voulons mettre en évidence est le fait que la solution d'adaptation est décrite de manière directive. La logique de sélection ne comporte pas de formalismes permettant d'envisager de choisir la mise en application d'un sous-ensemble optimal d'adaptations. Par exemple, lorsque l'on applique des adaptations relatives au contexte de diffusion il se peut qu'en ne substituant que quelques objets on arrive à une scène pouvant être délivrée dans de bonnes conditions. Les critères pour la sélection des variants ne sont plus réévalués au fur et à mesure que le processus d'adaptation se déroule. Cependant, il est envisageable que, suite à une première adaptation, l'ensemble des contraintes matérielles ou des préférences utilisateur soit respecté. La solution proposée applique toutes les adaptations paramétriques ou structurelles préconisées à l'avance par les variants.

CPS [Chittaro *et al.*, 2004]

[Chittaro *et al.*, 2004] s'intéressent à la description et la formalisation des adaptations au sein des scènes 3D. Les auteurs ne suivent pas la même approche que [Dachselt *et al.*, 2006], où pour les objets visés par l'adaptation, l'ensemble des variants est décrit, mais ils désignent pour chaque objet l'adaptation qui s'y prête le mieux. Les auteurs s'appuient sur l'encodage XML de X3D afin de mettre en place une architecture similaire à AHA définie dans [De Bra *et al.*, 2003] pour les hypermédia adaptatifs 2D. Cependant les passages de 2D en 3D et de XHTML à X3D obligent les auteurs à apporter certaines modifications. Dans AHA, le problème d'adaptativité était résolu par l'introduction des éléments <object> instanciés, lors de la diffusion, avec l'information adéquate en respectant les contraintes de déploiement et le profil utilisateur. L'utilisation de ces éléments propres au langage XHTML rend le processus d'adaptation transparent vis-à-vis d'une architecture standard de déploiement. En X3D, il n'y a pas de tel élément à vocation générique et les auteurs décident de garder l'information relative aux adaptations à part dans un fichier XML appelé *Content Personalisation Specification* (CPS). Ce fichier CPS est associé à un fichier X3D et contient une liste d'éléments `adaptativeContent` qui désigne les éléments sujets aux adaptations.

```
<X3D>
  <head> … </head>
  <Scene>
  <Transform DEF="prod_1"  translation="…">
    <Shape>…</Shape>
  </Transform>
  <Transform DEF="prod_2" translation="…">
    <Shape>…</Shape>
  </Transform>
  …
  <Scene>
</X3D>
```

```
<CPS>

  <adaptiveContent
        DEF="prod_1"
        name="specialOffer_1"/>

  <adaptiveContent
        DEF="prod_2"
        attribute="scale"
        name="prod_2.size"/>
    …
</CPS>
```

Figure 4.12 Notation CPS [Chittaro et al.*, 2004] pour décrire des adaptations 3D.*

Dans la partie gauche de la Figure 4.12, est illustré le fichier X3D qui est assujetti aux adaptations décrites dans la partie gauche à l'aide de la notation CPS. Le premier élément `adaptativeContent` indique le nœud X3D dont l'identifiant (`DEF`) est `prod_1`, doit être substitué avec le contenu (`specialOffer_1`) dérivé par le moteur d'adaptation. Le deuxième `adaptativeContent` indique que l'élément dans la scène dont l'identifiant est `prod_2` doit subir l'insertion de l'attribut `scale` qui régit la taille de l'objet. La valeur de cet attribut est fournie par le moteur d'adaptation à l'aide de la référence saisie (`prod_2.size`). Lorsque le fichier X3D

associé est demandé pour visualisation, le fichier CPS est transformé en utilisant XSLT dans une série de requêtes adressées au moteur d'adaptation. Ensuite les fragments résultats sont insérés dans le fichier X3D à l'emplacement désigné par l'attribut `DEF` de chaque élément `adaptativeContent`.

Cette séparation, entre le contenu et les règles d'adaptation qui le concernent, nous paraît une très bonne perspective de travail. Ceci permet d'envisager la construction de profils de règles d'adaptation à appliquer en fonction d'un contexte de diffusion spécifique. Ce que nous reprochons à cette proposition est la pauvreté d'expression offerte par le CPS. Au niveau de l'indication du contenu à adapter dans la scène, l'utilisation des identifiants uniquement suppose qu'auparavant tous les objets concernés par des futures adaptations sont identifiables de manière unique. Ainsi, les règles sont spécifiques et il n'est pas du tout envisageable d'associer le même profil d'adaptation à plusieurs scènes en ne désignant pas des objets précis mais plutôt des classes d'objets. Tel que cela est présenté dans [Chittaro *et al.*, 2004], il est impossible d'indiquer plusieurs types d'adaptation pour un même item. De plus, il n'est pas possible d'indiquer quels sont les facteurs qui imposent le déclenchement d'une adaptation. Seules les adaptations par rapport aux préférences des utilisateurs sont effectuées. La mise en relation entre les préférences utilisateur et les attributs des objets 3D (comme par exemple la taille) peut être vue comme un premier pas dans cette direction. Cependant, cette méthode est assez pauvre car d'autres contraintes (comme par exemple le débit du réseau, la taille de l'écran), n'ayant pas une correspondance directe avec les attributs des objets 3D, ne peuvent pas être intégrées dans le CPS. Cela est principalement dû au fait que la notion de contexte ne semble pas très bien intégrée dans le CPS. À notre avis, les CPS peuvent être vus plus comme des feuilles de styles associées aux données 3D, étant en quelque sorte l'équivalent de CSS pour les pages HTML.

4.4. Synthèse

Dans ce chapitre nous avons présenté un large panorama des techniques d'adaptation développées au sein de la communauté 3D. Nous avons également inclus des propositions, issues des travaux de la communauté multimédia, dont le degré de généricité les rend intéressantes pour la problématique spécifique aux données 3D.

Une section a été dédiée à la présentation de cette problématique en illustrant l'ensemble de dimensions (géométrie, apparence, structure logique et encodage) auxquelles nous devons nous intéresser afin de cerner la complexité des processus d'adaptation. En plus des aspects techniques liés à la mise en œuvre des adaptations, **nous nous sommes également intéressés au degré de généricité des solutions proposées et aux moyens de décrire les éventuelles stratégies d'adaptation.**

Cette grande variété de solutions offre un grand choix pour la diffusion, cependant les solutions étudiées imposent **des contraintes de représentation ou d'expression de critères d'adaptation** qui restent implicites au niveau de chaque proposition. **Ce que nous regrettons c'est le manque de standardisation des outils de transformation à appliquer à la scène.** Dans le domaine du multimédia 2D, les propositions NAC [Lemlouma, 2004] et MPEG-21 DIA [Vetro *et al.*, 2005] semblent contenir des pistes potentiellement intéressantes. Cependant, elles non plus **ne permettent pas aux utilisateurs de définir des règles d'adaptation de manière déclarative**.

Une autre critique que nous apportons à l'ensemble des méthodes présentées ici, est qu'elles **ne mettent pas la sémantique des objets au centre du processus d'adaptation**. Nous ne pouvons pas adapter de la même manière un modèle de terrain qu'un bâtiment ou un arbre.

L'absence d'un moyen de formaliser les techniques permettant de définir la manière dont les objets, ou les classes d'objets au sein d'une scène 3D, sont adaptés (tous les arbres sont dégradées de manière uniforme) nuit à adoption à grande échelle des techniques d'adaptation dans les diverses applications de la 3D.

Les solutions concernant les adaptations de données 3D **s'appuient très souvent sur des connaissances indiquées par les concepteurs de l'adaptation** (listes de variants, identification directe des objets). Ceci limite les possibilités de formuler des stratégies d'adaptation qui jouissent d'un certain degré de généricité et de réutilisation. Avec les outils disponibles aujourd'hui il est **difficilement envisageable de construire des stratégies d'adaptation génériques** qui puissent s'appliquer à toute une famille de scènes (par exemple, les scènes modélisant des environnements urbains).

Dans la mesure où les concepteurs disposeraient de moyens leur permettant de **décrire les transformations les mieux adaptées à telle ou telle catégorie d'objets**, le processus d'adaptation resterait fidèle à l'intention initiale du concepteur. **La formalisation** d'un tel moyen de description **permet aux concepteurs de scènes ou aux utilisateurs ne disposant pas forcément des compétences requises, de mettre en place des transformations** sur la scène. Ces transformations visent à réduire la complexité de la scène, ou à personnaliser certains de ses aspects (apparence de certains objets, filtrage sémantique, etc.).

Proposition

Dans l'état de l'art que nous venons de conclure, nous avons remarqué un engouement certain autour de la sémantique, cependant la représentation et l'utilisation de celle-ci ne dépassent pas le cadre d'une application spécifique. Les mécanismes de réutilisation sont très peu formalisés et l'apport de la sémantique est très peu exploité. En dépit de leur grand nombre et de leur diversité, les techniques d'adaptation sont généralement appliquées de manière ponctuelle, en étroite relation avec une solution logicielle spécifique. Les concepteurs de scènes ne disposent pas d'outils et de modèles permettant de décrire des stratégies d'adaptation indépendamment d'une application spécifique.

Ce que nous cherchons principalement dans cette thèse, est de proposer des solutions d'enrichissement sémantique qui ouvrent la voie vers des moyens de réutilisation et d'adaptation jouissant d'une certaine indépendance par rapport à une modélisation 3D des données et à une représentation spécifique de la sémantique.

Dans cette deuxième partie du manuscrit, nous décrivons nos propositions pour enrichir les possibilités de réutilisation des données et de leurs sémantiques par rapport à une large variété de domaine d'application. Nous œuvrons également pour une formalisation des requêtes visant la réutilisation et l'adaptation de données 3D par génération automatique de scènes. Nous organisons notre contribution en quatre parties :

- **3D SEmantic Annotation Model (3DSEAM) –** est un *modèle* générique permettant de caractériser *la géométrie*, *l'apparence*, *la dimension document ou profil média* et les informations relatives aux objets du monde réel modélisés par les objets 3D d'une scène. Pour répondre à la variabilité de la sémantique dans différentes applications, nous optons pour une solution flexible qui sépare le contenu et la sémantique. Cette contribution est détaillée dans le chapitre 5.
- **3D Annotation Framework (3DAF)** – est une plate-forme qui permet d'exploiter les connaissances renseignées dans le modèle 3DSEAM. Elle introduit une couche abstraite qui assure l'indépendance entre une représentation particulière des instances du modèle 3DSEAM et le moyen d'accès à l'information. Cette contribution est détaillée dans le chapitre 6.
- **3D Semantic Data Library** – est une plate-forme pour faciliter la *recherche et la réutilisation de contenus 3D sémantiques*. Ces opérations sont mises en œuvre au-dessus de la plate-forme 3DAF. Nous proposons un langage de description de patrons de scènes 3D, dont le contenu inclus est concordant avec les critères sémantiques, géométriques et d'apparence formulés par l'utilisateur. Cette contribution est détaillée dans le chapitre 7.
- **Adapt3D** – est une plate-forme qui supporte la mise en place des adaptations sur les documents 3D. Nous proposons une solution qui permet pour chaque

catégorie d'objets, au sein d'une scène, d'indiquer la technique d'adaptation la plus appropriée. Les catégories sont définies à l'aide des critères portant sur les caractéristiques des objets 3D (sémantique, géométrie, …). Cette contribution est détaillée dans le chapitre 8.

Les plates-formes 3DAF, 3DSDL et Adapt3D sont en cours d'implémentation. Dans les chapitres qui leur sont consacrés nous insistons sur l'architecture des plates-formes, sur leurs fonctionnalités, sur les modèles sous-jacents et sur les conventions de représentation facilitant la communication entre les différents modules de chaque plate-forme.

L'ensemble de nos contributions (et notamment les solutions en termes de modèles) pour favoriser la réutilisation et l'adaptation est validé sur un cas d'étude décrit au chapitre 9. Celui-ci repose sur une modélisation 3D des bâtiments du campus universitaire de Grenoble. Au-dessus de cette modélisation, nous implémentons les approches de réutilisation et d'adaptation que nous avons proposées afin de générer des scènes 3D en accord avec les critères exigés par les utilisateurs.

5. Modèle pour la caractérisation des données 3D par annotations sémantiques

Dans ce chapitre, nous présentons le modèle *3D SEmantic Annotation Model* (3DSEAM) que nous avons conçu dans le but de faciliter la caractérisation des données 3D. Les différents stades d'évolution du modèle ont été rapportés dans [Bilasco *et al.*, 2005b][Bilasco *et al.*, 2005c][Bilasco *et al.*, 2006a][Bilasco *et al.*, 2007b]. Le modèle est un support pour stocker et organiser les connaissances sur les données 3D, obtenues soit par extraction automatique, soit au moyen d'annotations. Nous considérons ces deux types de caractérisation de données car, comme montré dans l'état de l'art, à ce jour il n'existe pas de technologies matures et indépendantes d'un domaine spécifique d'application capables d'extraire la sémantique des données de manière automatique.

Ainsi, si nous voulons être en mesure de caractériser les données 3D sous tous leurs aspects (**géométrie** – position dans la scène, … ; **apparence** – couleur dominante, … ; **profil média** – taille du fichier, … ; **sémantique** – propriétés de l'entité du monde représentée par la donnée, …), **nous devons compter sur l'utilisation d'annotations** qui peuvent se révéler imprécises, incomplètes et très hétérogènes. L'alimentation en information d'un entrepôt de caractéristiques des données 3D se fait principalement à base d'annotations réalisées par des utilisateurs sur les données 3D de diverses scènes accessibles à travers le Web. Des clients logiciels, capables de combler l'écart qui existe toujours entre le niveau signal et le niveau sémantique d'un objet multimédia, peuvent également alimenter un modèle de caractérisation sémantique.

L'instanciation du modèle 3DSEAM se traduit par la construction d'un entrepôt de descriptions contenant des connaissances sur un large ensemble de modèles 3D. Ces connaissances jouent un rôle important dans les systèmes de recherche, réutilisation et adaptation de ces données considérées comme des entités indépendantes ou assemblées dans le cadre de scènes complexes. Nous proposons **un modèle générique qui permet de caractériser de**

manière homogène toutes les dimensions d'une scène 3D et les divers types de média (images, vidéo, sons) qui enrichissent la description géométrique.

Afin de mieux maîtriser le processus de gestion des annotations, nous les organisons en trois grandes catégories décrivant respectivement : la nature multimédia des documents matérialisant les scènes 3D (taille, type d'encodage, etc.), les caractéristiques intrinsèques de la forme des données 3D (géométrie, structure, apparence) et les connaissances sémantiques. La sémantique attachée à la donnée 3D peut être définie par rapport au rôle joué par cette donnée au sein d'une scène. En même temps, nous envisageons un deuxième niveau plus générique qui décrit les propriétés sémantiques de la donnée 3D indépendamment d'une scène ou application 3D spécifique.

En ce qui concerne la représentation utilisée pour modéliser ces connaissances, nous optons pour l'utilisation d'un modèle objet de description des dimensions d'une scène 3D. Nous souhaitons nous affranchir d'un encodage spécifique afin d'aboutir à une solution générique de représentation. Ainsi, nous voulons mettre l'accent sur l'importance de la prise en compte de toutes les caractéristiques d'une donnée 3D (géométrie, apparence, topologie, sémantique, profil média) dans les processus de réutilisation et d'adaptation de celle-ci et non pas sur un encodage spécifique des connaissances. Nous choisissions un modèle à base d'objets, car nous considérons que cela nous confère des possibilités de formalisation suffisamment génériques pour supporter l'ensemble de solutions d'encodage de connaissance possibles. Ce modèle est facilement transposable ensuite dans un encodage spécifique (MPEG-7 [Martinez *et al.*, 2002], RDF [Manola *et al.*, 2004]) lié à des outils comme AROM [Page *et al.*, 2000] ou comme un SGBD relationnel, etc.

Dans la suite, nous passons brièvement en revue le processus de caractérisation au niveau sémantique d'une scène 3D et nous mettons en évidence les dimensions des données 3D auxquelles nous nous intéressons. Nous poursuivons avec la présentation de chaque dimension de caractérisation : la localisation, la structure logique, la géométrie, l'apparence, la topologie et la sémantique.

5.1. Aperçu du processus de caractérisation des données

Nous considérons un processus de caractérisation qui se réalise en trois temps :

a) identifier les données 3D au sein d'une scène.

Suivant le langage de modélisation utilisé, l'organisation logique de la scène est plus ou moins enfouie dans le document matérialisant la scène. Des outils et des notations spécifiques correspondant à chaque type de modélisation supporté par la plate-forme doivent être pris en compte. Nous avons identifié deux grandes classes de notations permettant de localiser du contenu au sein d'une scène 3D :

- **les notations structurelles** qui indiquent la position dans le document du fragment multimédia correspondant au contenu cible. Dans le cas de documents structurés tels que l'encodage XML de X3D, la notation structurelle prend la forme d'une expression XPATH [Clark *et al.,* 1999]. Dans le cas d'un encodage binaire, la notation structurelle correspond à l'indication de la position du début et de celle de la fin dans le flux binaire encodant le contenu ciblé.
- **les notations spatiales** qui définissent des volumes et/ou surfaces dont l'enveloppe contient le contenu ciblé.

b) **analyser de manière automatique les fragments multimédia** identifiés afin d'extraire les informations relatives aux **caractéristiques multimédia**, de **la géométrie**, de **l'apparence**, de **la topologie**.

Ces dimensions des fragments ne sont pas stockées dans leur état brut (à bas niveau), mais nous retenons en priorité leurs caractéristiques de haut niveau plutôt que les détails de modélisation. Plusieurs types de fragments multimédia sont supportés. En plus des fragments multimédia représentant des contenus 3D, nous caractérisons également des contenus 2D et audio diffusés au sein de la scène. Les contenus 2D peuvent correspondre aux textures qui habillent la géométrie 3D de la scène.

c) **associer les fragments multimédia** à une ou plusieurs **entités sémantiques**.

Chaque entité est décrite par un ensemble de propriétés et relations organisées en plusieurs profils sémantiques. À ce niveau l'entité est caractérisée indépendamment des scènes où les fragments multimédia qui lui sont associés se trouvent. Un deuxième niveau permet de décrire les propriétés sémantiques de chaque fragment multimédia au sein de la scène d'origine. Nous considérons que ces deux niveaux (générique et local) de description sémantique sont nécessaires afin d'accroître la flexibilité du processus de gestion de connaissances sémantiques.

Nous proposons un modèle qui supporte le processus de caractérisation et stocke les principales propriétés des données 3D selon l'approche de caractérisation illustrée précédemment. Le modèle ne se veut en aucun cas être une solution ultime de caractérisation, mais il est issu de la volonté de montrer l'importance de la prise en compte des caractérisations de haut niveau des dimensions des données 3D. En fonction d'une application spécifique, certaines parties du modèle prennent une plus grande importance et sont détaillées d'une manière plus poussée que d'autres.

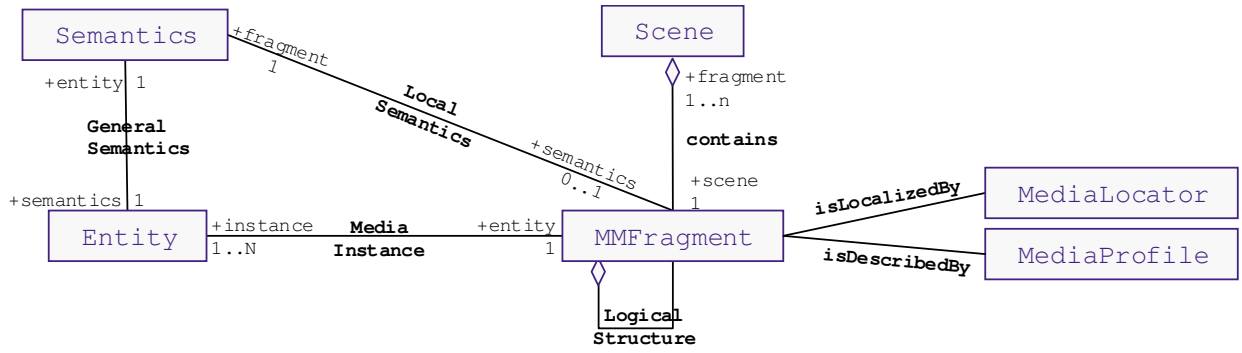


Figure 5.1 Aperçu du modèle 3D SEmantic Annotation Model.

Une vue générique du modèle, qui résume cette première section, est présentée dans la Figure 5.1. Une scène (`Scene`) est composée d'une série de fragments multimédia (`MMFragment` – objets 3D, textures, sons) localisés au sein de scènes par des repères spécifiques (`MediaLocator`). Ces fragments caractérisés par leur profil média (`MediaProfile`), peuvent également être associés aux entités du monde (`Entity`), chacune caractérisée par un ensemble d'informations sémantiques (`Semantics`). Les entités permettent donc, indirectement, d'associer de la sémantique aux fragments média indépendamment d'une scène et d'une application spécifique. Un niveau sémantique spécifique au fragment dans la scène est introduit grâce à une relation directe (`LocalSemantics`) entre le fragment multimédia et la classe sémantique (`Semantics`) représentant

la sémantique locale. Cette séparation entre les deux niveaux sémantiques confère une certaine flexibilité et facilite la réutilisation de notions sémantiques entre les différentes matérialisations d'une même entité à travers divers fragments multimédia.

Le modèle n'introduit pas directement la classe des fragments 3D, mais il utilise la notion de fragment multimédia qui est par la suite spécialisée soit en tant que fragment 3D, soit en tant que fragment 2D (texture image ou vidéo), soit en tant que fragment audio. Les caractéristiques d'apparence, géométriques et topologiques, que nous avons étudiées dans le paragraphe précédent, sont associées à l'extension `Object3D` de la classe `MMFragment` comme indiqué dans le diagramme UML présenté dans la Figure 5.2.

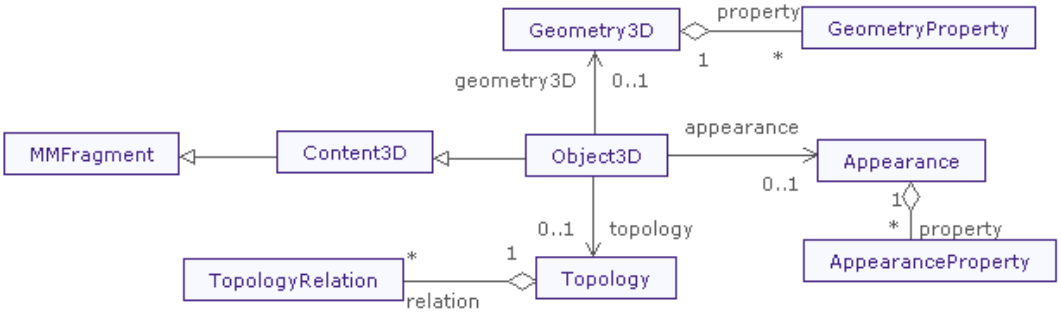


Figure 5.2 Partie spécifique 3D du modèle 3DSEAM.

Les deux diagrammes présentés dans la Figure 5.1 et la Figure 5.2 contiennent le squelette fixe de notre modèle. Nous considérons qu'indépendamment d'une application donnée il est impératif de séparer la sémantique, les entités et l'ensemble des caractéristiques média d'un objet 3D.

Pour chaque dimension de caractérisation que nous retenons dans notre modèle (géométrique, topologique, d'apparence), en fonction des besoins applicatifs spécifiques, une certaine variabilité des caractéristiques à exploiter peut être observée. Ainsi, par exemple, alors que des applications s'intéressent aux caractéristiques de forme de l'objet 3D en utilisant les descripteurs de forme de MPEG-7 [Zaharia *et al.,* 2002], d'autres applications s'appuient, pour cette même notion de forme, sur d'autres descripteurs déterminés par exemple, à partir de relations spatiales pondérées [Del Bimbo *et al.,* 1998], de projections sphériques [Kazhdan *et al.,* 2003], d'*images de spin* [Johnson *et al.,* 1999], de vues caractéristiques [Mahmoudi *et al.,* 2002], [Weiss *et al.,* 2001]. En conséquence, dans 3DSEAM les classes sont associées à des listes de propriétés qui stockent les descripteurs caractérisant les dimensions d'une donnée 3D. Nous proposons à la fin de ce chapitre un modèle de descripteurs qui permet de gérer la variabilité dans le choix des descripteurs.

Ce choix de modélisation, qui donne une grande flexibilité à notre proposition, permet d'accroître le niveau d'adaptation du modèle aux besoins potentiels. La liste des descripteurs pour une dimension peut, en effet, être constitué en fonction des besoins applicatifs. En contrepartie, il nous oblige à mettre à la disposition des utilisateurs du modèle des moyens de stockage, de gestion (ajout, suppression, mise à jour) et d'interrogations robustes, prenant en compte cette variabilité. En ce qui concerne l'exploitation effective de ces informations, des réponses dans ce sens sont apportées dans le chapitre suivant consacré à la gestion des instances du modèle 3DSEAM.

Après cet aperçu des principaux concepts mis en avant par le modèle 3DSEAM, nous présentons les méthodes qui nous permettent de localiser le contenu sémantique au sein de différents types de scènes 3D. Nous commençons par la dimension relative à la localisation des

contenus. Les repères utilisés nous permettent de bien cibler et délimiter les fragments de la scène auxquels nous nous intéressons et auxquels nous voulons associer une caractérisation de haut niveau. L'identification, au sein d'une scène 3D, de parties cohérentes d'un point de vue spatial et sémantique est déjà une manière basique d'apporter de la sémantique à la scène. En effet, cela reflète l'organisation logique d'une scène selon le point de vue des utilisateurs de la scène. Comme nous l'avons souligné dans le premier chapitre, la structure logique de la scène telle que celle encodée par le concepteur, traduit plus un objectif d'optimisation de la visualisation que l'organisation naturelle.

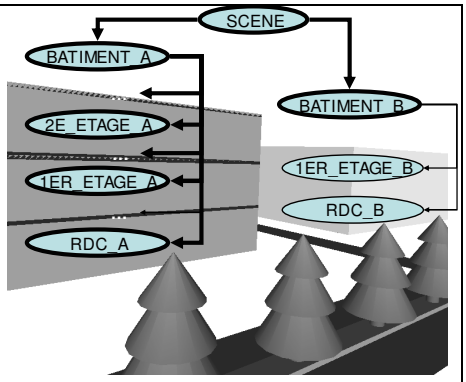
5.2. La localisation du contenu 3D

Le procédé de localisation exige l'identification des éléments géométriques qui forment la représentation d'objet. Dans le cas d'une image 2D, des objets sont associés aux sous-régions définies par des ensembles de polygones 2D. Dans les espaces 3D, on peut localiser des objets en choisissant les surfaces 3D et les volumes qui enveloppent un objet dans une scène. Par exemple, [Funkhouser *et al.*, 2004] proposent un outil « ciseaux magiques » qui aide les concepteurs à découper des parties de modèles 3D en définissant des volumes qui suivent les lignes et les surfaces de séparation. La nature des éléments localisés peut varier d'éléments géométriques simples (surfaces, volumes) à des fragments complexes de document (par exemple un groupe d'éléments géométriques). Nous distinguons deux types de localisation :

- localisation structurelle (pour les entrées de document) et
- localisation spatiale (consacrée aux éléments géométriques).

Pour illustrer ces deux approches de localisation, nous proposons une scène 3D décrite par un document X3D dont l'encodage XML, permettant une localisation structurelle, est présenté dans la *Figure 5.3*. D'autres scènes sont considérées plus loin afin d'illustrer les différents outils de localisation spatiale.

```
 1: <X3D version='3.0' profile='Interactive'>
 2:  <Scene>
 3:   …
 4:   <Group DEF='BATIMENT_A'>
 5:    <Transform translation='0.0 0.0 0.0'>
 6:     <Group DEF='ETAGE_BATIMENT_A'>
 7:      <Shape>
 8:       <IndexedFaceSet DEF='BASE_BATIMENT_A'…>
 9:         ...</IndexedFaceSet>
10:       <Appearance>
11:        <Material diffuseColor='0.6 0.6 0.6'/>
12:        </Appearance>
13:      </Shape>
14:     </Group>
15:    </Transform>
16:    <Transform translation='0.0 3.13 0.0'>
17:     <Group DEF='SEP_BATIMENT_A'>
18:      <Shape>
19:       <IndexedFaceSet USE='BASE_BATIMENT_A'…>
20:       ...</IndexedFaceSet>
21:       <Appearance>
22:        <Material diffuseColor='0.6 0.6 0.6'/>
23:       </Appearance>
24:      </Shape>
25:     </Group>
26:    </Transform>
27:    <Transform translation='0 3.33 0'>
28:     <Group USE='ETAGE_BATIMENT_A'/>
29:    </Transform>
30:    <Transform translation='0 6.46 0'>
31:     <Group USE='SEP_BATIMENT_A'/>
32:    </Transform>
33:    <Transform translation='0 6.66 0'>
34:     <Group USE='ETAGE_BATIMENT_A'/>
35:    </Transform>
```




```
36:    <Transform translation='0 9.79 0'>
37:     <Group USE='SEP_BATIMENT_A'/>
38:    </Transform>
39:   </Group>
40:   <Group DEF='BATIMENT_B'>
41:    <Transform translation='0.0 0.0 0.0'>
42:     <Shape>
43:      <IndexedFaceSet …>…</IndexedFaceSet>
44:      <Appearance>
45:       <Material diffuseColor='1 1 1' />
46:      </Appearance>
47:     </Shape>
48:    </Transform>
49:   </Group>
50:   ...
51:  </Scene>
52: </X3D>
```

Figure 5.3 Fragment d'une scène X3D représentant deux bâtiments.

La scène contient deux bâtiments (BATIMENT_A, BATIMENT_B) dont la modélisation partielle est présentée dans l'extrait, ainsi que quelques éléments de décor (arbres, routes) dont l'encodage n'est pas présenté dans l'extrait afin de faciliter l'illustration. La structure logique de la scène permet d'identifier les deux bâtiments (lignes 4-39 pour le BATIMENT_A et lignes 40-49 pour le BATIMENT_B). De plus, pour le bâtiment BATIMENT_A la description contient également la séparation du bâtiment en plusieurs étages (lignes 5-15 pour le rez-de-chaussée, lignes 27-29 pour le premier étage, lignes 33-35 pour le deuxième étage) et les plafonds les séparant (lignes 16-26, lignes 30-32 et lignes 36-38 respectivement).

La modélisation proposée fait appel aux moyens de réutilisation (DEF/USE) supportés par le langage X3D dont nous avons discutés dans le chapitre 2. Les valeurs des attributs DEF peuvent être utilisées afin d'identifier des parties du document 3D, cependant on ne peut pas toujours s'appuyer dessus. Dans le cas présent, en ce qui concerne les nœuds introduisant les deux bâtiments (lignes 4 et 40), les valeurs des attributs DEF correspondent effectivement au nom des objets que l'on souhaite localiser dans la scène, mais ce n'est pas toujours le cas. Les attributs DEF servent uniquement au processus interne de représentation efficace de la scène et ne peuvent pas être exploitées directement en tant que localisateur de contenu ayant une connotation sémantique bien établie.

Le fait que les étages du bâtiment BATIMENT_A soient introduits explicitement dans la structure logique du document, permet de traverser celle-ci (en utilisant XPATH [Clark *et al.*, 1999] par exemple) pour accéder directement à un fragment (par exemple, le bâtiment ou l'un de ses étages). Cette façon d'accéder à la représentation correspondant à un des étages ne peut pas être appliquée au bâtiment BATIMENT_B, car la primitive utilisée (ligne 43) pour la modélisation géométrique ne fait pas de distinction entre les deux étages du bâtiment.

Superposée sur la vue de la scène, on trouve la structure logique de la scène (*Figure 5.3*). Les traits épais indiquent les relations qui peuvent être extraites directement de la structure logique du document, et les traits fins représentent les relations qui n'ont pas d'équivalence structurelle dans le document. Ces relations sont mises en évidence par des constructions externes au document. Dans notre approche, ces constructions correspondent aux repères spatiaux qui délimitent les aires et les volumes correspondant aux objets.

Dans la suite de cette section, nous nous intéressons tout d'abord aux moyens permettant la localisation structurelle. Nous abordons ensuite la localisation spatiale. Nous consacrons également une partie aux repères obtenus en combinant la localisation spatiale et structurelle. Enfin, nous introduisons ces outils de localisation au sein du modèle 3DSEAM.

5.2.1. Localisation structurelle

La localisation structurelle exploite la structure du document matérialisant la scène pour indiquer les entrées du document qui correspondent à l'objet.

Localisation intra-document

Afin d'illustrer ce type de localisation, nous considérons la scène décrite dans la *Figure 5.3*. Pour référencer les fragments 3D représentant les différents étages du bâtiment A, qui ne sont pas directement définis par un identifiant (attribut DEF), nous proposons les constructions de type XPATH suivantes :

- `//Group[DEF='BATIMENT_A']/Transform[position()=0]` pour l'objet 3D représentant le rez-de-chaussée (lignes 5-15 dans la *Figure 5.3*) ;

- `//Group[DEF='BATIMENT_A']/Transform[position()=2]` pour l'objet 3D représentant le premier étage (lignes 27-29 dans la *Figure 5.3*) ;
- `//Group[DEF='BATIMENT_A']/Transform[position()=4]` pour l'objet 3D représentant le deuxième étage (lignes 33-35 dans la *Figure 5.3*).

De la même manière nous identifions l'ensemble des fragments 3D modélisant les plafonds séparant les étages du bâtiment A. Même si les objets géométriques sont éparpillés à travers le document, avec l'introduction d'un repère structurel complexe nous pouvons les référencer comme un élément unitaire. Le repère structurel de la Figure 5.4 regroupe l'ensemble des fragments 3D matérialisant les plafonds du bâtiment A (lignes 16-26, lignes 30-32, lignes 36-38 de la *Figure 5.3*).

```
{//Group[DEF='BATIMENT_A']/Transform[position()=1],
 //Group[DEF='BATIMENT_A']/Transform[position()=3],
 //Group[DEF='BATIMENT_A']/Transform[position()=5]}
```

Figure 5.4 Repére structurel multiple.

Ce moyen de localisation vient compléter la manière plus classique d'identification des objets au sein des documents 3D par leur identifiant. Cette solution complémentaire a un double intérêt :

- permettre **de référencer des objets qui ne sont pas munis d'un identifiant**,
- permettre **de définir une structure sémantique de la scène**, différente de celle imposée par défaut par le concepteur qui se calque sur la structure logique du document.

Nous considérons ce deuxième point très important car, par sa nature, la structure logique de la scène ne cherche pas directement à rendre compte de l'organisation sémantique de la scène. L'organisation structurelle du document 3D ne correspond pas toujours à la vision qu'ont les utilisateurs de l'organisation sémantique de la scène. De plus, lorsque l'on envisage de réutiliser une même scène dans des applications issues de domaines différents, on peut ne pas avoir la même vision en ce qui concerne l'organisation des objets au sein de la scène.

Localisation inter-documents

Les exemples de localisation que nous avons présentés jusqu'ici sont tous donnés à l'intérieur de la même scène. Cependant, une technique classique lorsque l'on traite de volumes importants de données est de découper la scène entière en plusieurs morceaux, chacun des morceaux couvrant une région spatiale spécifique de la scène. Afin de pouvoir référencer un objet 3D dont une partie de sa géométrie est éparpillée à travers plusieurs documents 3D, il est nécessaire de prévoir dans la description de chaque repère structurel le nom du document concerné.

Afin d'illustrer ce cas de figure, nous considérons un document 3D qui est construit à partir de deux sous documents importés (au moyen des éléments `Inline` – lignes 4 et 7 du fichier `BATIMENTS_CD.X3D` de la Figure 5.5), chacun présentant un bâtiment distinct. Supposons qu'il existe un service administratif dont les locaux sont répartis dans les deux bâtiments (par exemple les rez-de-chaussée des deux bâtiments). Si l'on veut associer un objet (nommé `SERV_ADM`) qui correspond à l'ensemble des éléments géométriques, nous devons employer un repère structurel multiple qui contient deux repères structurels locaux à chacune des scènes. Ainsi, nous utilisons le repère `//Group[DEF='BATIMENT_C']/Transform [position()=0]` afin de localiser l'élément

(SERV_ADM_C - lignes 5-12 dans le fichier BATIMENT_C.X3D de la Figure 5.5) modélisant le rez-de-chaussée, respectivement le repère //Group[DEF='BATIMENT_D']/Transform[position()=0] afin de localiser l'élément (SERV_ADM_D - lignes 5-12 dans le fichier BATIMENT_D.X3D de la Figure 5.5) modélisant le rez-de-chaussée du bâtiment D dans le document BATIMENT_D.X3D. Ceci est synthétisé dans la partie basse de la Figure 5.5, en utilisant un repère structurel multiple pour référencer les fragments (SERV_ADM) représentant les locaux du service administratif.

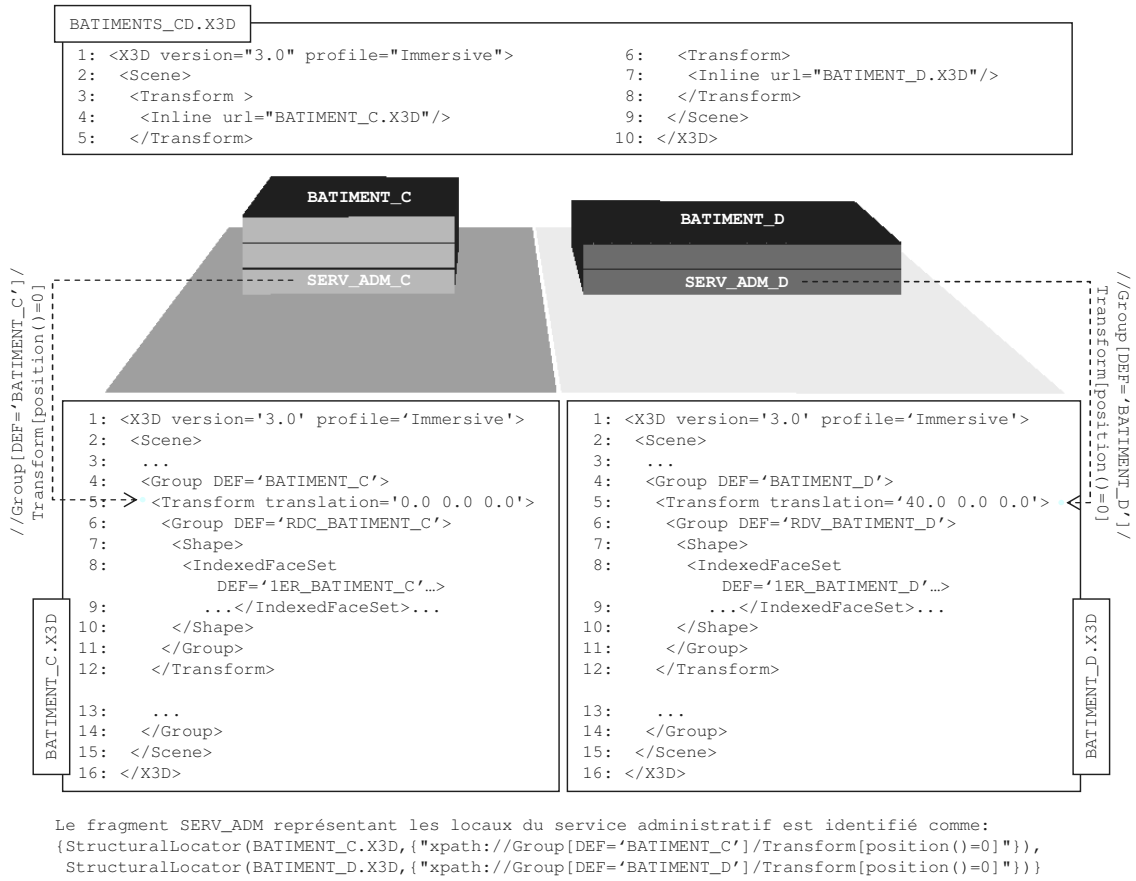


Figure 5.5 Identification des objets répartis sur plusieurs documents X3D.

Afin d'homogénéiser l'écriture d'un repère structurel inter et intra documents, nous adoptons la solution illustrée en bas de la Figure 5.5 : StructuralLocator(<Nom_Du_Fichier>, {<Repères_Locaux>}). Ainsi le repère multiple que nous avons présenté afin d'identifier les fragments 3D représentant les plafonds de bâtiment A de la *Figure 5.3* s'écrit désormais comme suit :

```
StructuralLocator(BATIMENT_A.X3D, {
     "xpath://Group[DEF='BATIMENT_A']/Transform[position()=1]",
     "xpath://Group[DEF='BATIMENT_A']/Transform[position()=3]",
     "xpath://Group[DEF='BATIMENT_A']/Transform[position()=5]"
}
```

5.2.2. Localisation spatiale

Le besoin d'utiliser des repères spatiaux pour identifier des objets dans une scène se produit, par exemple, dans le cas des documents qui ne présentent aucune structure interne ou dont la structuration ne permet pas d'isoler tous les objets qui sont pertinents pour une application particulière. C'est souvent le cas des Modèles Numériques de Terrain (MNT)

(habituellement représentés sous forme de grilles de points 3D ou sous forme de maillages triangulaires). Pour ce genre de documents, la localisation spatiale est la seule façon de localiser un objet réel (un pic ou une vallée) dans le contenu.

Parfois, même au sein des documents structurés, **la structure du document n'est pas toujours assez détaillée pour localiser convenablement le contenu ciblé**. Dans l'exemple de la *Figure 5.3*, la localisation du premier étage du bâtiment de droite n'est pas possible en utilisant uniquement les repères structurels. La localisation structurelle ne permet pas de mettre en évidence le premier étage du bâtiment de droite puisque aucun élément structurel ne lui correspond. Les deux étages et le rez-de-chaussée du bâtiment sont en effet représentés par une seule et même primitive de type `IndexedFaceSet`. Par conséquent, le même élément structurel est employé pour représenter les trois étages du bâtiment. Une solution pour identifier chaque étage est de **compléter la localisation structurelle des éléments par une localisation spatiale.** Dans notre exemple, nous pouvons isoler le premier étage dans un volume, une boîte ayant la même base que le bâtiment et une hauteur fixe.

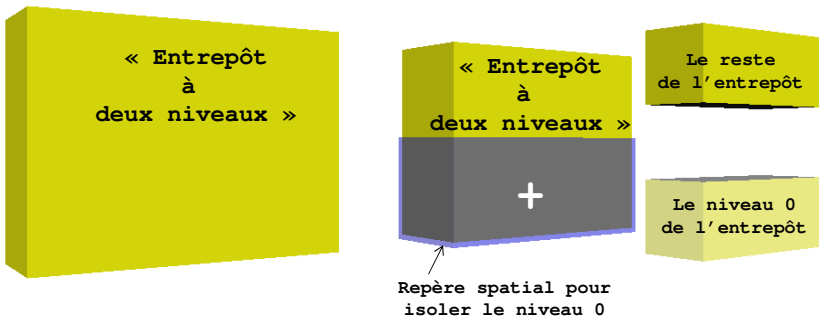


Figure 5.6 Illustration du processus de localisation spatiale.

Dans la Figure 5.6, nous illustrons le principe de localisation spatiale en prenant l'exemple d'un entrepôt à deux niveaux, modélisés par une boîte (à gauche dans la figure). Afin d'isoler le niveau 0 de l'entrepôt nous utilisons un repère spatial de type volume 3D qui matérialise la partie de l'entrepôt qui correspond au niveau 0 (au centre de la figure). La dernière étape présentée à droite dans la figure correspond à l'extraction du niveau 0 de l'entrepôt en déterminant l'intersection entre le volume indiqué par le repère spatial et la représentation initiale de l'entrepôt. Ce nouvel objet 3D peut, par la suite, être utilisé indépendamment de l'objet initial.

À présent, nous explorons l'usage de trois types de localisation spatiale : le type *sphère* (illustré dans la Figure 5.7), le type *boîte* et le type *cylindre* (illustrés dans la Figure 5.8).

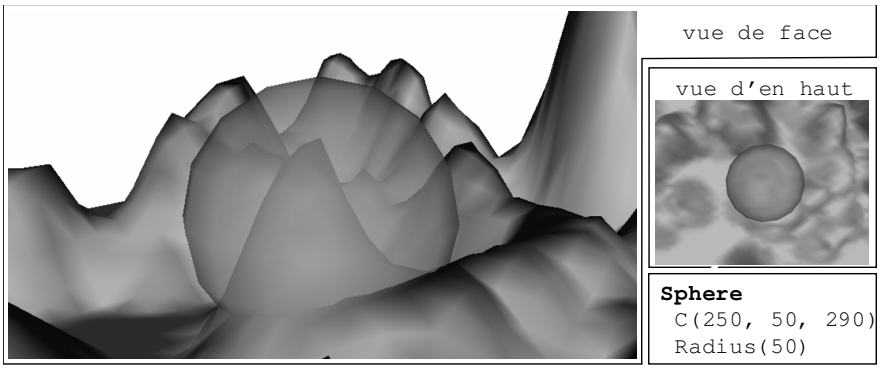
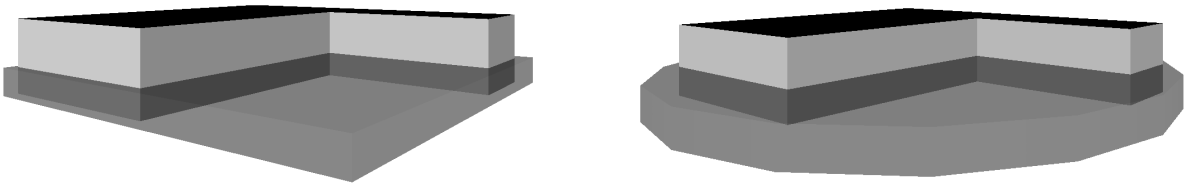


Figure 5.7 Localisation de contenu à l'aide d'un repère spatial (de type sphère) dans un MNT.

Dans la Figure 5.7, nous présentons la localisation d'une forme de relief (un sommet) à l'aide d'un repère spatial de type sphère. Le relief a été modélisé à l'aide d'une grille homogène dont les points caractérisés par leur position dans la grille et une hauteur sont utilisés afin de construire le relief par le processus d'interpolation. Le repère de localisation de type sphère utilisé, définit l'ensemble des points qui correspondent au sommet identifié. Des calculs géométriques permettent d'identifier la partie de la grille ainsi que les hauteurs associées aux points de la grille concernés par le repère spatial. Le sous-ensemble de points de la grille et leurs hauteurs sont directement associés à la définition de ce sommet et au besoin peuvent être réutilisés dans d'autre scènes, ou bien adaptés selon les besoins d'une application précise.

Dans la Figure 5.8, nous présentons de gauche à droite, la boîte et le cylindre utilisés pour identifier *spatialement* le premier étage d'un bâtiment. En plus de la définition des paramètres de chaque primitive géométrique (centre et les dimensions des caractéristiques géométriques) il est également possible d'indiquer l'orientation de la forme géométrique.



Box:
Center(60.6, 1.33, 39.7)
Width(39)
Height(3)
Depth(44)
Orientation: 0 1 0 0

Cylinder:
Center(60.6, 1.33, 39.7)
Radius(30)
Height(3)

Orientation: 0 1 0 0

Figure 5.8 Outils de localisations spatiales de type boîte et cylindre.

Ces outils de sélection géométrique définissent un fragment multimédia comme la somme des éléments géométriques se trouvant sous leur emprise (inclus spatialement dans leur volume). Les caractéristiques de chaque élément de localisation et notamment les coordonnées de son centre ainsi que l'orientation sont définies dans le repère cartésien par défaut de la scène initiale. Ainsi, les deux primitives (`Box` et `Cylinder`) présentées ci-dessus peuvent servir à définir spatialement le rez-de-chaussée du bâtiment comme étant la région se trouvant à l'intersection du bâtiment avec la boîte ou le cylindre de sélection. Lorsque, plus tard, un utilisateur veut afficher ou changer les propriétés de rez-de-chaussée, un nouvel élément est créé à partir du résultat de l'intersection. Les plates-formes qui implémentent ce genre de localisation sont tenues d'implémenter des modules capables de calculer l'intersection volumique et d'en extraire les résultats.

Les outils de localisation spatiale peuvent être utilisés conjointement afin d'accroître la précision du processus de localisation. Par exemple, dans la partie gauche de la Figure 5.9, nous présentons le même bâtiment à proximité duquel cette fois-ci il y a un arbre. Si la même boîte que celle présentée dans la Figure 5.8 est employée, lors de l'extraction des éléments composant le rez-de-chaussée du bâtiment une partie de l'arbre est également incluse. Afin d'éviter ceci et de s'approcher au plus près de la localisation réelle de l'objet à isoler, deux boîtes peuvent être utilisées ensemble afin de mieux délimiter le contour du rez-de-chaussée comme illustré dans la partie droite de la figure.

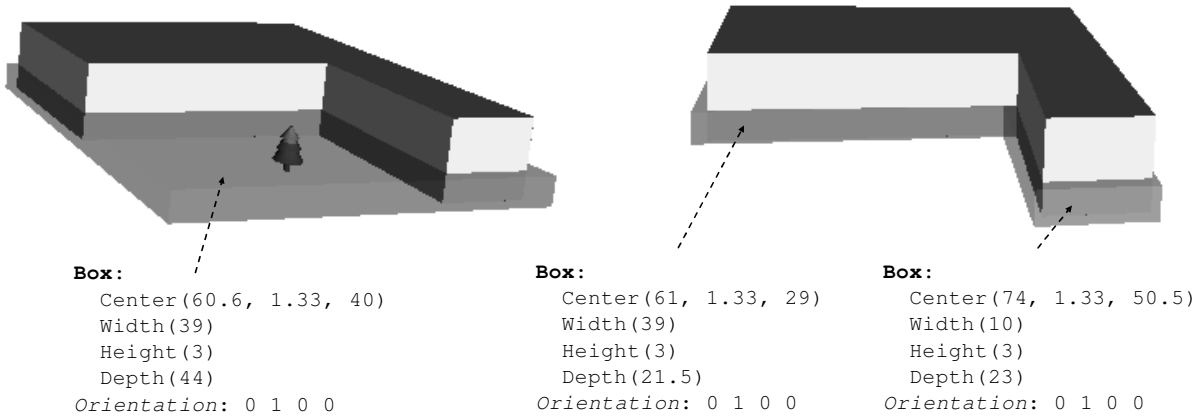


Figure 5.9 Augmenter la précision de localisation en utilisant des sélections spatiales multiples.

5.2.3. Localisation spatio-structurelle

Nous considérons un troisième type de localisation qui **combine les avantages des repères spatiaux et structurels**. Lorsque l'on ajoute un repère structurel à un repère spatial, les propriétés et les fonctions du repère spatial sont légèrement altérées. En premier lieu, les coordonnées des repères spatiaux sont définies par rapport à la boîte englobante des objets référencés par le repère structurel. En second lieu, lors de l'extraction du fragment sous l'emprise du repère spatio-structurel, uniquement les éléments conformes au repère structurel et inclus dans le volume décrit par le repère spatial sont extraits.

Par exemple, pour définir le rez-de-chaussée du bâtiment, nous pouvons employer une construction mixte qui combine :

- la localisation structurelle : `xpath://Group[@DEF='B_BUILDING']]`) et
- la localisation spatiale : `Box (Centre (0, 1.33, 0), length (1), height (1), depth (2))`.

Les coordonnées de la boîte (une localisation spatiale) sont définies relativement au système de coordonnées introduit par la localisation structurelle. La longueur, la taille et la largeur sont définies comme des distances entre le centre de la boîte et les facettes de la boîte. Même si le volume de la boîte couvre la partie de l'arbre tel qu'indiqué dans la partie gauche de la Figure 5.9, du fait que le repère structurel n'inclut pas l'arbre, aucune partie de celui-ci n'est incluse dans la définition du fragment désigné par ce repère mixte.

Nous avons été amenés à considérer ce troisième type de localisation comme un moyen de localisation à part entière pour éviter un certain nombre d'ambiguïtés lorsque les repères spatiaux et structurels sont utilisés ensemble. Par exemple, on peut imaginer qu'un fragment soit composé de deux parties distinctes dont une est référençable par un repère structurel et l'autre est référençable par un repère spatial. Dans ce cas, l'association des deux repères spatial et structurel définit un objet comme étant l'union des éléments géométriques sous l'emprise de chaque repère. En revanche, dans le cas d'un repère spatio-structurel, c'est l'intersection entre les éléments géométriques délimités par les repères qui définit un objet.

À partir de ces observations, dans la section suivante nous mettons en relief les bases de la partie du modèle 3DSEAM consacrée à la localisation de contenus 3D.

5.2.4. Vers un modèle de localisation de contenus 3D

Suite aux considérations relatives à la localisation du contenu 3D exposées dans les sections précédentes nous avons abouti à une modélisation UML qui concerne la manière dont un fragment multimédia est localisé au sein d'un document. Celle-ci est illustrée dans la Figure 5.10. Un fragment multimédia (`MMFragment`) est défini comme la somme des éléments géométriques identifiés par différents repères (`MediaLocator`) qui peuvent être : des repères structurels (`StructuralLocator`), des repères spatiaux (`SpatialLocator`), ou des repères spatio-structurels (`SpatioStructuralLocator`).

Le modèle de localisation que nous présentons couvre l'ensemble des contenus que l'on retrouve au sein d'une scène 3D (géométrie, textures – éléments 2D, etc.). Ainsi, en plus des outils de localisation spatiale 3D (`Spatial3DLocator`) que nous venons d'illustrer, nous introduisons une nouvelle classe de repères spatiaux 2D (rectangles - `RectLocator` et ellipses - `EllipseLocator`) (voir Figure 5.10). Ces repères sont utilisés principalement pour identifier des régions d'intérêts sur les textures 2D associées aux facettes des objets 3D.

Afin de pouvoir également traiter des documents ayant une dimension temporelle, nous considérons l'estampillage des repères avec des données temporelles (des instants ou des intervalles). Ces repères temporels sont utiles lorsque le contenu des documents évolue avec le temps. Ainsi, il est possible qu'à l'instant t0 un repère spatial contienne uniquement une parcelle de terrain et le même repère contienne, à l'instant t_1, une maison.

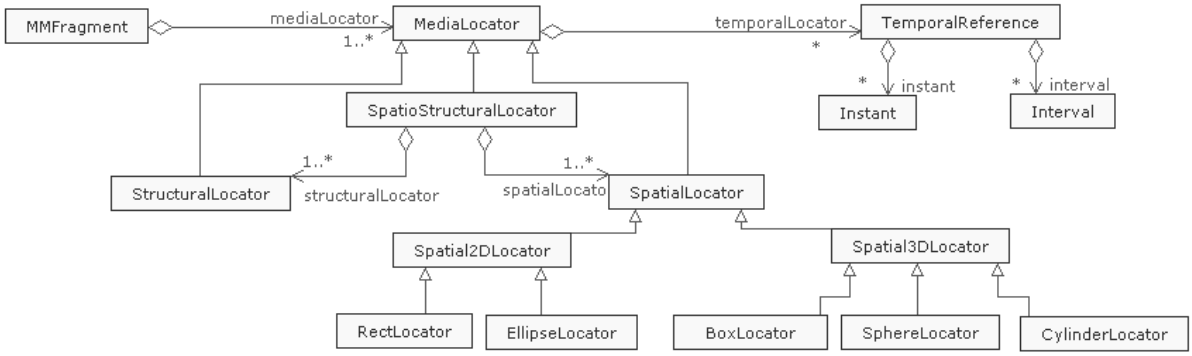


Figure 5.10 Partie du modèle 3DSEAM pour la localisation de contenus au sein d'une scène 3D.

La localisation est une étape nécessaire à la caractérisation et à l'annotation des objets 3D contenus dans une scène. L'identification par localisation structurelle et/ou spatiale des objets est une forme de base d'annotation. Elle enrichit le modèle 3D initial avec l'information au sujet de la localisation des objets à l'intérieur de la scène 3D en associant une structure sémantique à la scène concernée. Cependant, d'autres informations doivent être ajoutées afin de mieux caractériser les objets 3D. Dans les prochaines sections, nous présentons les autres dimensions du modèle générique pour l'annotation et la caractérisation de contenus 3D. Nous poursuivons en présentant la structure logique, la dimension géométrique (forme, dimension, etc.), la dimension média (type d'encodage, taille, contraintes logicielles, etc.), l'apparence (couleur dominante, etc.), la topologie et finalement la sémantique (propriétés relatives aux domaines d'application associés à la scène, ainsi que des relations entretenues avec d'autres objets).

Nous tenons à préciser que **les dimensions du modèle** que nous présentons par la suite **sont présentées sous la forme de liste de propriétés extensibles et modifiables**. Ceci est dû au fait que nous souhaitons être capable d'offrir **une solution supportant la variabilité qu'il peut y avoir au niveau des descriptions suivant les besoins d'une application particulière**. Ce

constat est extrait des observations que nous avons fait dans l'état de l'art par rapport à la multitude des propositions de caractérisation de données autant sur les aspects géométriques que sur l'apparence. Généralement, les dimensions présentées ci-dessus sont toutes caractérisées par des liste de couples : nom de propriété, valeur. Pour chacune des catégories que nous présentons par la suite, nous proposons un ensemble limité de propriétés dont le principal but est de valider notre approche de multi-caractérisation des données 3D. **Lorsque le modèle 3DSEAM et l'approche de caractérisation que nous défendons ici sont utilisés dans un cadre applicatif précis, il est envisageable d'enrichir l'ensemble des propriétés** que nous présentons afin de mieux répondre aux besoins d'une application particulière. **Un modèle de descripteurs assurant l'extensibilité de l'ensemble des propriétés de caractérisation associées au modèle 3DSEAM est présenté à la fin de ce chapitre.**

5.3. La structure logique de la scène

Nous introduisons la structure logique de la scène par le biais de la relation `LogicalStructure` au sein du modèle car nous souhaitons appuyer le fait que **le graphe de la scène ne correspond pas forcément à la structure logique associée aux éléments sémantiques de la scène**. Ici, **la structure logique rend compte de la manière dont l'organisation du monde réel a été transposée dans la scène**, tandis que l'organisation du graphe de scène est guidée par des choix d'optimisation de performance.

Plusieurs graphes de scène peuvent être utilisés pour représenter la même scène. Une illustration dans ce sens est présentée dans la Figure 5.11 où nous présentons deux graphes de scène pour la même scène 3D. Le graphe de scène de gauche rend compte de l'organisation hiérarchique des éléments géométriques (`Box`) au sein d'un même bâtiment, tandis que le graphe de scène de droite rend compte de l'organisation par niveau de la même scène.

Cette ambiguïté, qui peut apparaître au niveau de la description de la structure de document, nous motive à considérer la structure logique de la scène de manière explicite. Les utilisateurs peuvent construire une structure logique qui convient le mieux à la sémantique qu'il souhaite associer au document contenant la scène grâce aux outils de localisation mis à leur disposition. La structure logique de la scène est construite en associant chaque nouveau fragment identifié (`MM Fragment`) à son parent à travers la relation `LogicalStructure` du modèle 3DSEAM.

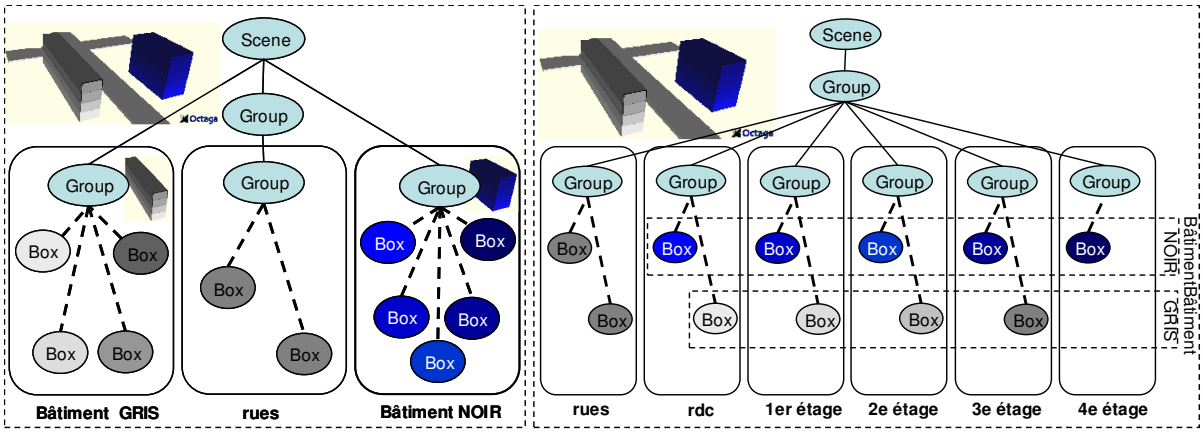


Figure 5.11 Scènes identiques encodées à l'aide de deux graphes de scène distincts.

5.4. La géométrie

Dans cette section consacrée à **la description de la géométrie** de chaque fragment 3D identifié par un ensemble de repères au sein d'une scène, nous visons le stockage d'**une abstraction de l'information géométrique contenue dans le document source**. En effet, nous ne souhaitons pas dupliquer la définition de la géométrie, mais nous voulons en retenir les principales caractéristiques. Ces caractéristiques servent à construire des critères de recherche intéressants pour les utilisateurs du modèle visant à mieux maîtriser les processus de réutilisation et d'adaptation.

Dans la partie de l'état de l'art consacrée à la recherche et à la réutilisation du contenu 3D, nous avons recensé quelques solutions qui s'intéressent à la caractérisation des objets 3D au moyen de descripteurs de formes, englobant la forme et la géométrie de ces objets. Ces solutions sont adaptées à la recherche de contenus similaires à un objet fourni par la requête. Cependant, les descripteurs qui se présentent comme une série de valeurs numériques ne peuvent pas être facilement exploités directement par un utilisateur *lambda* dans le cadre d'un système d'interrogation classique. Ici, nous souhaitons considérer également une approche de description plus proche d'un vocabulaire familier aux utilisateurs novices. Des pistes dans ce sens sont présentées dans [Attene *et al.*, 2005][Albertoni *et al.*, 2005]. Ces propositions, étudiées lors de l'état de l'art proposent d'associer des concepts appartenant à une ontologie de description de forme aux objets 3D.

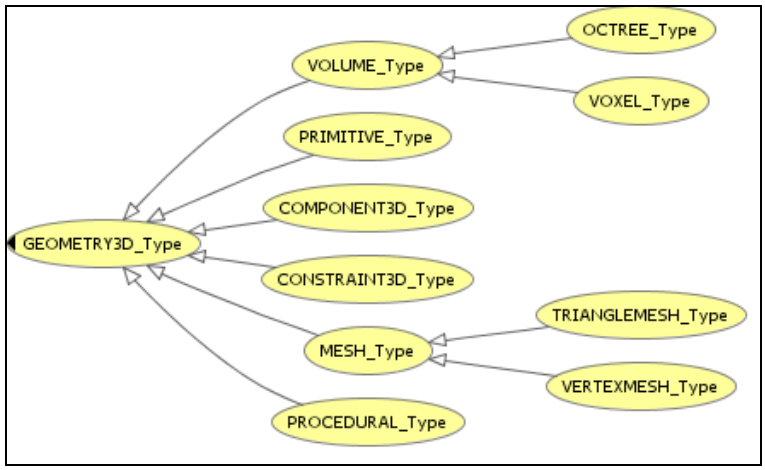


Figure 5.12 Fragment de l'ontologie décrivant la géométrie des objets 3D.

Nous souhaitons inclure dans la description des modèles, un ensemble d'informations caractérisant le type de modélisation utilisé pour définir les objets 3D. Nous souhaitons réutiliser la taxonomie réalisée dans le premier chapitre. Dans la Figure 5.12, nous présentons une partie de l'ontologie relative à la caractérisation de la géométrie.

D'autres informations relatives aux éléments géométriques utilisées (points, surfaces, volumes, primitives) et au nombre d'éléments sont également prises en compte. Ces éléments sont importants car, dans la plupart des cas, ils permettent d'identifier la complexité et le niveau de détail des modèles concernés.

Dans le Tableau 1, nous énumérons les principales propriétés retenues pour caractériser la géométrie d'un objet 3D. Chaque propriété est introduite par son nom, sa définition et le type de valeurs supportées. Cette liste n'est pas exhaustive et, suivant le domaine d'application, d'autres propriétés sont incluses. Nous retenons ici uniquement le fait que si des propriétés spécifiques sont stockées, il est nécessaire que leur nom et leur définition soient clairement spécifiés dans un

document indiquant le schéma de description géométrique. La clarté de la description des propriétés peut être acquise en utilisant les ontologies.

Nom de propriété	Définition
modelisationType	approche de modélisation pour la conception de l'objet 3D
nbOfVertex	nombre de sommets
nbOfVertices	nombre d'arêtes
nbOfPrimitives	nombre de primitives
area	surface des facettes
volume	volume de la forme
bbox	boîte englobante définie par les coordonnées 3D de son centre, de la longueur, de la hauteur et de la profondeur
shape3D	descripteur MPEG-7 proposé dans [Zaharia *et al.*, 2002]

Tableau 1 Propriétés géométriques retenues dans le modèle 3DSEAM.

5.5. Le profil média

Le profil média rassemble les **propriétés de la ressource média associée aux fragments de contenus (audio, 2D ou 3D)** : ses propriétés (taille de document, temps de téléchargement, etc.), les pré-requis logiciels (existence de *plugins*, etc.), les pré-requis matériels (puissance de calcul requise, mémoire disponible, accélérateur graphique, etc.), contraintes liées au contexte de diffusion (endroit calme pour faciliter la diffusion du message sonore inclus dans la scène), informations sur le créateur et les droits de diffusion et de reproduction. En plus de la taille du document, pour les fragments de type 3D, nous retenons séparément la taille de l'espace mémoire nécessaire pour stocker la géométrie et l'apparence des fragments 3D, afin de conférer aux utilisateurs de notre solution de description plus de détails sur la qualité de modélisation de l'objet.

Ces informations servent à mieux connaître les spécificités média des fragments. Elles peuvent aider un concepteur de scènes 3D lorsqu'il souhaite retrouver un objet libre de droits de diffusion et/ou nécessitant un *plugin* spécifique pour un bon rendu visuel.

Nous avons organisé les données du profil média en plusieurs catégories :

- dimensions de la représentation du fragment (*Size*),
- requis matériels et logiciels pour un déploiement (*Requirements*),
- informations sur le propriétaire et les droits de diffusion (*ProprietaryInformation*).

Dans la suite, nous passons en revue chacune de ces catégories. Dans le Tableau 2 nous présentons la liste de propriétés que nous avons retenues pour caractériser les dimensions matérielles de la représentation du fragment.

Size	
Nom de propriété	**Définition**
totalSize	taille du fragment
appearenceSize	taille de la représentation de l'apparence
geometrySize	taille de la représentation de la géométrie

Tableau 2 Caractéristiques de taille dans le profil média.

Le Tableau 3 recueille l'ensemble des requis matériels et logiciels dont l'importance justifie l'intégration au sein de cette première version du modèle 3DSEAM.

Requirements	
Nom de propriété	**Définition**
requiredPlugins	liste deS plugins capables de jouer le contenu
requiredCPU	vitesse minimum du CPU
requiredMem	Mémoire disponible nécessaire
required3DGraphicsCard	modèle de carte 3D requis
minimumScreenHeight	hauteur minimum de l'écran
minimumScreenWidth	hauteur maximale de l'écran
mouseRequired	souris classique nécessaire
mouse3ButtonsRequired	souris à trois boutons nécessaire
keyboardRequired	clavier nécessaire pour interaction
specificInteractionDevice	indique un dispositif spécifique d'accès (souris 3D, gant, etc.)

Tableau 3 Requis matériels et logiciels pour la diffusion du fragment média.

Dans le Tableau 4 nous présentons l'ensemble deS données sur le propriétaire et les droits de diffusion associés à un fragment multimédia.

Proprietary Informations	
Nom de propriété	**Définition**
url	URL du document contenant le fragment
name	nom du document
author	nom de l'auteur
coauthor	noms des éventuels co-auteurs
owner	nom du propriétaire du fragment
copyright	type de copyright
reuse	réutilisation de fragment autorisée
costForReuse	coût de réutilisation
cost	coût de diffusion

Tableau 4 Informations concernant le propriétaire et les droits de diffusion des contenus média.

La section suivante s'intéresse à la manière dont l'apparence est décrite dans le modèle 3DSEAM.

5.6. L'apparence

Afin de caractériser l'apparence d'un objet 3D nous considérons qu'il est nécessaire d'avoir des informations sur le type de texture utilisée :

- *texture de type image ou vidéo*. Les textures de type image ou vidéo définissent un objet média 2D qui est appliqué à la surface d'un objet 3D ;
- *texture de type motif*. Les textures de type motif associent une couleur et un moyen de recouvrement de la surface avec le motif choisi ;
- *texture définie au niveau pixel*. Dans une texture au niveau pixel, la couleur de chaque pixel de la surface de l'objet est définie explicitement ;

- *texture paramétrique*. Pour une texture paramétrique, la couleur associée à un point sur l'objet dépend éventuellement de la position de ce point dans l'espace et des paramètres externes fournis par le concepteur de la texture ;
- *multi-texture*. Une multi-texture correspond à la superposition de plusieurs textures sur le même objet. La définition d'une multi-texture inclut en plus de la définition de chaque texture atomique, les opérateurs qui décident de la manière dont les couleurs présentes sur les différentes couches (textures atomiques) sont combinées afin d'obtenir les couleurs finales.

Chaque type de texture a des traits qui lui sont propres. Nous n'entrons pas dans le détail de la description de l'apparence de l'objet 3D, mais **nous choisissons de retenir les principales caractéristiques telles que la couleur dominante, l'uniformité de la texture, etc.** Des caractéristiques plus poussées sont considérées dans la littérature de spécialité concernant les média 2D telles que les contours [Cohen *et al.*, 1993], la distribution des couleurs [Carson *et al.*, 1999], etc. Ce type de caractéristique peut être intéressant lorsque l'on recherche par exemple un objet en fournissant une image 2D de l'objet. Des mesures de similarités s'appuyant sur des critères tels que les contours ou les diagrammes de couleurs, permettraient de repérer les objets satisfaisants la requête. Cependant, comme nous l'avons déjà précisé lors de la caractérisation de la géométrie, nous avons conçu le modèle de caractérisation de telle façon à permettre la personnalisation de la liste des propriétés décrivant l'apparence. Dans notre proposition, nous retenons uniquement un ensemble minimal de propriétés permettant de valider notre approche.

Le Tableau 5 illustre quelques propriétés d'apparence que nous considérons dans cette version de base du modèle 3DSEAM.

Nom de propriété	Définition
textureType	type de texture
dominantColor	couleur dominante de la texture
density	densité de pixels
resolutionX	largeur de la texture
resolutionY	hauteur de la texture

Tableau 5 Propriétés décrivant l'apparence des scènes 3D.

Toutefois, **si la texture associée à un objet est de type image ou vidéo, nous poussons la caractérisation de l'apparence en considérant le fragment multimédia qui correspond spécifiquement au document image ou vidéo contenant la texture**. Comme l'indique la Figure 5.13, nous pouvons associer à un élément décrivant l'apparence (`Appearance`) un élément décrivant le fragment 2D (`Content2D` – image ou vidéo) qui contient la texture. Ainsi, l'ensemble des caractéristiques de fragment 2D, telles que le profil média, est accessible depuis les propriétés d'apparence d'un objet 3D. Ceci est important, lorsque, par exemple, nous disposons de deux données 3D similaires, et que nous souhaitons sélectionner celle dont la taille de la texture est moindre. Puisqu'à travers le lien reliant la classe `Appearance` de la classe `Content2D` nous pouvons récupérer les tailles sur disque des textures, nous sommes en mesure de choisir l'objet 3D dont l'apparence est la plus légère.

Par ailleurs, ce choix de modélisation permet de **réutiliser les caractéristiques d'une image ou d'une vidéo dès lors que la même texture est réutilisée sur plusieurs objets 3D.** Dans le cas d'une multi-texture à base d'images ou vidéo, la même approche peut être utilisée en associant l'ensemble de textures atomiques en tant que fragments 2D, au descripteur d'apparence de la donnée 3D en question.

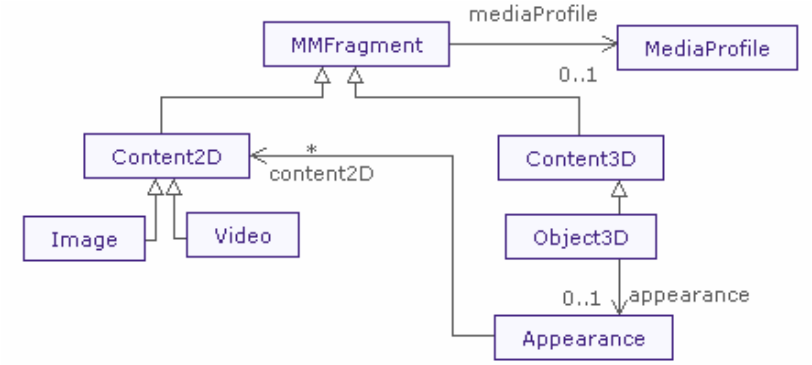


Figure 5.13 Partie du modèle consacrée à la caractérisation de l'apparence d'un objet 3D.

5.7. La topologie

Une relation topologique correspond à une relation spatiale invariable quel que soit le point de vue à partir duquel on évalue cette relation spatiale [Egenhofer *et al.*, 1990]. Dans la littérature lorsque l'on s'intéresse au terme relation topologique 3D les auteurs identifient plusieurs classes de relations topologiques en fonction des types d'éléments que l'on veut mettre en relation [Egenhofer *et al.*, 1995][Clementini *et al.*, 1997][Zlatanova, 2000]. Ainsi, on trouve plusieurs relations topologiques 3D, organisées en classes selon les types des éléments (des points, des lignes, des surfaces et des volumes) que les relations mettent en correspondance. Nous nous intéressons ici plus particulièrement aux relations topologiques dont les deux opérandes sont des volumes (voir Figure 5.14).

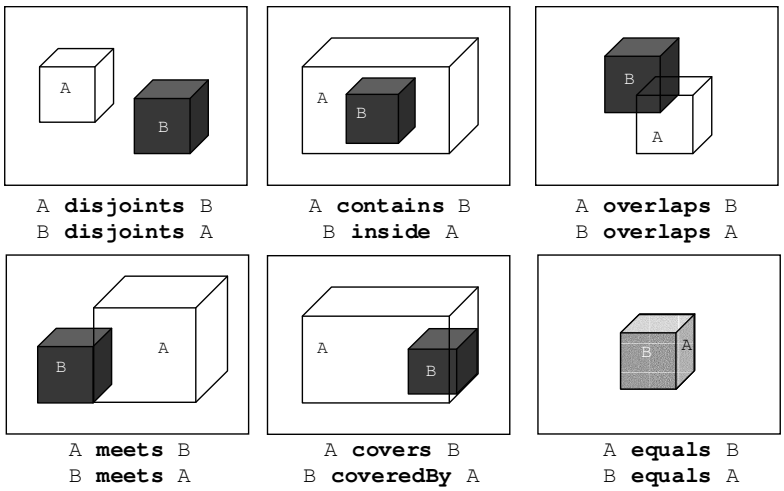
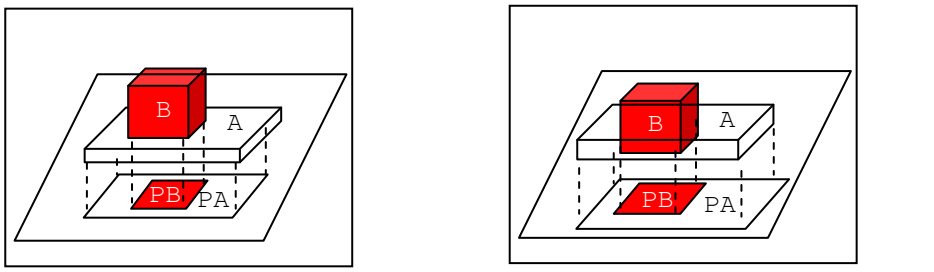


Figure 5.14 Relations topologiques entre objets 3D selon [Egenhofer et al.*, 1990].*

Les relations topologiques que nous considérons dans notre modèle servent à compléter les informations locales spécifiques à chaque objet (comme le profil média, l'apparence et les caractéristiques géométriques) avec des connaissances relatives au voisinage d'une donnée 3D. Ces informations sont, à notre avis, d'une grande importance lorsque l'on veut être capable d'analyser de manière approfondie les propriétés d'une scène 3D dans son ensemble. Les relations topologiques offrent la possibilité de caractériser l'organisation spatiale d'une scène, et par la suite d'en faire l'analyse. Par rapport aux relations topologiques illustrées dans la Figure 5.14, **nous souhaitons introduire une relation supplémentaire afin de mieux caractériser les**

situations dans lesquelles deux objets se chevauchent (situation décrite par l'opérateur *overlaps)* **ou se touchent** (situation décrite par l'opérateur *meets*)**.**

Ainsi, si deux objets A et B se chevauchent ou se touchent et que, de plus, l'objet A contient intégralement une section transversale de l'objet B, nous introduisons une nouvelle relation topologique que nous appelons : *containsSection* (voir Figure 5.15).



A **containsSection** B =
{ A **meets** B, PA **contains** PB}

A **containsSection** B =
{ A **overlaps** B, PA **contains** PB}

Figure 5.15 Illustration de la relation topologique containsSection.

Ce type de relation peut être important dans les situations où l'on s'intéresse à des objets greffés (par exemple, des arbres ou des bâtiments) **sur une surface plane *(par exemple, un modèle numérique du terrain ou plus simplement une boîte modélisant une parcelle de terrain sur laquelle sont disposés des bâtiments).*** Si, pour ce cas de figure, nous essayons de trouver une relation topologique dans celles proposées dans la Figure 5.14, les seules qui peuvent convenir sont *meets* ou *overlaps*. Ces relations correspondent respectivement à la situation où la base du bâtiment correspond à la face supérieure de la boîte ou du modèle numérique de terrain modélisant la parcelle (*meets*), et à la situation où la base de l'arbre ou du bâtiment se trouve enfouie dans la boîte modélisant la parcelle (*overlaps*). Cependant, ces deux relations ne rendent pas directement compte du fait que les éventuels arbres ou bâtiments sont situés à l'intérieur de la région spatiale définie par l'objet parcelle. Effectivement, une relation *meets* ne différencie si deux objets sont adjacents horizontalement *ou* verticalement. La relation *containsSection* que nous proposons peut être calculée relativement facilement. En plus des relations topologiques entre volumes, nous considérons les relations topologiques entre les projections de ces volumes sur un plan (voir Figure 5.15). En effet, la relation *containsSection* peut être déduite en considérant les couples de volumes qui sont en relation soit de type *meets*, soit de type *overlaps* et dont les projections sur un plan donné sont en relation (topologique 2D) *contains*.

La plupart des relations que nous considérons ici sont extraites automatiquement à partir des informations de localisation associées à chaque objet 3D. En effet, chaque objet 3D est décrit par un repère qui peut être de nature spatiale, auquel cas il est possible d'en déduire les frontières et de s'en servir pour calculer le positionnement de cet objet par rapport à d'autres objets de la scène. À défaut, la description de la géométrie des objets 3D contient des informations sur la boîte englobante de l'objet. Cette information peut également servir pour mettre l'objet en rapport avec ses voisins.

Cependant, nous n'imposons pas le fait que des relations topologiques soient calculées entre tous les objets d'une scène. **Les propositions faites ici constituent un support pour l'annotation exploitable au besoin pour caractériser une scène. C'est à la guise de l'utilisateur, ou de l'application réalisant la caractérisation des scènes, de renseigner ou bien de demander l'extraction automatique des propriétés relatives à une donnée 3D.** Pour ce qui est d'une relation topologique, il est souhaitable que la relation ne soit pas saisie directement par l'utilisateur, mais qu'elle soit calculée automatiquement par un module

spécifique à la demande de l'utilisateur, afin de réduire les risques d'erreur dans les analyses futures.

En plus de ces relations spatiales, il est fortement envisageable que les utilisateurs formulent des requêtes spatiales qui concernent les positions relatives de certains objets par rapport aux autres (par exemple, sélectionner les maisons du côté gauche de la route). Ce type de relation est dépendant du point de vue de l'utilisateur sur la scène. Ainsi, il est inutile de les pré-calculer et de les stocker au sein du modèle car elles ne sont pas invariantes. Toutefois, le modèle contient assez d'informations, et notamment la partie de caractérisation de la géométrie et la partie de localisation, pour évaluer ce genre de relations à la volée. Dans le chapitre 0, nous illustrons l'utilisation de ce type de relation dans le cadre d'une plate-forme construite autour du modèle 3DSEAM et dédiée aux Systèmes d'Information Géographique 3D.

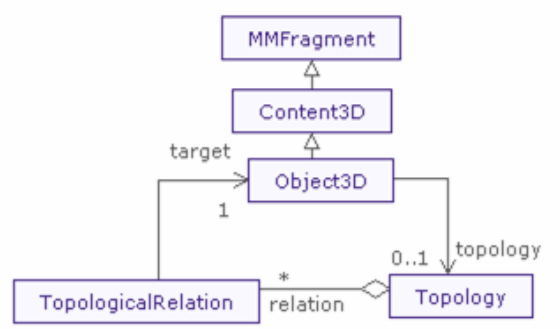


Figure 5.16 Modélisation d'une relation topologique dans 3DSEAM.

Dans notre modèle, nous introduisons une nouvelle classe `TopologicalRelation` qui permet de retenir le type de relation topologique existant entre l'objet pour lequel on renseigne la topologie et l'objet cible. Une association entre la classe `TopologicalRelation` et la classe `Object3D` permet de définir la cible de la relation. Une modélisation UML est présentée dans la Figure 5.16.

5.8. La sémantique

Dans le modèle 3DSEAM, la sémantique se retrouve à tous les niveaux de la caractérisation d'une donnée 3D. Ainsi, nous avons identifié quatre catégories qui regroupent respectivement les connaissances géométriques, les connaissances liées à l'apparence, les connaissances topologiques et les connaissances média. Chaque catégorie regroupe des propriétés matérialisant les différentes connaissances que l'on peut avoir sur une donnée 3D. Chacune de ces catégories est représentée par une classe dans le modèle. Cependant, les connaissances que l'on associe à une donnée 3D sont également issues des domaines d'application relatifs aux objets du monde réel modélisés par les données 3D.

5.8.1. Niveaux de caractérisation sémantique

À une donnée multimédia, il est possible d'**associer de l'information relative à l'objet réel représenté par la donnée**. Si par exemple, la donnée multimédia représente la Tour Eiffel, des propriétés issues du monde réel (hauteur réelle, poids de la structure, propriétaire, …) pourront être associées à la donnée et éventuellement utilisées par des scripts ou lors des interrogations effectuées sur les scènes contenant la donnée respective.

Cette dimension recueille l'ensemble des descriptions portant sur les propriétés d'une donnée :

- en rapport avec la scène la contenant – **sémantique du fragment multimédia** ou
- en rapport avec un domaine d'application spécifique – **sémantique de l'entité représentée**.

Par exemple, une propriété sémantique liée au fragment peut retenir le rôle que la donnée joue dans la scène – élément central, ou bien le comportement de l'objet vis-à-vis d'autres éléments de la scène. La sémantique d'une entité peut contenir des informations relatives au domaine géographique telles que les coordonnées (latitude, longitude) ou l'altitude des données. Ces informations peuvent servir également à définir des catégories d'objets partageant les mêmes propriétés sémantiques.

Nous identifions trois **niveaux sémantiques d'une donnée 3D** :

- **le niveau local spécifique à la scène** qui contient la donnée – correspond à une instance bien précise de l'entité (la Tour Eiffel est au centre de la scène) ;
- **le niveau applicatif** – correspond aux propriétés de l'entité du monde réel relatives à un domaine d'application (par exemple, pour une application liée à la métallurgie on retiendra que la Tour Eiffel est construite en *Fer puddlé* provenant des aciéries de Pompey en Lorraine) ;
- **le niveau général** – correspond à tout ce qui reste quand l'objet est sorti du contexte applicatif de la scène (la taille réelle de la Tour Eiffel est de 324 m).

Dans le cas de la sémantique relative à la scène qui contient le fragment, le lien entre la donnée et ses propriétés sémantiques est réalisé sans intermédiaire. Dans le cas où la sémantique concerne les propriétés de l'élément du monde représenté par la scène, nous mettons en relation la donnée et sa sémantique en plusieurs étapes :

a) nous introduisons cet élément dans le modèle en tant qu'instance de la classe `Entity` s'il ne l'est pas déjà ;

b) nous associons l'entité au fragment multimédia ;

c) nous mettons en relation l'entité avec l'ensemble des propriétés sémantiques caractérisant l'élément et ses profils sémantiques. Un profil sémantique sert à regrouper l'ensemble des propriétés et relations sémantiques en rapport avec un domaine d'application spécifique. Si l'entité est déjà présente dans le système, la troisième étape est optionnelle car le fragment multimédia hérite de toutes les propriétés sémantiques associées auparavant avec cette entité.

La sémantique applicative ou générale liée aux fragments multimédia est accessible en deux étapes :

- l'obtention de l'entité associée au fragment multimédia,
- l'obtention de la sémantique (dans le cadre générale) ou d'un profil sémantique (dans le cadre d'une application spécifique) associés à l'entité.

Nous avons fait ce choix afin de faciliter le maintien de la cohérence. La sémantique peut changer (ajout d'une nouvelle information, suppression des données obsolètes) sans toucher au fragment multimédia et *vice-versa*.

De plus, **plusieurs fragments multimédia peuvent représenter la même entité du monde**. Ainsi, un autre bénéfice de ce découplage entre la sémantique et le fragment multimédia à travers la classe `Entité` est de **centraliser les descriptions sémantiques des données multimédia équivalentes d'un point de vue sémantique**. Un deuxième rôle tout aussi important est qu'elle permet de **rendre indépendante l'évolution de la sémantique de l'évolution des fragments multimédia**.

5.8.2. Les profils sémantiques

La notion de profil sémantique, que nous avons brièvement introduit dans la sous-section précédente, est à la base de l'organisation des informations sémantiques dans 3DSEAM. Que ce soit pour les informations relatives à la sémantique générale, applicative ou locale, **un profil sémantique regroupe un ensemble de propriétés ou de relations** qui décrivent respectivement **:**

- une entité par rapport à un domaine ou une application spécifique,
- un fragment multimédia par rapport aux catégories d'information sémantique relative à la scène.

La notion de profil impose l'existence d'une structuration de l'information, propriété intéressante pour la gestion a posteriori de l'information.

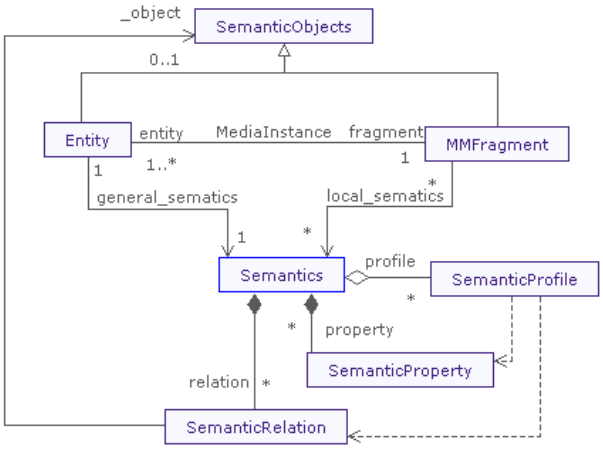


Figure 5.17 La dimension sémantique du modèle 3DSEAM.

Dans la Figure 5.17, nous présentons la formalisation de ces notions dans un diagramme de classes UML [Booch *et al.*, 1998]. Nous introduisons des relations d'agrégation entre les classes `Semantics` et `SemanticProfile` afin de tenir compte de la possibilité de réutiliser le même profil sémantique pour représenter la sémantique commune à plusieurs entités du monde ou à plusieurs fragments multimédia. La relation entre les classes `SemanticProfile` et `SemanticProperty` laisse entrevoir à la fois la possibilité de retrouver la même propriété sémantique dans plusieurs profils. Les relations sémantiques peuvent être définies soit par rapport à une entité (objet du monde réel), soit par rapport à un fragment multimédia (objet de la scène 3D). Afin de faciliter la description du modèle, nous introduisons une classe (`SemanticObjects`) représentant les objets du monde réel (ou les entités) et les fragments.

Il est souhaitable que toute information sémantique associée à une entité (si l'information est de niveau général) ou à un fragment (si l'information est locale) soit liée à un profil (`SemanticProfile`). Ceci est justifié par le fait qu'un profil est intrinsèquement lié à un domaine d'application ou à un type de caractérisation spécifique ce qui permet de mieux cerner la nature

et le sens de l'information. Un profil sémantique constitue en effet une vue sur l'ensemble des propriétés (`SemanticProperty`) et relations sémantiques (`SemanticRelation`) d'une entité. Ainsi, il est possible que deux ou plusieurs profils sémantiques partagent les mêmes propriétés. Nous ne souhaitons pas maintenir des doublons dans la description sémantique car cela demande un gros effort de maintenance de la cohérence lors des éventuelles mises à jour ou suppressions. Afin d'éviter ce type de problèmes, nous attachons l'ensemble des propriétés (rôle `property` de la classe `Semantics`) au niveau de l'objet représentant la sémantique de l'entité ou du fragment multimédia et non pas au niveau de chaque profil indépendamment.

La définition d'un profil sémantique est réalisée en indiquant le nom du profil, la liste des propriétés et éventuellement les noms des domaines auquel le profil peut être associé. Chaque propriété est définie par son nom, une description de ce qu'elle représente et éventuellement un nom étendu correspondant à un terme dans une ontologie de domaine. Ce nom étendu permet de donner plus de sens à la valeur contenue dans la propriété.

Le type d'association que nous voulons construire ici entre la sémantique d'un objet et un profil sémantique trouve quelques similitudes avec le concept d'interface dans les langages de programmation orientée objet. De la même manière qu'une interface représente la garantie que les instances des classes l'implémentant exhibent un certain type de comportement, l'association d'un profil à la sémantique est une garantie que les informations dénotées par le profil se retrouvent dans la liste de propriétés de l'objet représentant la sémantique d'une entité ou d'un fragment. Cependant, dans un objet représentant la sémantique, nous pouvons également trouver des propriétés sémantiques qui sont définies indépendamment d'un profil quelconque.

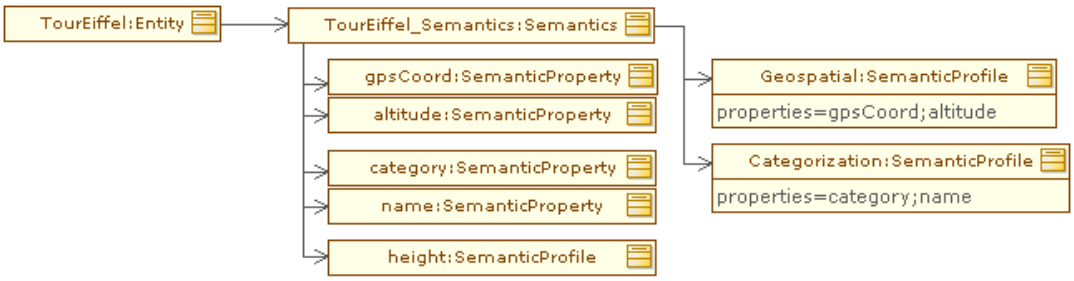


Figure 5.18 Exemple d'instanciation de la sémantique d'un objet représentant la Tour Eiffel.

Afin d'illustrer la mise en œuvre de ces concepts nous proposons dans la Figure 5.18 un exemple de description de la sémantique de l'entité `TourEiffel` correspondant à la Tour Eiffel. L'objet `TourEiffel_Semantics` contient l'ensemble des propriétés sémantiques que l'on a associé à cette entité. Dans cet exemple nous considérons cinq propriétés :

- les coordonnées GPS[36] de l'objet réel (`gpsCoord`),
- l'altitude à laquelle l'objet réel se trouve par rapport au niveau de la mer (`altitude`),
- la catégorie (`category`) définie par rapport à une taxonomie des objets réels (bâtiment, arbre, …),
- le nom de l'objet réel (`name`, dans ce cas « La Tour Eiffel ») et
- la hauteur en mètres de l'objet réel (`height`).

[36] Dans ce travail nous désignons les coordonnées GPS comme étant la latitude et la longitude précises d'un objet. Cet abus de langage a été réalisé afin de faciliter l'exposé de nos idées. La notion de coordonnées GPS comporte une inhérente imprécision, que nous ne traitons pas ici, due à la qualité du signal et au nombre de satellites utilisés pour son calcul.

Ces propriétés couvrent les deux profils sémantiques (`Geospatial` et `Categorization`) que nous présentons sur la droite de la Figure 5.18. Le profil `Geospatial` concerne les propriétés `gpsCoord` et `altitude`. Le profil `Categorization` concerne les propriétés `category` et `name`. Il y a également une propriété `height` qui ne fait partie d'aucun profil à présent.

5.9. Un modèle pour l'extensibilité des descripteurs

Tout au long de l'exposé des caractéristiques du modèle 3DSEAM, nous avons insisté sur le fait que le modèle doit être facilement extensible pour répondre aux besoins du plus grand nombre. **Ce que nous défendons avec ce modèle, c'est la possibilité d'offrir un moyen de caractérisation et d'analyse des données 3D sous tous leurs aspects.**

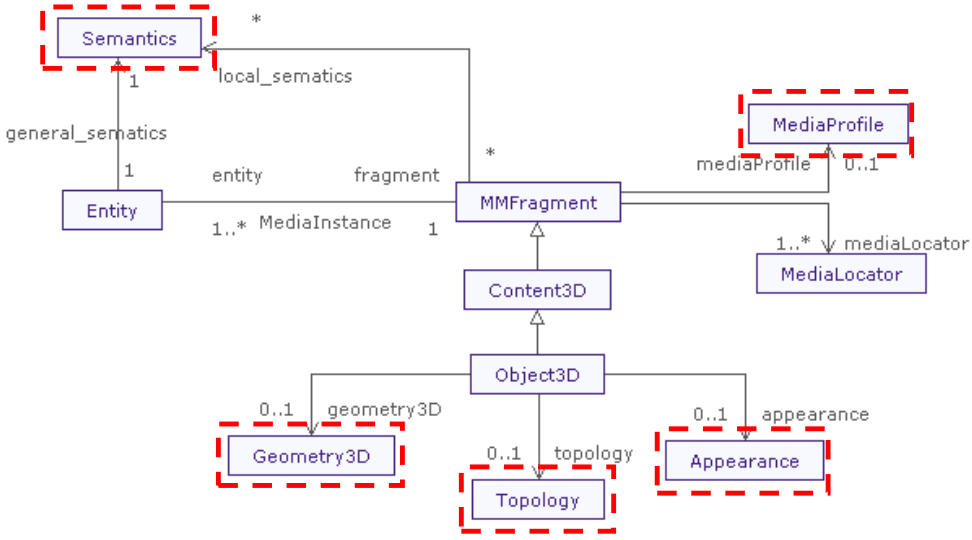


Figure 5.19 Vue partielle du modèle avec les classes à liste de descripteurs évolutive en pointillés.

Les descripteurs que nous avons retenus dans cette version du modèle couvrent un ensemble minimal de propriétés pour chaque dimension du contenu 3D que nous avons identifiée. Cependant, dans la mesure où certains utilisateurs voudraient ajouter leurs propres caractéristiques, nous avons décidé de **donner la possibilité d'étendre la liste des propriétés pour chacune des catégories** visées (géométrie, topologie, apparence, profil média, sémantique). Dans la Figure 5.19, nous présentons une vue partielle du modèle qui reprend les principales classes du modèle, celles dont la liste de descripteurs peut varier en fonction des besoins applicatifs (encadrées de traits en pointillés).

Afin de maîtriser le processus d'extensibilité sans limiter les libertés en termes d'annotations, nous proposons un modèle de descripteurs utilisés pour la caractérisation des différentes dimensions d'une donnée 3D. Ce modèle recueille les intitulés de la totalité des propriétés et relations utilisées pour décrire les différentes dimensions d'une donnée 3D.

La Figure 5.20 présente le diagramme UML du modèle des descripteurs de 3DSEAM. Ce modèle décrit l'ensemble des propriétés et relations utilisées afin de décrire individuellement chaque partie potentiellement extensible du modèle 3DSEAM. Ainsi, on retrouve la classe `DimensionDescription` qui correspond à la description d'une dimension du contenu. Elle est déclinée en cinq sous-classes, chacune d'entre elles étant dédiée à une partie extensible du modèle : de gauche à droite, `GeometryDescription`, `AppearanceDescription`, `MediaProfileDescription`, `SemanticDescription`, `TopologicalDescription`.

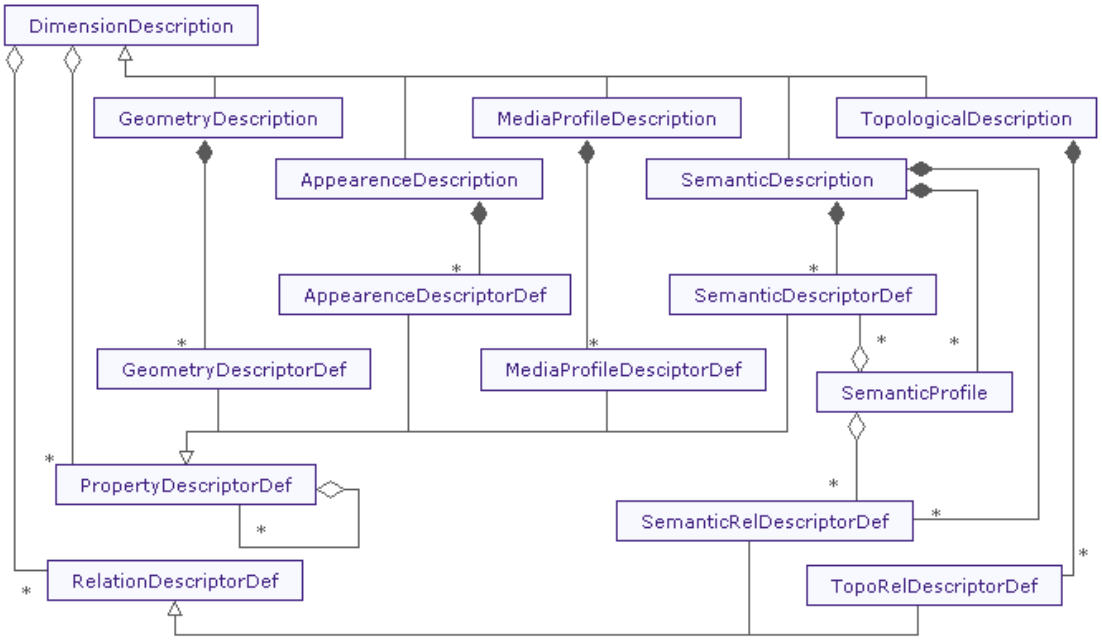


Figure 5.20 Modèle de descripteurs 3DSEAM.

La classe de description d'une dimension du contenu regroupe un ensemble de descripteurs de propriétés (`PropertyDescriptorDef`) et d'éventuelles relations (`RelationDescriptorDef`). Ce regroupement est réalisé à l'aide de relations qui sont ensuite déclinées au niveau de chacune des dimensions traitées.

La définition des descripteurs est spécialisée pour chaque catégorie en spécifiant en autant de sous-classes, la classe `PropertyDescriptorDef`. Nous mentionnons que la dimension topologique ne contient pas de descripteurs de propriétés car au niveau de cette dimension nous considérons uniquement des relations topologiques. La dimension sémantique, en plus d'un ensemble de descripteur de propriétés et de relations, accueille également la liste des profils sémantiques instanciés au sein du modèle.

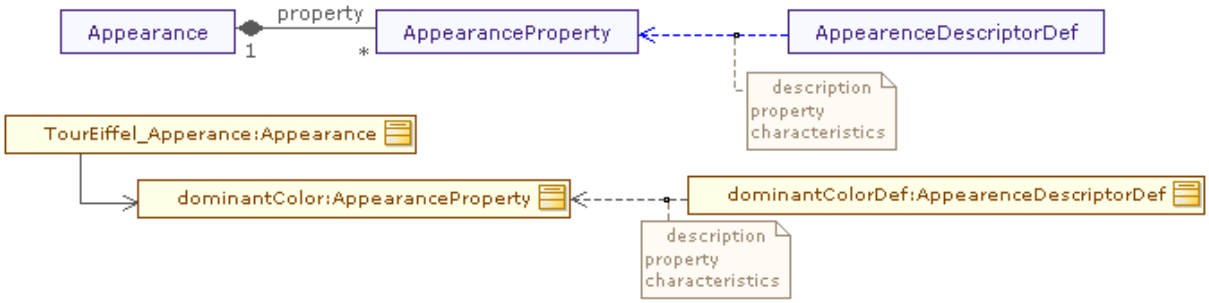


Figure 5.21 Lien entre les propriétés de classe du modèle 3DSEAM et le modèle de descripteurs.

Avant de détailler la définition d'un descripteur nous présentons dans la Figure 5.21 le lien qui existe entre les instances du modèle 3DSEAM et les instances du modèle de descripteurs en considérant comme exemple la classe `Apparence`. La classe `Apparence` est caractérisée par un ensemble de propriétés d'apparence (`AppearanceProperty`) dont la définition de chacune (`AppearanceDescriptionDef`) est répertoriée dans le modèle de descripteurs. Ainsi, lorsque l'on s'intéresse par exemple à la couleur dominante de la Tour Eiffel, sa définition est disponible dans l'instance `dominantColorDef` de la classe `AppearanceDescriptorDef` du modèle de descripteurs.

Généralement, la définition d'un descripteur comporte le nom de la propriété ou de la relation qu'il représente, le type de valeurs supportées, une description textuelle qui présente sa nature et son utilité, et un lien avec une ontologie dans laquelle la propriété est définie (à travers l'utilisation des espaces de noms). Ces deux dernières informations peuvent virtuellement servir à mettre en place des processus d'inférence automatique si la plate-forme implémentant le modèle met en œuvre des systèmes d'interprétation et d'évaluation de propriétés de manière automatique. Le rattachement sémantique des propriétés à une ontologie permet aux machines d'interpréter par elles-mêmes le sens des données saisies. Ceci va dans le sens des efforts de la communauté du Web sémantique pour une automatisation du traitement de l'information. Lorsque l'on définit une propriété dont la valeur n'est pas atomique, il est possible de décrire les sous-propriétés comme précédemment. Pour les propriétés complexes dont on ne définit pas les sous-propriétés, il est impossible de les référencer directement au sein du modèle.

La définition d'un descripteur de relation suit les mêmes règles que celles spécifiées pour une propriété à l'exception du fait que le type de la valeur est remplacé par un attribut indiquant une classe du modèle 3DSEAM. En effet, lorsque l'on définit une relation topologique on indique le nom de la relation et l'objet auquel se rapporte l'objet source. Cette information indiquant le type d'instance qui rentre dans la relation est importante pour pouvoir ensuite exploiter cette connaissance lorsque l'on veut naviguer au sein du modèle en s'appuyant sur les relations (topologiques ou sémantiques) définies entre objets.

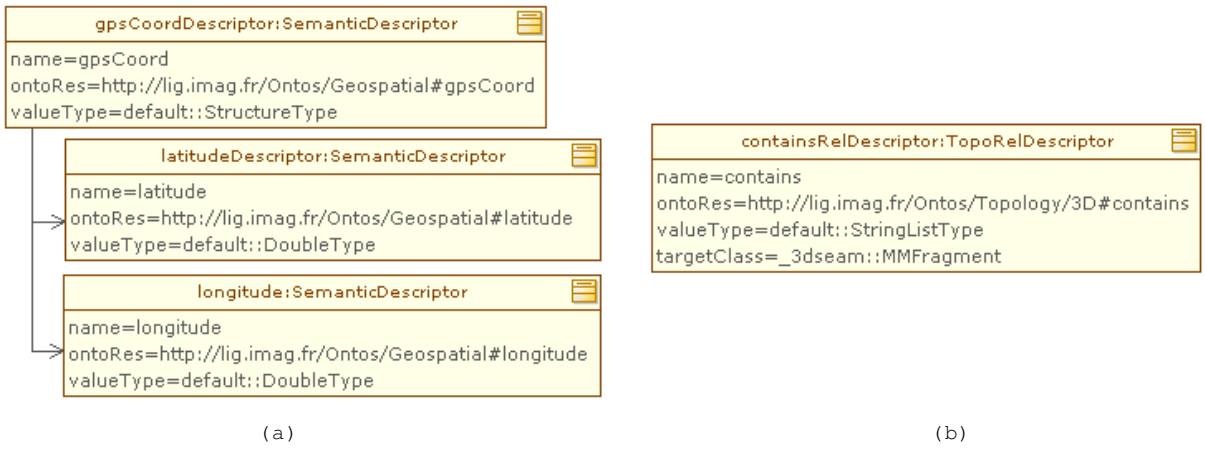


Figure 5.22 Définition d'un descripteur propriété (a) et d'un descripteur relation (b).

Dans la Figure 5.22, nous illustrons la définition d'un descripteur sémantique (`gpsCoordDescriptor`) et d'un descripteur de relation topologique (`containsRelDescriptor`). Le descripteur sémantique `gpsCoordDescriptor` correspond à une propriété nommée `gpsCoord` dont le type de valeur associé est un type structuré, car la propriété contient deux sous-propriétés (`latitude` et `longitude`) dont nous présentons également la définition. Ces propriétés sont toutes définies par rapport aux termes de l'ontologie `http://lig.imag.fr/ Ontos/Geospatial`. En ce qui concerne le descripteur topologique, en plus du nom (`contains`) de l'ontologie de référence (`http://lig.imag.fr/Ontos/Topology/ 3D#contains`), du type de valeur (une liste de chaîne de caractères retenant les identifiants des instances associées) et de sa définition, on indique aussi le type d'instances qui lui est associé (`_3dseam::MMFragment`).

Initialement, le modèle de descripteurs contient l'ensemble des descripteurs de base que nous avons proposé tout au long de ce chapitre dans les différentes sections. Cependant, **le modèle de descripteurs évolue dans le temps, de paire avec les nouvelles propriétés, descripteurs ou relations saisies lors du processus d'instanciation du modèle 3DSEAM.**

Lorsque les annotations saisies par l'utilisateur introduisent de nouvelles propriétés, non saisies auparavant dans le modèle, ou bien lorsque de nouveaux descripteurs géométriques sont renseignés pour la géométrie d'une forme, le modèle de descripteurs de 3DSEAM doit être mis à jour en concordance. **L'évolutivité du modèle de descripteurs est du ressort de l'application qui est responsable du stockage des nouvelles connaissances au sein de l'entrepôt construit au-dessus du modèle.**

Cette évolution, en parallèle des annotations et du modèle de descripteurs, **est fondamentale pour une utilisation efficace du modèle**, notamment en ce qui concerne le traitement des données (réutilisation, adaptation, recherche) au moyen de requêtes dont les critères sont construits sur les caractéristiques retenues dans les instances du modèle. Le modèle de descripteurs constitue le miroir de la structure des instances du modèle 3DSEAM. Il offre à l'utilisateur une vue générique de l'ensemble des propriétés et des relations stockées à travers l'ensemble des instances dans les diverses catégories d'information que nous avons retenues dans le modèle. Ainsi, la variabilité du modèle est contrôlée sans entraver la manière dont un utilisateur souhaite décrire telle ou telle dimension du modèle.

5.10. Synthèse

Dans ce chapitre nous avons présenté un modèle de description qui couvre l'ensemble des dimensions d'une donnée 3D. Les informations concernant la géométrie, l'apparence, la topologie, la sémantique et le profil média d'une donnée, sont matérialisées à travers les instances de classes homonymes.

Le modèle est organisé en trois parties :

- **la partie sémantique** – qui couvrent la sémantique locale et la sémantique générale,
- **la partie entité du monde** – qui fait le lien entre les entités du monde réel et leurs matérialisations dans les documents multimédia en tant que données 3D,
- **la partie fragment multimédia** – qui introduit les caractéristiques géométriques, topologiques, d'apparence et média des données 3D.

Ce découplage permet aux instances de chaque partie d'évoluer indépendamment, sans que cela affecte les autres instances du modèle. Par exemple, lorsque l'on modifie la sémantique associée à une entité, il n'est pas nécessaire d'en informer chaque fragment multimédia lui étant associé.

Par rapport aux travaux relatifs à la sémantique dans les scènes 3D que nous avons présenté dans l'état de l'art, le modèle 3DSEAM se propose comme une solution générique qui ne dépend pas d'un domaine d'application spécifique. Plus précisément, les propositions que nous avons étudiées peuvent reposer sur le modèle 3DSEAM pour la représentation de la sémantique. Les propriétés sémantiques auxquelles les différentes applications s'intéressent peuvent constituer des profils sémantiques rattachés aux entités pertinentes pour chaque application particulière.

Notre proposition, à la différence de celles que nous avons présentées dans l'état de l'art, couvre l'ensemble des caractéristiques d'une donnée 3D : sa géométrie, son apparence, sa topologie, son profil média, sa sémantique générale.

Par rapport à des solutions génériques de description telle que RDF, MPEG-7, notre solution propose une organisation des informations autour des principales catégories d'information caractérisant une donnée 3D. Cette organisation vise la réutilisation de la

sémantique en découplant le contenu et les informations sémantiques les caractérisant. RDF et/ou MPEG-7 sont considérés comme des modalités de représentation des instances de 3DSEAM.

Afin de conférer une large capacité de prise en compte des domaines d'application divers et variés et de supporter l'évolutivité de 3DSEAM, nous avons proposé un modèle de descripteurs associé au modèle 3DSEAM. Ce modèle de descripteurs contient l'ensemble des descripteurs qui sert à caractériser les instances de 3DSEAM.

Après avoir décrit le modèle 3DSEAM et les principes qui assurent son évolutivité sans entraver les principales activités autour du modèle (ajout d'information, interrogation et stockage), dans le chapitre suivant, nous présentons les principales fonctionnalités et une architecture générique pour une plate-forme qui se propose comme une solution pour la gestion du modèle 3DSEAM. Cette plate-forme gère l'évolutivité du modèle 3DSEAM en séparant les fonctionnalités de gestion des représentations spécifiques d'instances du modèle.

6. Vers une solution générique de gestion des entrepôts 3DSEAM

Dans ce deuxième chapitre de notre proposition, nous traitons le problème de la gestion homogène de l'information contenue dans les entrepôts d'annotations 3DSEAM. Le terme *entrepôt d'annotations* correspond ici à l'ensemble des instances du modèle 3DSEAM.

Nous proposons des solutions pour l'ajout et l'extraction d'informations au moyen de requêtes dont la formulation est indépendante d'un choix d'implémentation spécifique du modèle 3DSEAM. Suivant le type d'application implémentée au-dessus de l'entrepôt, il est possible qu'une représentation physique des annotations (bases de connaissances orientées objet telles qu'AROM [Page *et al.*, 2000], de type XML [Yergeau *et al.*, 2004] telle que RDF [Manola *et al.*, 2004] ou MPEG-7 [Martinez *et al.*, 2002], ou bien toute autre solution spécifique (voir Figure 6.1) soit plus adaptée qu'une autre).

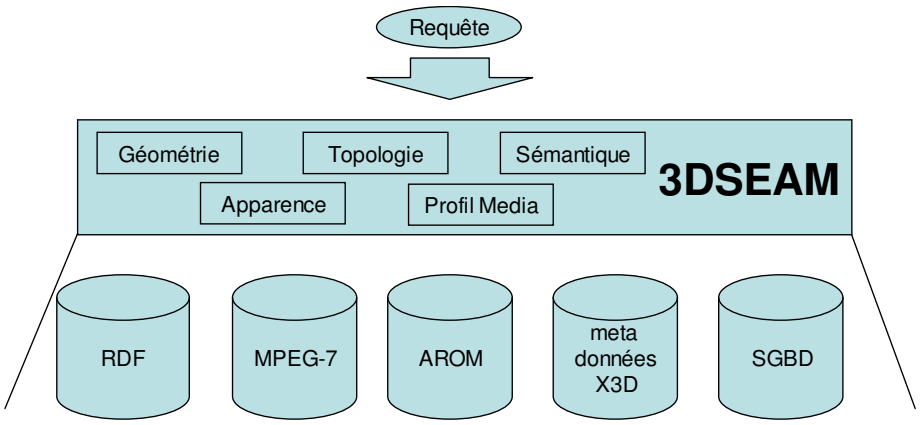


Figure 6.1 Séparation entre la représentation interne de l'entrepôt et le modèle.

Cette indépendance entre l'accès aux données et la représentation des données elles-mêmes, permet d'envisager des perspectives intéressantes de travail, telles que la construction d'un réseau d'entrepôts d'annotations dont chaque élément peut être construit, de manière indépendante et sans aucune contrainte imposée par la mise en réseau. L'indépendance est acquise en s'appuyant sur le modèle que nous avons décrit dans le chapitre précédent.

Afin d'aboutir à cette séparation entre les choix de représentation des connaissances et les méthodes de stockage et d'interrogation, nous proposons une architecture conceptuelle pour une plate-forme de gestion d'annotations dénommée 3D Annotation Framework (3DAF). Cette plate-forme, initialement présentée dans [Bilasco *et al.,* 2006b], permettra, à terme, la mise en place d'opérations élémentaires pour la gestion d'un entrepôt 3DSEAM : d'une part l'ajout, la suppression et la mise à jour des informations relatives aux données 3D et d'autre part la recherche des informations. Dans la suite de ce chapitre nous décrivons les fonctionnalités attendues de cette plate-forme.

Grâce à la représentation homogène des différents types de propriétés (géométrie, topologie, apparence, …), il est possible d'interroger chacun des aspects indépendamment, ou de constituer des critères portant sur plusieurs aspects. Nous choisissons, donc, un langage capable d'interroger l'ensemble des dimensions d'une donnée 3D (géométrie, apparence, sémantique) en s'appuyant sur l'organisation du modèle 3DSEAM. Le langage doit être flexible et extensible afin qu'il puisse être étendu pour répondre aux besoins spécifiques des systèmes relevant d'un domaine d'application précis, tel que le domaine géographique, le domaine médical [Polys, 2003][Polys *et al.*, 2004][Willis, 2007], le domaine architectural [Cruz *et al.*, 2005][Pittarello *et al.*, 2006], etc. Dans cet esprit, une extension du langage et de la plate-forme, permettant de mettre en œuvre des requêtes géospatiales au-dessus du langage d'interrogation conçu, est présentée dans le chapitre suivant.

Compte tenu du fait que nous considérons une modélisation orientée objets, nous adoptons le langage OQL [Cattel, 1994] que nous personnalisons avec des constructions et opérateurs spécifiques à la structure du modèle 3DSEAM. Le langage OQL proposé par l'ODMG[37] est fondé sur une extension de SQL supportant chemins, méthodes, héritage et collections.

Dans la suite de ce chapitre, nous passons premièrement en revue l'ensemble des fonctionnalités mises à la disposition des clients de la plate-forme qui se propose comme une solution logicielle de gestion d'annotations.

Deuxièmement, nous présentons l'organisation, dans l'entrepôt sous-jacent à la plate-forme, des instances issues de différentes classes du modèle 3DSEAM.

Troisièmement, nous détaillons le module prenant en charge l'alimentation en information de l'entrepôt.

Avant de conclure, nous insistons sur le module d'interrogation et le langage qui permet aux clients de la plate-forme de construire des requêtes couvrant l'ensemble des dimensions du modèle 3DSEAM.

6.1. Aperçu de la plate-forme 3DAF

3DAF est dédié à l'exploitation des informations sémantiques attachées aux objets 3D. Cette plate-forme facilite le processus d'annotation et gère (stockage, interrogation, mise à jour)

[37] Object Data Management Group

les annotations existantes. La plate-forme est considérée comme une couche intermédiaire, s'appuyant sur le modèle 3DSEAM, entre la formulation des besoins d'un agent (utilisateur ou application) en termes de connaissances, et la représentation réelle de la connaissance (MPEG-7, RDF, OWL, SGBD, ...).

3DAF (voir la Figure 6.2) repose sur trois composants principaux : l'*Entrepôt d'annotations 3DSEAM*, le *Gestionnaire des annotations*, et le *Gestionnaire des requêtes*.

L'entrepôt d'annotations est utilisé pour stocker les instances de 3DSEAM. Le stockage de chaque partie du modèle se fait de manière indépendante. Trois *bases de connaissances* dans l'entrepôt traitent, chacune une dimension : la sémantique, les entités et les fragments multimédia.

Des liens transversaux sont employés afin d'associer :

- les fragments à l'entité (objet réel) qu'ils représentent ;
- les entités à l'ensemble des profils sémantiques qui les décrivent ;
- les fragments à leurs propriétés sémantiques au sein de la scène d'origine.

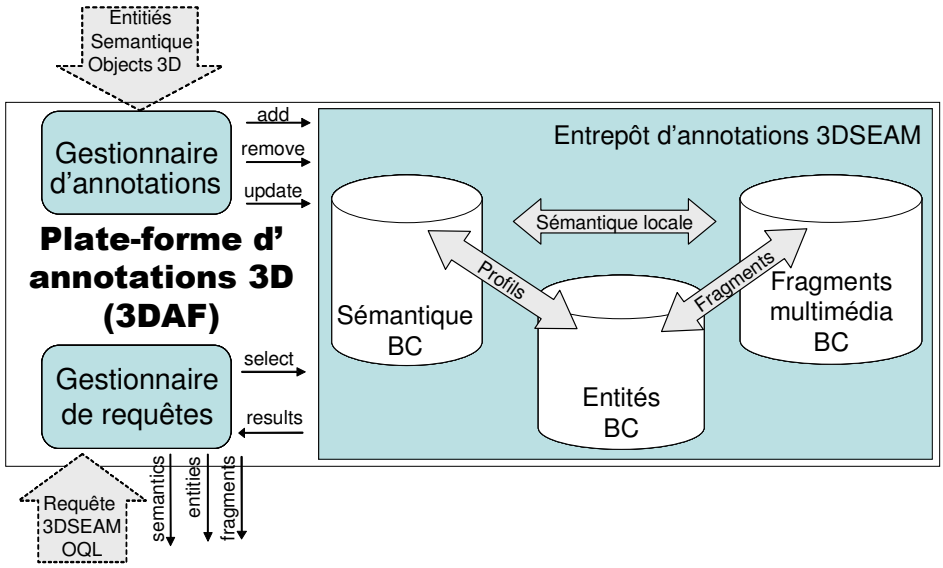


Figure 6.2 L'architecture de la plate-forme 3DAF.

Le *Gestionnaires d'annotations* et le *Gestionnaire de requêtes* représentent une couche intermédiaire entre une organisation spécifique de l'entrepôt d'annotations et une gestion externe et générique de la plate-forme. Ainsi, le monde extérieur n'est pas concerné par le choix d'une représentation interne des instances des classes du modèle 3DSEAM.

L'alimentation de l'entrepôt d'annotations est effectuée par le *Gestionnaire d'annotations*. Chaque insertion est traduite par un ensemble d'opérations qui ajoutent l'information dans le composant de stockage spécifique. Les opérations d'effacement et de mise à jour sont exécutées de manière similaire.

6.2. L'entrepôt de descriptions 3DSEAM

Dans la Figure 6.3 nous présentons une description détaillée de l'organisation de l'entrepôt 3DSEAM.

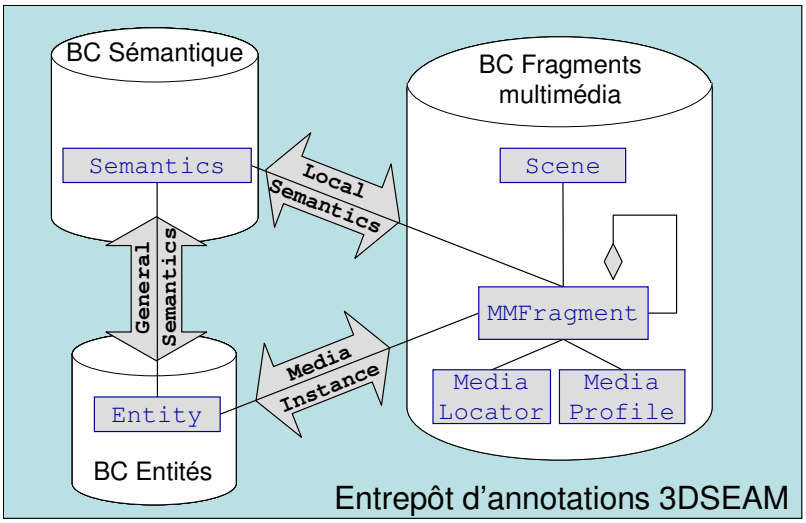


Figure 6.3 Organisation des instances 3DSEAM au sein de l'entrepôt.

Nous organisons les instances des classes du modèle 3DSEAM en trois entrepôts de connaissances, dédiés respectivement à la description de la sémantique, de l'entité et du fragment multimédia. Nous faisons ce choix afin d'isoler le plus possible les trois parties importantes du processus de caractérisation de données 3D :

- la *Base de Connaissances (BC) Sémantique*, regroupe l'ensemble des instances de profils sémantiques renseignés pour l'ensemble des entités et fragments multimédia contenus dans l'entrepôt.

  La *BC Sémantique* supporte la variété des profils sémantiques qui organisent l'ensemble des informations sémantiques associées aux entités ou aux fragments multimédia. Une autre spécificité de cette BC est de contenir des propriétés complexes permettant de mieux structurer les propriétés d'un profil. Par exemple, lorsque l'on travaille avec des profils incluant de l'information géographique il est fort possible de gérer des coordonnées modélisées comme un couple de propriétés : latitude, longitude.

- la *Base de Connaissances (BC) Entités* contient l'intégralité des entités identifiées.

  La *BC Entités* contient des instances caractérisées par leur identifiant et les éventuels liens de parenté (entités mères) avec d'autre entités.

- la *Base de Connaissances (BC) Fragments Multimédia* de l'entrepôt regroupe l'ensemble des propriétés du fragment multimédia : sa localisation, son profil média, éventuellement sa géométrie et son apparence.

  La *BC Fragments Multimédia* suppose l'existence d'un ensemble de classes prédéfinies prêtes à accueillir de l'information le plus souvent atomique en rapport avec un type spécifique de propriétés : propriétés média, propriétés d'apparence, propriétés de géométrie, etc.

La Figure 6.4 montre une vue partielle du résultat d'annotation obtenu avec 3DSEAM sur une scène 3D modélisant deux bâtiments reliés par trois passerelles. Le processus d'annotation correspond à l'instanciation de trois bases de connaissances distinctes de descriptions concernant respectivement les fragments multimédia, les entités et les profils sémantiques. Dans l'entrepôt de descriptions de fragments multimédia, un repère structurel – pointant vers l'élément racine (X3D) de la scène – est utilisé afin d'associer l'objet *ScèneRéelle* à la scène entière. L'objet

ScèneRéelle contient trois objets principaux : le *Bâtiment Gris*, le *Bâtiment Noir* et les *Passerelles*. Un troisième niveau de décomposition est considéré pour le bâtiment gris (décomposition en rez-de-chaussée, 1er, 2eme, 3eme étage gris ...) et les passerelles dont on détaille les différents niveaux (*Niveau1* à *Niveau3*). L'objet *Niveau1* est localisé en utilisant un `StructuralLocator`. Un `SpatioStructuralLocator` est utilisé pour pointer vers le rez-de-chaussée du bâtiment gris (*RdcGris*). De façon similaire, des `SpatioStructuralLocator` sont utilisés pour identifier les autres étages. Les fragments multimédia identifiés sont associés aux entités du monde réel. Par exemple, le *Bâtiment Noir* est associé à l'entité *ENSIMAG_E*. Cette entité correspond aux locaux de l'école ENSIMAG de Grenoble. Un profil sémantique *Catégorisation* est associé aux entités du monde réel. Par le biais des entités, l'information sémantique est à la fois factorisée et mise en correspondance avec la matérialisation de l'entité dans la scène. Ainsi, l'objet *Niveau1* associé à l'entité *Passerelle_1e* est caractérisé en tant que *passerelle*.

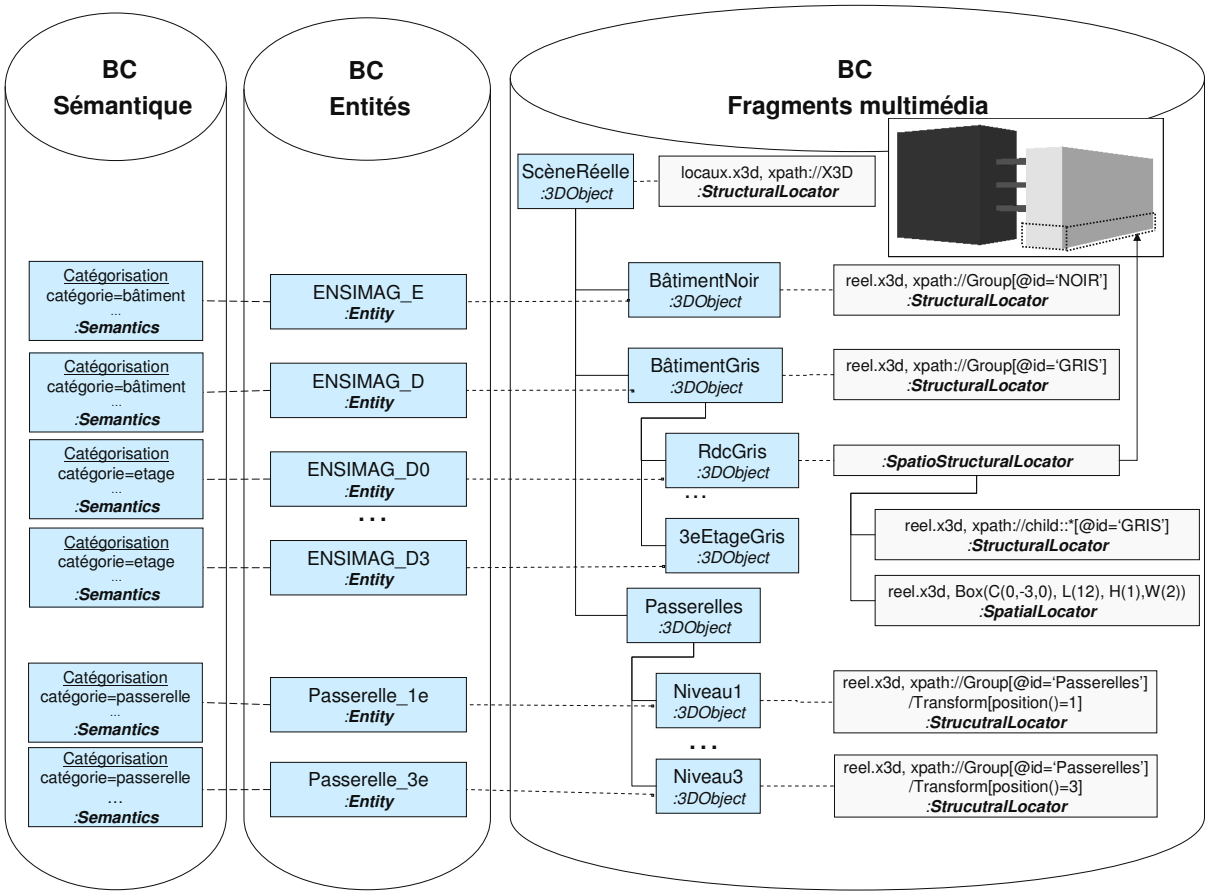


Figure 6.4 Vue partielle d'un entrepôt 3DSEAM concernant la caractérisation d'une scène.

6.3. Le module de gestion d'annotations

Comme nous n'avons pas souhaité arrêter un choix de représentation de connaissances au sein de l'entrepôt 3DSEAM, dans cette section, nous analysons uniquement les besoins et les fonctionnalités qu'un module de gestion d'annotations doit impérativement mettre en œuvre. Cette réflexion à un niveau élevé d'abstraction nous permet d'isoler les principales caractéristiques du module qui sont mises en pratique, lorsque dans un contexte applicatif spécifique, un choix de représentation est imposé.

La gestion des informations dans les bases de connaissances suppose d'avoir un moyen d'atteindre et de nommer les informations concernées. À ce niveau nous supposons qu'il n'y a pas d'autres moyens de localiser une information à part à l'aide de son identifiant. Des moyens pour identifier une information, autres que par son identifiant, sont présentés dans la section suivante dans laquelle nous étudions le module d'interrogation.

Afin d'identifier les fonctionnalités requises nous considérons le cas d'utilisation le plus générique : l'introduction de l'ensemble des connaissances relatives à un fragment 3D. Ceci correspond au processus d'instanciation des diverses classes du modèle 3DSEAM à partir d'un ensemble d'informations fourni par le client (utilisateur ou application).

Nous avons identifié trois groupes d'opérations atomiques. Chacun correspond à une étape spécifique du cycle de vie de l'entrepôt : la création des instances des classes principales du système (`MMFragment`, `Entity`, `Semantics`), la saisie des propriétés et des relations spécifiques à chaque dimension (par exemple, la couleur dominante de la texture d'un objet 3D), la mise en relation des instances de 3DSEAM situées dans différents entrepôts (par exemple, l'association d'une entité avec un fragment multimédia).

a) L'instanciation de nouveaux éléments se traduit par :

   i) L'introduction d'un nouveau fragment multimédia (instanciation de la classe `Object3D` ou `MMFragment`) – lors de la création d'un nouveau fragment multimédia, toutes les propriétés (localisation, profil média et pour les objets 3D la description de la géométrie, de l'apparence et de la topologie) caractérisant le fragment lui-même, sont instanciées et prêtes à accueillir les informations relatives au fragment multimédia ;

   ii) L'introduction d'un nouvel objet recueillant la sémantique d'un fragment ou d'une entité (instanciation de la classe `Semantics`) ;

   iii) L'introduction d'un nouveau profil sémantique (instanciation de la classe `SemanticProfile`) – lors de l'introduction d'un nouveau profil sémantique on doit être en mesure d'indiquer le nom du profil et la description de chaque propriété ou relation sémantique recueillie par le profil ;

   iv) L'introduction d'une nouvelle entité (instanciation de la classe `Entity`) – lors de l'introduction d'une nouvelle entité il suffit d'indiquer le nom de l'entité.

b) L'ajout de nouvelles informations aux éléments présents se traduit par :

   i) L'ajout d'information spécifique à la caractérisation du fragment multimédia (sa localisation, son profil média et selon les cas son apparence, sa géométrie, sa topologie et sa sémantique locale) ;

   ii) L'ajout d'information dans une instance de la classe `Semantics` (associée à un fragment ou à une entité) directement ou à travers un profil sémantique.

c) La mise en relation des éléments (création de tuples) se traduit par :

   i) La mise en relation entre un objet sémantique et un fragment multimédia dans le cadre d'une caractérisation locale du fragment (la relation `Local Semantics`) ;

   ii) La mise en relation entre un objet sémantique et une entité (la relation `Global Semantics`) ;

   iii) La mise en relation entre une entité et un fragment multimédia ;

iv) L'association d'un profil sémantique à la sémantique d'un fragment ou d'une entité.

Les points a) et b) correspondent à l'instanciation de la partie fixe du modèle, celle qui précise l'organisation et les principes de caractérisation de l'information 3D. Le point c) correspond à la gestion de la partie extensible du modèle, c'est-à-dire les propriétés de chaque classe de description (géométrique, d'apparence, etc.).

6.3.1. Création de la structure fixe du modèle

Le module d'annotation permet uniquement d'instancier les structures du modèle 3DSEAM faisant partie du squelette fixe du modèle : la classe `Semantics`, la classe `SemanticProfile`, la classe `Entity`, la classe `MMFragment`, les associations `Global Semantics`, `Local Semantics` et `Media Instance`. Nous proposons un ensemble de méthodes pour la création, respectivement la destruction, des instances et des tuples :

- *CreateEntity(id_entity[, id_parent])* – création d'une entité (objet réel). Le paramètre *id_entity* représente l'identifiant de l'entité. Le paramètre optionnel *id_parent* permet d'indiquer l'entité parente de l'entité courante par rapport à l' organisation hiérarchique des entités dans le monde réel (par exemple, une entité *bâtiment* est composée de plusieurs entités *étages*) ;
- *CreateSemanticProfile(id_profile, property_list, relation_list)* – création d'un profil sémantique. Le paramètre *id_profile* représente l'identifiant du profil. Le paramètre *property_list* contient une liste de couples précisant les noms des propriétés et les alias des propriétés dans le cadre du profil. Le paramètre *relation_list* précise au moyen de couples (nom de relation, alias dans le cadre du profil) la liste de relations sémantiques du profil sémantique. Si le nom d'une propriété se révèle complexe (utilisation d'espace de noms, etc.), afin d'en simplifier l'adressage au sein d'un profil, on peut lui associer un alias ou un nom court. La même approche est également applicable à la construction de la liste *relation_list ;*
- *CreateSemantics (id_semantics, profile_id_list)* – création d'un objet contenant la sémantique. Le paramètre *id_semantics* représente l'identifiant de l'objet. Le paramètre *profile_id_list* contient la liste des profils identifiés par leur nom qui sont associés à l'objet sémantique que l'on construit. Cette opération peut également avoir un impact sur le modèle de descripteurs car de nouvelles propriétés sont incluses comme moyen de caractérisation ;
- *CreateMMFragment(id_frag, type_frag, localisation [, id_parent])* – création d'un fragment multimédia. Le paramètre *id_frag* représente l'identifiant du fragment. Le paramètre *type_frag* indique le type de fragment (`Object2D`, `Object3D`, `MMFragment`, `Content3D`, `Content2D`, etc.). Le paramètre *localisation* contient une chaîne de caractères correspondant au repère de localisation du fragment dans le document le contenant. Cette chaîne est construite suivant la grammaire présentée dans la Figure 6.5. La grammaire reprend la structure logique du modèle de localisation spécifique à 3DSEAM. Le paramètre optionnel indique le parent du fragment courant et permet de définir la structure logique de la scène ;
- *AssociateSemanticProfile(id_sem, id_profile)* – associe un profil sémantique à un objet contenant la sémantique d'un fragment ou d'une entité. Le paramètre *id_sem* indique l'objet cible. Le paramètre *id_profile* indique le nom du profil ;

- *AssociateSemantics(id_entity, id_sem)* – associe un objet sémantique à une entité. Le paramètre *id_entity* indique l'entité. Le paramètre *id_sem* définit l'objet sémantique ;
- *AssociateEntity(id_frag, id_entity)* – associe un fragment multimédia à une entité. Le paramètre *id_frag* correspond à l'identifiant du fragment. Le paramètre *id_entity* correspond à l'identifiant de l'entité concernée ;
- *AssociateLocalSemantics(id_frag, id_sem)* – associe un fragment multimédia à un objet sémantique dans le cadre de la relation `LocalSemantics`. Le paramètre *id_frag* indique l'identifiant du fragment multimédia. Le paramètre *id_sem* correspond à l'identifiant de l'objet sémantique.

```
<Localisation>      ::= Mixed "(" <SpStructExpr> ")" | SpatialRef "(" <SpatialExpr> ")" | StructRef "(" <StructExpr> ")"

<SpStructExpr>      ::= "[" <URL> "," <StructLocator> "," <SpatialLocator> "]" |
                        "[" <URL> "," <StructLocator> "," <SpatialLocator> "]" "+" <SpStructExpr>

<StructExpr>        ::= "[" <URL> "," <StructLocator> "]" | "[" <URL> "," <StructLocator> "]" "+" <StructExpr>
<StructLocator>     ::= "{" <UniStructLocator> "}" | "{" <UniStructLocator> "}" "+" <StructLocator>
<UniStructLocator>  ::= <XPathExpr>

<SpatialExpr>       ::= "[" <URL> "," <SpatialLocator> "]" | "[" <URL> "," <SpatialLocator> "]" "+" <SpatialExpr>
<SpatialLocator>    ::= "{" <UniSpatialLocator> "}" | "{" <UniSpatialLocator> "}" "+" <SpatialLocator>
<UniSpatialLocator> ::= <Locator2D> | <Locator3D>

<Locator2D>         ::= Rect "(" C "(" <N> "," <N> ")" "," L "(" <N> ")" "," W "(" <N> ")" ")" |
                        Ellipse "(" C "(" <N> "," <N> ")" "," RX "(" <N> ")" "," RY "(" <N> ")" ")"
<Locator3D>         ::= Box "(" C "("<N> "," <N> "," <N> ")", "," W "(" <N> ")" "," H "(" <N> ")", "," D "(" <N> ")" ")" |
                        Sphere "(" C "("<N> "," <N> "," <N> ")", "," R"(" <N> ")" ")" |
                        Cylinder "(" C "("<N> "," <N> "," <N> ")", "," R "(" <N> ")" "," H "(" <N> ")" ")"

<N>                 ::= … a real value …
<XPathExpr>         ::= … an XPath expression cf. to [Clark et al., 1999]
```

Figure 6.5 Grammaire BNF [Backus, 1959] pour les repères de localisation en 3DSEAM.

Pour la suppression des descriptions, des méthodes similaires à la création sont fournies (*RemoveEntity*, *RemoveMMFragment*, *RemoveSemantics*, *RemoveSemanticProfile*, *DissociateSemanticProfile*, *DissociateEntity*, *DissociateSemantics*, *DissociateLocal Semantics*). Cependant, lors de la suppression il est nécessaire de considérer l'ensemble des instances mises en relation avec l'entité ciblée. Par exemple, lors de la suppression d'une entité il faut détruire également les associations dont elle fait partie (vis-à-vis des fragments média et de la sémantique).

Afin d'illustrer l'utilisation de ces outils, nous présentons dans la Figure 6.6 la partie du scénario d'instanciation concernant les fragments multimédia, les entités, la sémantique et les profils sémantiques. Le scénario est découpé en trois parties :

- création des fragments multimédia (lignes 1-3 de la Figure 6.6),
- création de entités et mise en relation avec les fragments multimédia existants (lignes 4 et 5 de la Figure 6.6) et
- création de la dimension sémantique et de l'association des informations sémantiques à une entité (lignes 6 - 8 de la Figure 6.6).

Initialement, on crée le fragment correspondant à la scène entière. Un repère structurel (*StructRef*) est introduit (ligne 1 de la Figure 6.6) pour pointer vers la racine du fichier X3D (le nœud `X3D`).

```
1: CreationMMFrag("ScèneRéelle", "Scene", "StructRef([http://.../scene.x3d,{xpath://X3D}])");
2: CreationMMFrag("BâtimentGris", "Object3D", "StructRef([http://.../scene.x3d,
        {xpath://Group[@DEF='Gris']}]), "SceneReelle");
3: CreationMMFrag("rdcGris","Object3D", "Mixed([http://.../scene.x3d,
        {xpath://Group[@DEF='Gris']},{Box(0,-3,0,12,1,2)}])","BatimentGris");

4: CreateEntity("ENSIMAG_D");
5: AssociateEntity("BâtimentGris", "ENSIMAG_D");

6: CreateSemanticProfile("Catégorisation", "['catégorie','en::category'],['nom','en::realName']");
7: CreateSemantics("ENSIMAG_D_SEM","{Catégorisation}");
8: AssociateSemantics("ENSIMAG_D","ENSIMAG_D_Sem");
```

Figure 6.6 Extrait du scénario de caractérisation d'un fragment de la scène de la Figure 6.4.

Ensuite, on définit (ligne 2 de la Figure 6.6) le fragment *BâtimentGris* à l'aide d'un repère structurel qui référencie le nœud `Group` dont l'attribut `DEF` a la valeur *Gris*. Ce fragment est instancié comme faisant partie de la *ScèneRéelle*.

En troisième lieu, on instancie (ligne 3 de la Figure 6.6) l'objet 3D *RdcGris* correspondant au rez-de-chaussée du bâtiment gris. Cet objet est localisé par un repère mixte combinant le repère structurel identifiant l'élément *BâtimentGris* et un repère spatial définissant la boîte englobant le rez-de-chaussée du bâtiment gris. Ce nouvel objet 3D est inscrit parmi les fils de l'objet *BâtimentGris*.

La deuxième partie du scénario (lignes 4-6 de la Figure 6.6) est consacrée à l'association du fragment *BâtimentGris* avec l'entité qu'il représente dans la scène. L'entité en question est dénommée *ENSIMAG_D* et elle est mise en relation, grâce à l'opérateur *AssociateEntity,* avec le fragment *BâtimentGris*. La création de l'entité est nécessaire car nous considérons la situation où le système ne contient pas l'entité représentée par le fragment.

Finalement, la création de la dimension sémantique se réalise en trois temps :

- en premier lieu, on doit créer ou récupérer les noms des profils sémantiques que l'on souhaite associer à la sémantique de l'objet caractérisé, qu'il soit entité ou fragment multimédia. Dans l'exemple, nous créons (ligne 7 de la Figure 6.6) un nouveau profil sémantique *Catégorisation* qui regroupe deux propriétés : la *en::category* et la *en::realName* de l'objet. À chacune des propriétés, on associe un alias au sein du profil (*catégorie*, respectivement *nom*) afin d'en simplifier l'écriture.
- deuxièmement, on crée une instance de la classe `Semantics` regroupant l'ensemble de propriétés sémantiques (*ENSIMAG_D_SEM*) en précisant la liste des profils sémantiques qui est couverte par celui-ci (*Catégorisation*).
- en dernier, on associe cette instance *ENSIMAG_D_SEM* à l'entité correspondante (*ENSIMAG_D*) qui devient ainsi associée à l'ensemble des propriétés sémantiques contenus par l'instance.

6.3.2. Gestion de la partie extensible du modèle

Les structures accueillant les parties variables du modèle (`Semantics`, `Geometry3D`, `Apparence`, `Topology`, `MediaProfile`) sont créées automatiquement lors de l'instanciation des éléments. Le choix d'implémentation de ces éléments doit être sensible à l'extensibilité de l'ensemble des propriétés qui les composent.

```
<PropLongName>      ::= <DimensionName> "." <PropertyConstruct> |
                        Semantics "!" <ProfileName> "." <PropertyConstruct>
<DimensionName>     ::= Geometry | Appearance | MediaProfile | Semantics
<PropertyConstruct> ::= <PropertyName> |
                        <PropertyName> "." <PropertyConstruct>
<PropertyName>      ::= <Name> | <Namespace> "::" <Name>
<ProfileName>       ::= <Name>
<Name>              ::= regexp([a-zA-Z_]([a-zA-Z0-9_])*)
<Namespace>         ::= regexp([a-z]+)
```

Figure 6.7 Grammaire BNF [Backus, 1959] pour la construction des noms longs de propriétés.

En ce qui concerne la mise à jour des informations se trouvant dans la partie flexible du modèle, nous proposons une convention de noms capable de gérer la variabilité des structures et de garantir à chaque propriété un nom unique au sein du modèle. Nous introduisons la notion du *nom long* comme moyen d'identification non ambiguë d'une propriété au sein du modèle. Dans la Figure 6.7 nous présentons la grammaire régissant la construction de *noms longs* pour les propriétés (*PropLongName*). Nous retrouvons la notation portant uniquement sur les noms de propriétés, ainsi que celle qui prend en compte les noms de profils pour la dimension sémantique. Le *nom long* inclut le nom de la classe (*DimensionName*), dans laquelle la propriété est localisée, suivi du nom de la propriété (*PropertyConstruct*). Lorsque la propriété fait partie d'une propriété composite, dans la construction du nom étendu nous incluons également la position de la sous-propriété au sein de la propriété principale. Dans le cas de propriétés associées aux profils sémantiques, il est possible d'utiliser le nom du profil et l'alias de la propriété au sein du profil afin d'y accéder.

Les dimensions de caractérisation sont soit associées à un fragment multimédia, soit à une entité, soit aux deux dans le cas de la sémantique. Ainsi, nous proposons deux types d'opérations visant respectivement la gestion des propriétés relatives aux fragments multimédia et la gestion des propriétés sémantiques relatives aux entités. Ce choix permet de répondre à la situation où un fragment multimédia est relié à un objet sémantique soit directement, soit à travers l'entité qui lui est associée. Dans le premier cas, la relation `LocalSemantics` sert à définir la sémantique locale à la scène de l'objet, tandis que dans le deuxième, il est nécessaire de retrouver l'entité associée au fragment avant de traverser la relation `GeneralSemantics`. Afin d'éviter toute ambiguïté lors des opérations de gestion nous proposons les opérations suivantes :

- *AddFragmentRelatedProperty(id_frag, long_name, value)* – ajoute dans les instances associées au fragment multimédia *id_frag*, la propriété dénotée par le *long_name*, ainsi que sa valeur. Dans le cas où le *long_name* référencie la dimension sémantique, la propriété est associée à la sémantique locale du fragment. Pour saisir des informations sémantiques d'ordre général, il est nécessaire de s'appuyer sur l'opération présentée ci-dessous en indiquant le nom de l'entité associée.
- *AddEntityRelatedProperty(id_entity, long_name, value)* – ajoute dans l'objet sémantique associée à l'entité *id_entity* la propriété définie par le *long_name* et sa valeur *value*.

Des opérations permettant la suppression des valeurs d'une propriété sont également considérées : *RemoveFragmentRelatedProperty*, *RemoveEntityRelatedProperty*. Chacune de ces deux méthodes suppose la saisie d'un identifiant de fragment, respectivement entité, ainsi qu'un nom étendu indiquant la propriété concernée.

Dans la Figure 6.8, nous illustrons l'utilisation de ces opérations afin d'ajouter de l'information sémantique à l'entité *ENSIMAG_D* et d'indiquer l'auteur du fragment appelé

BâtimentGris que nous avons déjà introduit dans la Figure 6.6. Deux types de notations sont employés pour ajouter la sémantique. Pour la première ligne, nous utilisons la notation à base de profil et d'alias (*Semantics!Catégorisation.nom*), tandis que dans la deuxième ligne nous utilisons la notation à base de nom complet de propriété (*Semantics.en::category*). Le troisième exemple présente le cas d'une propriété composée (*gpsCoord* – une coordonnée GPS est caractérisée par la latitude et la longitude). Ici, on s'intéresse à l'attribut *latitude* de la propriété *gpsCoord* du profil sémantique *Geospatial* associé à la sémantique de l'entité *ENSIMAG_D*. Finalement, nous présentons un deuxième exemple concernant l'ajout d'une propriété relative au fragment *BâtimentGris*. Cette propriété *author*, qui désigne la personne qui a construit le fragment multimédia, est stockée dans le profil média du fragment (*MediaProfile*).

```
1: AddEntityRelatedProperty
        ("ENSIMAG_D", "Semantics!Categorisation.nom","Bâtiment de l'ENSIMAG");

2: AddEntityRelatedProperty
        ("ENSIMAG_D", "Semantics.en::category","bâtiment à étages");

 3: AddEntityRelatedProperty
        ("ENSIMAG_D", "Semantics!Geospatial.gpsCoord.latitude","3°51");

4: AddFragmentRelatedProperty
        ("BâtimentGris", "MediaProfile.author","MB");
```

Figure 6.8 Convention de noms pour les propriétés complexes.

Les opérateurs proposés et les conventions de noms que nous avons introduits dans cette section couvrent l'ensemble des démarches nécessaires pour l'alimentation de l'entrepôt de description en données sémantiques. Les conventions de noms proposées permettent notamment de faire face à la variabilité en termes de propriétés associées aux différentes dimensions de caractérisation.

Dans la section suivante nous nous intéressons à la conception du module d'interrogation.

6.4. Le module d'interrogation

Le module d'interrogation vient compléter les fonctionnalités mises à disposition par le module d'annotation afin d'aboutir à une complète indépendance entre la représentation des données contenues dans le modèle 3D et l'usage de ces informations.

Nous n'abordons pas cette partie de la plate-forme en proposant un ensemble prédéfini d'opérations permettant l'accès aux informations de manière directe, à l'image des méthodes permettant l'ajout d'information présentée dans la section précédente (*CreateSemantics*, *CreateEntity*, *AddEntityRelatedProperty*, …). Nous justifions la non adoption de cette approche à base de méthodes d'accès prédéfinies, par le fait que lors de l'interrogation, les utilisateurs ne connaissent pas les identifiants des objets qu'ils cherchent, puisqu'ils disposent uniquement de l'information définissant les critères de recherche. Par exemple, ils savent qu'ils cherchent un bâtiment gris mais il ne connaissent ni son identifiant, ni la scène qui le contient. Dans le cas de l'alimentation de l'entrepôt cette approche est suffisante, car lorsque les utilisateurs saisissent l'information dans le système, ils savent par rapport à quelle entité ou fragment multimédia cela ce fait. De plus, il est irréaliste d'essayer de prévoir à l'avance par un ensemble d'opérations d'accès prédéfinies, la totalité des critères et l'usage que les utilisateurs en font afin d'indiquer l'information recherchée.

Nous considérons plus adaptée une approche visant la définition d'un langage d'interrogation qui donne la liberté aux utilisateurs experts de formuler les critères de recherche en fonction de leurs besoins. Toutefois, nous ne négligeons pas que, dans certains cas spécifiques, la proposition d'un nombre limité d'opérations d'accès à l'information peut se révéler assez utile. Nous considérons que ces opérations doivent s'inscrire dans la démarche de séparation entre l'usage et la représentation. À ce sujet, nous pensons qu'il est raisonnable que ces opérations d'accès à l'information soient construites au-dessus du langage de requêtes que nous développons dans la suite de cette section.

Une extension du langage OQL [Cattel, 1994] utilisant les concepts et l'organisation de 3DSEAM (fragment, entité, sémantique, profil de média représenté comme une liste de propriétés, etc.) **est proposée pour faciliter l'interrogation des instances stockées au sein de la plate-forme 3DAF**. Le rôle principal du module d'interrogation est de traduire ces requêtes à base de concepts 3DSEAM en requêtes qui dépendent des conventions de stockage propres à chaque type d'entrepôt.

Avant de présenter l'extension du langage, nous nous penchons sur une analyse des besoins en termes de requêtes, dans le but de munir le langage d'interrogation de constructions et d'opérateurs capables d'exploiter pleinement l'ensemble des connaissances stockées dans l'entrepôt, et subvenir aux besoins en termes d'information des utilisateurs de la plate-forme.

6.4.1. Une typologie de requêtes sémantiques sur les scènes 3D

Dans cette section nous proposons une analyse des requêtes types qu'une application, ou des utilisateurs, pourrait formuler afin d'accéder aux différents types d'information de l'entrepôt.

Comme illustré dans le chapitre précédent, un objet 3D est caractérisé par sa géométrie, son apparence, les relations topologiques entretenues avec ses voisins, et sa sémantique. **Puisque l'entité et le fragment multimédia sont les parties centrales du modèle, ils sont employés pour accéder directement, et sans équivoque, à toutes les informations qui les caractérisent (par exemple, récupérer la couleur/la géométrie/la topologie/la sémantique de l'objet). Nous appelons ce type de requête une *requête directe*.** Une *requête directe* ne présente pas un niveau important de variabilité, car les propriétés et les relations considérées sont définies sans équivoque par rapport à une entité ou fragment multimédia identifiés par leur identifiant.

Réciproquement, chaque type d'information stocké dans 3DSEAM peut être employé comme critère de recherche afin de retrouver d'autres informations sur les scènes indexées. Par exemple, on peut demander de montrer tous les objets *rouges*, tous les objets ayant la forme d'une *sphère* ou seulement les bâtiments à trois étages. Nous appelons ce type de requête une *requête inverse*. **Une *requête inverse* suppose la formulation d'une requête voulant accéder, soit à un fragment, soit à une entité spécifique, à partir de leurs propriétés respectives.** Différents types de critères peuvent être combinés afin de formuler des requêtes plus abouties et par conséquent, obtenir des résultats plus précis. Par exemple, la requête « *montrer seulement les bâtiments rouges à quatre étages situés dans le campus universitaire* », combine respectivement la sémantique, l'apparence et la topologie. Dans le cas d'une requête inverse, il est fort probable que le résultat soit composé de plusieurs objets (fragments ou entités) même si plusieurs critères sont utilisés, car ils peuvent partager les mêmes propriétés. Ainsi, lorsque les critères ne sont pas assez puissants pour isoler l'instance particulière recherchée par l'utilisateur, il est souhaitable de pouvoir assurer des tris et regroupements des éléments afin de faciliter la recherche (visuelle) d'une instance particulière. Cependant, dans ce travail nous n'abordons pas ce point.

Un troisième type de requête, fondamental pour notre modèle, sont les *requêtes transversales*. **Une *requête transversale* suppose des opérations de mise en relation des**

éléments se trouvant dans les différentes bases de connaissances d'un entrepôt 3DSEAM. Elle suppose également des informations sur l'identifiant de l'objet (entité ou fragment) constituant le point de départ de la requête. Ainsi, lorsque l'on veut accéder aux propriétés sémantiques d'une entité, en partant de l'identifiant d'un fragment, on formule une requête transversale. Réciproquement, si, en partant d'une entité, l'on veut accéder aux propriétés d'une de ses représentations média (fragments), on doit transiter par le lien entre les deux bases de connaissances : celui des entités et celui des fragments.

D'autres types de requête peuvent être dérivés des précédents. Il est possible de **retrouver des objets, ou des propriétés d'objets en relation avec d'autres objets et/ou propriétés à l'aide des *requêtes indirectes***. À partir de ces types de base, nous pouvons envisager de construire des requêtes qui utilisent plusieurs niveaux d'indirection.

Une *requête indirecte* s'exécute en deux étapes :

- le passage de la valeur d'une propriété à un ensemble d'objets (entités ou fragments) dont la valeur de la propriété concernée est égale à celle indiquée, et
- le passage de l'objet résultat à une autre propriété de ce dernier. Généralement, une *requête indirecte* suppose l'exécution d'une *requête inverse* et d'une *requête directe*.

Cependant, des requêtes transversales peuvent également faire partie du programme d'exécution. Ce type de requête peut être utilisé pour trouver une propriété en fonction d'une autre propriété d'un objet quelconque. Nous considérons une requête qui consiste à trouver la hauteur moyenne des bâtiments situés à l'intérieur du campus universitaire. Une *requête inverse* portant sur un critère topologique (*contains*) permet d'extraire l'ensemble des fragments représentant des objets sur le campus universitaire. Ensuite, une *requête transversale* qui filtre l'ensemble des fragments est appliquée afin d'en extraire les bâtiments (*critère sémantique*). Finalement, par le biais d'une *requête directe* on peut obtenir les hauteurs associées à chaque bâtiment. Ce deuxième cas de figure peut être illustré par la requête suivante « *trouver deux bâtiments ayant la même couleur dominante* ».

Lorsque les requêtes concernent plusieurs objets, dont l'évaluation des propriétés doit se faire simultanément, il est nécessaire de prévoir des mécanismes de jointures efficaces. Du fait que nous focalisons notre effort uniquement sur le modèle 3DSEAM, il est envisageable d'imposer au niveau du langage des constructions spécifiques facilitant l'écriture de requêtes et potentiellement leur exécution efficace.

Un autre point que nous souhaitons aborder ici est l'extensibilité du langage. Le modèle 3DSEAM présente une grande variabilité au niveau des propriétés et de leurs significations. Dans ce contexte, il est difficile de proposer un ensemble convenable d'opérateurs prédéfinis qui subviennent aux besoins des domaines d'application spécifiques. Les types de requête qui ont été présentés dans les paragraphes précédents concernent exclusivement les concepts représentés dans le modèle 3DSEAM. Cependant, des requêtes combinant des propriétés, des relations ou des opérateurs externes avec les propriétés et les relations internes de 3DSEAM, offrent des possibilités plus intéressantes encore. Ce genre de requête doit être traité au niveau de l'application où la sémantique des propriétés, relations et opérateurs externes est connue, et donc où ces entités peuvent être exploitées. Ainsi, nous souhaitons faciliter l'introduction de nouveaux opérateurs spécifiques aux diverses implémentations de la plate-forme. Pour atteindre ce niveau de transparence, nous proposons un répertoire qui renseigne sur le type d'opérateurs supportés au sein de la plate-forme et les modules externes capables de les interpréter à l'image du travail proposé dans [Bilasco *et al.*, 2006c] et [Bilasco *et al.*, 2007a].

Dans la suite, nous présentons l'extension d'OQL que nous proposons afin de combler les besoins identifiés dans cette section.

6.4.2. Une extension OQL pour 3DSEAM

Nous souhaitons proposer un outil d'interrogation spécifique au modèle 3DSEAM afin de faciliter la tâche d'extraction de l'information sur les données 3D référencées dans le modèle. À l'image de la proposition de [Fatemi *et al.*, 2003], l'extension OQL que nous proposons masque aux utilisateurs la complexité du modèle, de sa structure et de sa représentation.

En fonction des choix spécifiques de représentation des bases de connaissances, la plate-forme 3DAF implémentée doit s'appuyer sur le langage d'interrogation sous-jacent à la représentation, afin d'exécuter correctement les requêtes génériques OQL.

6.4.2.1. Structure générale d'une requête 3DSEAM

Sur le modèle d'une requête classique OQL, une requête 3DSEAM OQL est composée de trois parties distinctes : la clause SELECT, la clause FROM et la clause WHERE. La clause SELECT associe des noms de variable aux informations que l'on souhaite extraire en exécutant la requête. La clause FROM relie les variables utilisées dans la clause SELECT aux classes du modèle. Les critères effectifs de recherche sont indiqués dans la clause WHERE. Cette clause contient l'ensemble des contraintes qui s'appliquent aux variables de la clause SELECT afin d'en garder uniquement les valeurs qui sont en accord avec les critères de la requête.

Dans la Figure 6.9, nous illustrons la structure générale d'une requête (`<Query>`).

```
<Query>    ::= SELECT <ResultExprList>
               FROM <DeclList>
               [WHERE <Criteria>]
```

Figure 6.9 Règle BNF [Backus, 1959] de formation d'une requête 3DSEAM OQL.

Avant de détailler les éléments de chaque clause de la requête nous rappelons notre objectif concernant le langage de requêtes. Nous voulons mettre au cœur de l'extension proposée des constructions facilitant la formulation des requêtes en masquant l'organisation de l'entrepôt à l'utilisateur et lui permettant également d'accéder de manière homogène à la partie extensible du modèle.

6.4.2.2. Désignation de résultats de la requête

La clause SELECT contient la liste des résultats que l'utilisateur veut extraire à partir de l'entrepôt de données. Chaque élément de cette liste est mis en relation avec la *structure* du modèle à laquelle il correspond. Par rapport aux différents composants de modèle 3DSEAM, nous avons identifié quatre types de résultats qui désignent respectivement une instance, une propriété d'une instance, une collection de propriétés d'une instance ou un profil sémantique. Les deux derniers types présentent certaines similitudes car ils correspondent à l'extraction de plusieurs propriétés attachées à une même instance. En effet, l'extraction d'un profil sémantique est équivalente à l'extraction des valeurs de l'ensemble des propriétés regroupées au sein du profil sémantique en question.

Tous ces types de résultats ont en commun le fait qu'ils sont évalués par rapport à une instance. Nous utilisons la notion de *variable* pour désigner une instance du modèle dans une requête de type SELECT. Ainsi, la spécification de tous les résultats suit le modèle suivant

nomVariable{partieSpécifiqueAChaqueType}. Par exemple *appearance{dominantColor}* dénote la propriété dont le nom correspond à la *dominantColor* en rapport avec l'instance représentée par la variable *appearance*.

Afin de maîtriser l'hétérogénéité qui peut être observée au niveau des propriétés attachées aux classes correspondant aux dimensions descriptives du modèle (*Geometry*, *Appearence*, *Topology*, *Semantics, MediaProfile*), nous proposons une solution similaire à celle proposée pour l'ajout d'information dans le système. Ainsi, pour indiquer la liste des variables résultats de la clause SELECT, la désignation d'une propriété reprend partiellement les règles de la grammaire BNF [Backus, 1959] pour la construction d'un nom long de propriété présentée dans la Figure 6.7. Nous modifions la désignation d'une propriété par rapport au nom long de la façon suivante : là où le premier préfixe de la propriété correspond à la dimension descriptive à laquelle la propriété appartient (`Semantics!Geospatial.gpsCoord.latitude`), nous mentionnons le nom de la variable (`s!Geospatial.gpsCoord.latitute`).

Une autre différence par rapport à la notation précédente est que lorsque l'on interroge l'entrepôt il se peut que l'on ait besoin de plusieurs propriétés de la même instance. Ainsi nous devons faire évoluer la notation pour qu'elle supporte l'association d'une liste de propriétés avec l'instance. Nous rappelons qu'au niveau de la requête l'instance est désignée par la variable que nous lui avons associée (*s* dans le cas présent). Chaque propriété de la liste est introduite en suivant les règles énoncées dans le paragraphe précédent, à l'exception des propriétés auxquelles on accède à travers un profil sémantique. Dans ce cas, la construction définissant la propriété débute par le caractère '!', suivie du nom du profil et du nom local de la propriété. Afin d'illustrer cette notation nous proposons une construction qui demande le regroupement des propriétés *latitude* et *longitude* du profil sémantique `Geospatial` associé à une variable de type `Semantics`. La construction est la suivante `s{!Geospatial.gpsCoord.latitude, geo::gpsCoord.longitude}`. Dans cet exemple, deux modalités d'accès sont illustrées. Le premier élément correspond à l'accès à une propriété à travers le profil sémantique, le deuxième accède à la propriété à l'aide du nom complet de la propriété définie dans le cas présent dans l'espace de noms `geo::`.

```
1a: <ResultExprList>          ::= <ResultExpr> | <ResultExpr> "," <ResultExprList>
2a: <ResultExpr>              ::= <SimpleResultExpr> | <CompoundResultExpr>

3a: <SimpleResultExpr>        ::= <VarName> | <VarName> "!" <ProfileName> |
3b:                               <VarName> "." <PropertyConstruct> |
3c:                               <VarName> "!" <ProfileName>.<PropertyConstruct>

4a: <CompoundResultExpr>      ::= <VarName> "{" <PropertyConstructList> "}"

5a: <PropertyConstructList>   ::= <PropertyConstruct> | "!" <ProfileName> "." <PropertyConstruct>
5b:                               <PropertyConstruct> "," <PropertyConstructList> |
5c:                               "!" <ProfileName> "." <PropertyConstruct> "," <PropertyConstructList>

6a: <VarName>                 ::= regexp([a-zA-Z_][a-zA-Z0-9_])*)
```

Figure 6.10 Règles BNF[Backus, 1959] de désignation des variables résultats de la requête.

Dans la Figure 6.10, nous présentons l'ensemble des règles définissant la manière dont on désigne les résultats d'une requête. Les lignes 3a, 3b, 3c correspondent à la définition d'un résultat simple (nom de variable, nom de profil, nom de propriété). Les lignes 4a, 5a, 5b, 5c désignent la définition d'un résultat comportant plusieurs propriétés indiquées explicitement ou à

travers un profil sémantique. La ligne 6 indique la règle de construction d'un nom de variable à l'aide d'une expression régulière.

Ces règles viennent compléter les notations proposées par OQL. Au niveau d'OQL on peut uniquement construire des résultats en s'appuyant sur la structure statique du modèle. Ainsi, par exemple si l'on veut obtenir la couleur dominante d'un objet `Apparence`, nous devons introduire dans la requête explicitement la manière dont la couleur dominante est associée à travers une propriété à l'objet `Apparence` (voir requête OQL de la Figure 6.11). La requête 3DSEAM OQL s'appuie sur la convention que nous avons choisie afin représenter les propriétés des objets sous forme de liste de couples (nom, valeur). Ces conventions nous permettent en effet, en partant de la requête 3DSEAM OQL, de construire la requête OQL valide équivalente.

Requête OQL

Select p.value from Appearance a, a.property p
Where p.name=«*dominantColor* »

Requête 3DSEAM OQL

Select a.*dominantColor* from Appearance a

Figure 6.11 Simplification de requêtes OQL en utilisant les spécificités du modèle 3DSEAM.

6.4.2.3. Modalités de navigation au sein du modèle

En ce qui concerne la navigation au sein du modèle, en OQL elle est réalisée en traversant les différentes relations reliant les classes du modèle. Du fait que l'ensemble des associations définies entre les différentes classes du modèle n'évolue pas, les types de lien qu'il peut y avoir au sein du modèle 3DSEAM sont figés. Par conséquent l'ensemble des chemins de navigation peut être déterminé à l'avance.

- **e isA Entity**
  - e->MMFragment (or any sub-class)
  - e->Semantics
- **s isA Semantics**
  - s->MMFragment (or any sub-class)
  - s->Entity
- **sc isA Scene**
  - sc->MMFragment (or any sub-class)

…

- **mf isA MMFragment**
  - mf->MediaProfile
  - mf->MediaLocator
  - mf->Entity
  - mf->Semantics
- **o3d isA Object3D**
  - o3d->Geometry
  - o3d->Apperance
  - o3d->Topology

Figure 6.12 Navigation simple dans le modèle 3DSEAM.

Dans la Figure 6.12, nous illustrons un sous-ensemble de chemin valides pour cinq classes de notre modèle : `Entity`, `MMFragment`, `Semantics`, `Scene`, `Object3D`. Les chemins sont tous définis par rapport à une instance de la classe source. Lorsqu'il est évalué, un chemin mène à l'ensemble des instances de la classe cible qui est associé à l'instance source. La notation ne comprend pas le nom de l'association car, dans le modèle 3DSEAM tel que nous l'avons conçu, entre deux classes il y existe au plus une association et donc il n'y a pas de risque de confusion. L'association est directement déduite à partir des classes source et cible. Par exemple, lorsque pour un fragment multimédia (`mf`) on souhaite obtenir la sémantique (`mf->Semantics`), le système s'appuie sur la relation `LocalSemantics`, car c'est la seule qui relie directement les classes `MMFragment` et `Semantics`. Parmi les cibles d'une opération de navigation au sein du modèle, nous retenons également les profils sémantiques que l'on peut relier à la classe `Semantics`. Comme les profils sémantiques bénéficient d'un traitement spécial au sein du modèle, nous décidons de leur

associer une notation spéciale qui correspond à préfixer leur nom de « `Semantics!` ». Ainsi lorsque l'on veut accéder au profil sémantique `Geospatial` d'une entité à partir de cette dernière, nous utilisons l'expression suivante : `e->Semantics!Geospatial`.

La connaissance de la structure du modèle permet également d'identifier les mauvaises constructions. Par exemple, la notation `e->Geometry` est incorrecte car il n'existe aucun lien direct entre les deux classes (`Entity` et `Geometry`). Néanmoins, le langage doit permettre à l'utilisateur de mettre en relation des instances des classes dont la distance en termes de relations est supérieure à un. Cette fonctionnalité est mise en œuvre par l'enchaînement des opérateurs « -> ». Ainsi, l'expression `e->Object3D->Geometry` permet d'atteindre l'ensemble des descriptions géométriques des objets 3D matérialisant l'entité `e`.

Afin d'accéder à une propriété de l'élément ou des éléments cibles d'un chemin navigationnel, nous rajoutons à la fin de la notation de jointure le signe *point* suivi du nom local de la propriété dont la construction est régie par l'élément *PropertyConstruct* de la Figure 6.7. Les règles de la Figure 6.13 résument les modalités d'accès aux propriétés d'un objet suite à une opération de jointure.

<PathExpr>	::= <VarName> "->" <PathTarget> "." <PropertyConstruct> |
<PathTarget>	::= <ClassName> | <ClassName> "->" <PathTarget> | Semantics "!" <ProfileName>
<ClassName>	::= Semantics | Entity | MMFragment | Object3D | Scene | MediaProfile | MediaLocator | Appearence | Geometry | Topology

Figure 6.13 Règles BNF [Backus, 1959] pour la construction d'une expression de jointure.

Nous avons choisi ce type de notation pour spécifier les chemins de navigation car cela évite une écriture complexe qui suppose l'indication des rôles utilisés pour relier deux classes.

L'adoption d'un tel type simplifié de notation a cependant un certain nombre d'implication directe sur l'évaluation d'une requête. Si dans une requête nous utilisons deux chemins distincts (`obj3d->Semantics!Catégorisation.catégorie` et `obj3d->Semantics!Catégorisation.nom`) l'évaluation des propriétés ciblées (catégorie et nom) par les chemins ne se fait pas sur la même instance de la classe cible (`Semantics`). En effet, chaque jointure implique une nouvelle opération de calcul d'instances mise en relation avec l'instance source de la jointure. Au moment de l'évaluation d'une expression de jointure, une variable temporaire et anonyme est associée aux instances résultantes de la jointure. Dans le cas présent, nous proposons l'utilisation de variables intermédiaires introduites dans la clause `FROM`. Le but est d'associer à chaque résultat de navigation que l'on veut réutiliser à d'autres endroits dans la requête, une variable intermédiaire. Une variable peut être utilisée à plusieurs endroits sans que cela induise de nouvelles opérations de jointures.

Pour l'exemple que nous avons introduit ci-dessus, nous définissons une variable alias `sem` de la sorte: `FROM Object3D obj3D, obj3d->Semantics sem`. Ceci permet de formuler des critères qui portent sur les propriétés `sem!Catégorisation.catégorie` et `sem!Catégorisation.nom` d'une même instance.

Nous envisageons le même type d'approche pour l'exploitation des relations sémantiques et topologiques qui peuvent être instanciées au sein de l'entrepôt. Dans la clause `FROM` on peut définir une variable qui correspond aux instances mises en correspondance avec l'instance source à travers une relation spécifique. Ainsi, la notation `From obj3d->Topology.contains containedObj` introduit une nouvelle variable `containedObj` qui peut être par la suite utilisée afin de référencer l'ensemble des fragments multimédia contenu spatialement dans l'objet `obj3d`. Le

type d'instances référencées est indiqué par la valeur `targetClass` de la définition du descripteur de relation comme présenté dans le modèle de descripteurs de 3DSEAM (voir section 5.9).

6.4.2.4. Critères de recherche

Les conventions de notations que nous avons introduites nous permettent d'envisager la formulation de critères de recherche. Les critères de recherche se trouvent au niveau de la clause WHERE.

Dans la Figure 6.14 nous présentons les règles qui définissent les possibilités de formulation d'un critère de recherche. Comme l'indique la première règle, le critère simple peut être assemblé au sein de constructions complexes au moyen d'opérateurs booléens binaires (*and, or, xor*) ou unaires (*not*). Les critères simples peuvent être exprimés soit par une comparaison entre deux expressions, soit par le résultat d'une fonction booléenne dont le ou les paramètres correspondent aux expressions construites avec les éléments de la requête. Une expression peut désigner : le nom d'une variable résultat, le nom d'une variable intermédiaire, une expression de jointure, une fonction appliquée à une ou plusieurs autres expressions, une opération portant sur deux expressions.

```
<Criteria>      ::= <Criterion> | "(" <Criteria> ")" <BoolOp> "(" <Criteria> ")" |
                    not "(" <Criteria> ")"
<Criterion>     ::= <Expr> <CompOp> <Expr> | <BoolFunct> "(" <Expr> ")" |
                    <BoolFunct> "(" <ExprList> ")"

<Expr>          ::= <VarName> | <SimpleResultExpr> | <JoinExpr> |
                    <FunctName> "(" <Expr> ")" | <FunctName> "(" <ExprList> ")"
                    "(" Expr ")" <BinaryOp> "(" <Expr> ")" | $<ParamName>
<ExprList>      ::= <Expr> "," <ExprList>

<ParamName>     ::= <Name>
<FunctName>     ::= <Name> | <NameSpace>::<Name>
<BoolFunct>     ::= <Name> | <NameSpace>::<Name>

<BinaryOp>      ::=  <CompOp> | <ArithmOp>
<CompOp>        ::=  "==" | "<" | ">" | "<=" | ">="
<ArithmOp>      ::=  "+" | "-" | "/" | "*"
```

Figure 6.14 Règles BNF [Backus, 1959] pour la construction de critères de recherche en 3DSEAM.

Les noms de fonctions peuvent être précédés par un espace de noms. Chaque plate-forme a pour responsabilité de tenir à la disposition de ses utilisateurs une liste complète des fonctions spécifiques supportées par le moteur d'interrogation.

L'intégralité des règles BNF, régissant la construction d'une requête 3DSEAM OQL, est présentée à l'adresse suivante :

`http://www-lsr.imag.fr/Les.Personnes/Marius.Bilasco/these/annexes/3DSEAM_OQL.g`

6.4.2.5. Exemple de requêtes 3DSEAM

Dans la *Figure 6.15*, nous présentons quatre requêtes 3DSEAM qui mettent en évidence différents niveaux d'expression du langage que nous proposons.

La première requête correspond à l'extraction de la propriété `s!Catégorisation.catégorie` liée à la sémantique d'une entité (`s->Entity`) associée à un Object3D (`s->Entity->Object3D`) connu par son identifiant.

La seconde demande les objets (`obj3d`) de couleur grise (`obj3d->Appearance.dominantColor=GRAY`) ainsi que leur profil géospatial et leur catégorie (`s{!Geospatial, !Catérgorisation.catégorie}`). La condition `obj3D->entity.Semantics.id=s.id` a pour rôle d'imposer que l'extraction des informations se fasse par rapport au même objet sémantique.

La troisième requête concerne l'extraction des repères `MediaLocator` qui définissent les endroits où se trouvent des objets 3D qui appartiennent aux catégories sémantiques « bâtiment » ou « étage ». Cette requête nécessite l'utilisation d'un alias pour associer directement la sémantique aux repères de localisation. La variable intermédiaire `s` introduite dans la clause `FROM`, correspond à l'ensemble des instances de la sémantique (`ml->MMFragment->Entity->Semantics`) associées aux entités (`ml->MMFragment->Entity`) représentant l'objet localisé (`ml->MMFragment`) par le repère de localisation initial (`ml`).

1) Trouver les catégories (du profil Catégorisation) pour un object 3D dont l'identifiant est $FRAG_ID.

```
SELECT s!Catégorisation.catégorie FROM Semantics s
        WHERE s->Entity->Object3D.id=$FRAG_ID
```

2) Trouver tous les objets de couleur grise, leur profil géospatial et leur catégoire.

```
SELECT obj3d, s{!Geospatial, !Catégorie.catégorie} FROM 3DObject obj3D, Semantics s
    WHERE (obj3d->Entity.Semantics.id=s.id) and
        (obj3d->Apperance.dominantColor=GRAY)
```

3) Trouver la localisation des objets qui appartiennent aux catégories « bâtiment » ou « étage ».

```
SELECT ml FROM MediaLocator ml, ml->MMFragment->Entity->Semantics s
    WHERE (s!Catégorisation.catégorie="bâtiment") or
        (s!Catégorisation.catégorie="étage")
```

4) Trouver les bâtiments rouges à quatre étages situés sur le campus universitaire.

```
SELECT ml
    FROM   Object3d obj3d,
        obj3d->MediaLocator ml, obj3d->Entity->Semantics s,
        obj3d->Appearance app, obj3d->Topology.containsSection relatedObj
    WHERE
        (s!Catégorisation.catégorie="bâtiment") and (s!BâtimentDesc.nb_etages=4) and
        (app.dominantColor=RED) and (relatedObj.id=CAMPUS_UNIV)
```

Figure 6.15. Requêtes 3DSEAM formulées en 3DSEAM OQL.

La quatrième requête illustre, en plus des mécanismes de navigation et d'introduction d'alias, l'utilisation de critères portant sur plusieurs dimensions descriptives, ainsi que l'exploitation des relations (ici, topologiques) définies entre les instances du modèle. Plus précisément, la requête porte sur les repères associés aux objets 3D qui satisfont un ensemble de critères portant sur leur apparence `app.dominantColor=RED`, sur leur sémantique `sem!Catégorisation:catégorie="bâtiment"`, `sem!BâtimentDesc.nb_etages=4` et sur les relations topologiques (`containsSection`) entretenues avec un autre objet 3D représentant la parcelle correspondant au campus universitaire. Ici, nous supposons qu'une entité appartenant à la catégorie *bâtiment* est associée également à un profil sémantique décrivant les propriétés de bâtiments (`BâtimentDesc`). La variable `app` est associée aux instances reliées à l'objet 3D (`obj3d`) localisé par la variable `ml (obj3d->MediaLocator ml)`. La variable `sem` référencie les instances sémantiques reliées à l'objet 3D à travers les entités qu'il matérialise `obj3d->Entity->Semantics sem`. La variable `relatedObj` est définie comme l'ensemble d'objets 3D `obj3d->Topology.containsSection relatedObj` reliés à l'objet initial par la relation topologique *containsSection* que nous avons introduit dans la section 5.7.

La formulation de certaines requêtes présentées ici n'est pas triviale. Même si nous en avons simplifié l'écriture, au moyen de jointures simplifiées et de la navigation dans les relations sémantiques et topologiques, le langage reste principalement un outil destiné aux concepteurs d'applications au-dessus du modèle 3DSEAM et non pas aux utilisateurs novices. C'est aux applications de traduire les besoins des utilisateurs en requêtes adaptées à l'entrepôt de description sous-jacent.

6.4.2.6. Le format de sortie des résultats

Afin de pouvoir envisager la collaboration entre plusieurs plates-formes nous imposons un format de sortie pour les résultats d'une requête.

Les résultats (informations sémantiques, informations sur l'entité ou sur un fragment) sont transformés dans un format auto-descriptif structuré par XML. Chaque résultat correspond à un *item*. Un *item* peut contenir plusieurs *éléments*. Chaque *élément* est défini par rapport aux notions 3DSEAM en utilisant l'attribut *type*. L'attribut *type* correspond au nom long de la propriété (tel défini dans la Figure 6.7) dont la valeur est représentée par le contenu de l'*élément.* Une illustration du format est donnée dans la Figure 6.16.

```
<items>
  <item>
    <element type="Semantics!Catégorisation.catégorie">
        bâtiment
    </element>
  </item>
  …
</items>
```

Figure 6.16 Réponse XML auto-descriptive à la première requête de la Figure 6.15.

Nous avons choisi cette solution XML pour représenter les résultats de la requête du fait des larges possibilités reconnues de XML pour partager et échanger les données entre différents acteurs humains ou logiciels. Du plus, les efforts de la communauté XML, et notamment du consortium W3C autour d'un encodage binaire des fichiers XML [Goldman *et al.*, 2005], laisse entrevoir la disparition d'un des désavantages majeurs de l'encodage XML textuel lié au temps conséquent de transmission pour les grands volumes de données.

Nous cherchons également, avec l'adoption de XML comme format fédérateur des résultats, un moyen homogène d'encoder les différents types de résultats d'une requête. L'extrait présenté dans la Figure 6.16 correspond aux résultats d'une requête qui renvoie des valeurs de propriétés. Cependant, les requêtes 3DSEAM SQL peuvent produire des résultats combinant plusieurs types d'éléments : valeurs de propriétés, collection de propriétés, instances. Les conventions de notation introduite dans ce chapitre, et notamment dans les Figure 6.7 et Figure 6.10, nous permettent d'associer à chaque type résultat une notation unique.

Afin d'illustrer ceci, nous présentons dans la Figure 6.17 une partie des résultats de la deuxième requête de la *Figure 6.15* sur les instances de l'entrepôt de la Figure 6.4. Chaque résultat comporte un élément de type instance, un élément de type profil sémantique et un élément de type propriété. L'encodage d'une instance suppose l'encodage des sous-éléments présentant l'ensemble des propriétés de l'instance suivant le modèle présenté précédemment. L'attribut *type* de l'élément représentant l'instance indique le nom de la classe à laquelle l'instance appartient. Ainsi pour l'instance d'`Object3D` présentée dans la Figure 6.17, on retrouve l'ensemble de ses propriétés (son identifiant, l'identifiant de l'entité lui étant associé,

l'identifiant de son repère de localisation, etc.). L'encodage d'un profil sémantique se fait sur le même principe, en incluant, à l'aide des sous-éléments, la totalité des propriétés le référençant.

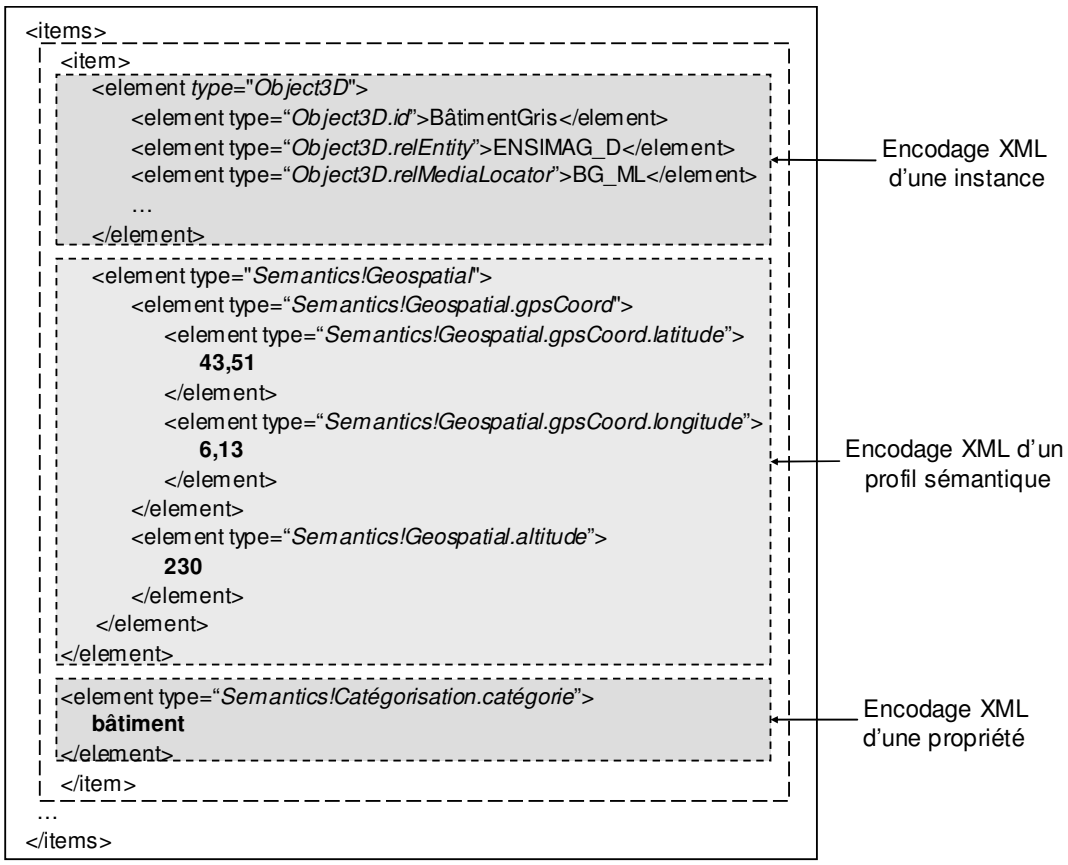


Figure 6.17 Réponse XML auto-descriptive de la deuxième requête de la Figure 6.15.

6.5. Synthèse

Dans ce chapitre nous avons présenté une plate-forme de gestion (3DAF) des entrepôts de description de type 3DSEAM. Cette plate-forme est munie de modules spécifiques pour l'ajout et l'exploitation de connaissances dans l'entrepôt de description 3DSEAM. L'accès aux fonctionnalités de ces modules se fait à travers des notations capables de tenir compte de l'hétérogénéité qu'il peut y avoir en termes de propriétés au niveau des instances. Des conventions de noms ont été introduites afin d'adresser les instances et les types de contenus de l'entrepôt de manière unique. Ces conventions nous ont notamment permis de construire des méthodes permettant d'alimenter l'entrepôt et d'envisager la construction d'un langage de requêtes pour exploiter les connaissances stockées dans l'entrepôt. Une extension du langage de requête OQL a été proposée pour le modèle 3DSEAM. Afin de simplifier l'écriture des requêtes, nous avons introduit un ensemble de notations (écriture simplifiée de jointures, accès aux éléments d'un profil sémantique, etc.) en étroite relation avec l'organisation du modèle 3DSEAM. L'extension du langage OQL permet aux utilisateurs (humain ou logiciel) de la plate-forme d'en exploiter le contenu sans avoir à gérer explicitement l'éventuelle diversité des représentations des connaissances dans chaque entrepôt ou base de connaissances. Nous considérons qu'il est important d'œuvrer pour cette indépendance car, en termes de représentation de connaissances, il n'y a pas une solution qui convienne plus qu'une autre pour tout domaine et contexte particulier d'application.

Par rapport, aux travaux que nous avons présentés dans l'état de l'art, cette plate-forme envisage d'utiliser la sémantique et autres caractéristiques des objets 3D dans un cadre plus large que celui d'une application spécifique. L'exploitation, et plus particulièrement la réutilisation, des descriptions sémantiques est au cœur de nos préoccupations.

Cette plate-forme constitue une base, au-dessus de laquelle, des solutions spécifiques aux différents domaines d'application de la 3D peuvent alors être utilisées pour exploiter les connaissances sémantiques et les caractéristiques relatives aux données 3D. Dans ce sens, dans les chapitres suivants nous présentons deux applications construites au-dessus de cette plate-forme s'intéressant respectivement à la réutilisation de scènes sur critères sémantiques et à l'adaptation des scènes au moyen de règles. **Ayant introduit, à travers cette plate-forme 3DAF, un niveau d'abstraction entre la couche fonctionnelle et la représentation des connaissances, les applications que nous présentons par la suite ne font aucune référence au type de représentation utilisée.**

7. Exploitation de la sémantique dans la réutilisation de scènes 3D

L'ensemble des informations réunies au sein du modèle 3DSEAM peut répondre au besoin, en termes d'annotations sémantiques, à de nombreuses applications manipulant des scènes 3D. Toutefois en l'état, les informations qui peuvent être extraites du modèle 3DSEAM en utilisant la plate-forme 3DAF, sont purement textuelles et dissocient la sémantique de la géométrie. En fonction des besoins applicatifs spécifiques, un certain nombre de couches de pré-traitements sont nécessaires pour rendre utilisables les informations fournies par 3DAF.

Par exemple, les applications 3D à base de scripts, telles que [Hetherington *et al.*, 2004] et [Polys, 2003], s'appuient sur les métadonnées qui sont incluses dans la scène 3D. Ainsi, les informations brutes (géométrie et sémantique) doivent être assemblées et mises en relation au sein d'une même scène 3D. L'indépendance qui a été acquise en séparant la géométrie et la sémantique au sein de 3DSEAM permet de combiner différents types de profils sémantiques avec une même description géométrique. Cela permet de répondre au besoin de plusieurs applications construites sur une même scène 3D.

Nous voulons permettre au concepteur d'une application, demandeur d'un certain type d'informations sémantiques, d'obtenir des scènes 3D prêtes à l'emploi incluant la sémantique en tant que métadonnées associées aux objets annotés.

Nous proposons une solution logicielle, qui permet d'attacher de la sémantique à la géométrie au sein d'une scène 3D (plus précisément X3D), et qui répond aux besoins des concepteurs d'applications. Il s'agit d'une bibliothèque de données sémantiques spécialisée dans les contenus 3D. Appelée 3DSDL (*3D Semantic Data Library*), cette bibliothèque doit permettre :

a) d'interroger les instances 3DSEAM afin de retrouver des objets 3D ou d'obtenir des informations concernant les objets 3D répertoriés, et

b) d'offrir des scènes 3D annotées convenablement pour des applications 3D aux besoins sémantiques spécifiques.

La solution que nous proposons ici cible les scènes X3D car ce standard 3D est l'un des seuls qui mettent les métadonnées associées aux objets 3D au premier rang. Tous les éléments définis par le standard peuvent comporter une description à base de métadonnées types. Les informations nécessaires sont portées par les nœuds `MetadataSet` attachés aux primitives géométriques modélisant les objets.

7.1. Analyse des besoins d'une plate-forme de réutilisation

Parmi les fonctionnalités que nous souhaitons placer au sein de cette plate-forme de réutilisation, nous pouvons isoler deux grandes catégories : celles qui concernent la réutilisation des fragments et celles qui s'intéressent à la réutilisation des aspects sémantiques associées aux données à travers des métadonnées.

La réutilisation des fragments suppose deux actions distinctes :

a) la récupération des fragments de la scène concernés par la réutilisation, et

b) l'inclusion des fragments récupérés au sein de la nouvelle scène.

Le processus de récupération du fragment peut être, à son tour, décomposé en deux sous processus :

- l'un concerné par la recherche effective des fragments par rapport à une description de ses propriétés, et
- l'autre concerné par l'extraction du code représentant effectivement le contenu à partir de documents contenant l'objet.

La recherche effective des fragments correspondant aux critères de réutilisation imposés par un client est prise entièrement en charge par la plate-forme 3DAF qui donne accès à l'ensemble des informations dont on dispose sur les scènes enregistrées auprès de la plate-forme.

Une étape préliminaire au processus de réutilisation est de faire connaître aux clients l'ensemble des types d'information disponible sur un objet ou une scène en particulier. Ces informations sont toutes répertoriées dans le modèle de descripteurs de 3DSEAM associé à l'entrepôt ou aux bases de connaissances sous-jacentes. La publication de ce schéma sous la forme d'une ontologie de description nous semble pertinente.

Afin de mieux s'adapter au niveau d'expertise de chaque client (utilisateur ou logiciel), la formulation des critères de recherche doit être indépendante des conventions de représentation 3DSEAM. Lorsque les clients formulent leurs souhaits de sélection, la formulation des critères de recherche doit se faire le plus naturellement possible pour eux. Ainsi, il est envisagé de supporter dans la plate-forme des fonctions de sélections spécifiques au domaine d'application visé par la plate-forme. Néanmoins, au moment de la conception de la plate-forme, il est impossible d'inclure dans le moteur d'évaluation des requêtes un ensemble exhaustif d'opérateurs. Pour trouver une solution à ce problème, nous envisageons l'enregistrement auprès de la plate-forme de modules externes capables d'interagir avec la plate-forme et d'interpréter des opérateurs spécifiques concernant l'extraction du contenu.

Du fait que nous nous trouvons dans un contexte de réutilisation, les recherches visent à récupérer une collection d'objets (fragments multimédia dans 3DSEAM) qui répondent aux critères formulés par le client de la plate-forme. Si le client demande également à inclure, dans le

document 3D résultat, de la sémantique sous la forme de métadonnées sur les objets retrouvés, des requêtes pour extraire la sémantique sont effectuées. Nous décidons de séparer les deux fonctionnalités suivantes : *la récupération des identifiants d'objets* et *la récupération de propriétés sémantiques*. Cette séparation est guidée par un souci de simplification de la formulation des requêtes d'extraction de contenu. Les utilisateurs de la plate-forme ne sont pas forcément ni des experts de langages de requêtes, ni supposés connaître l'organisation interne de l'entrepôt. Les clients n'indiquent ni la source, ni la manière dont le contenu est extrait. Un client doit pouvoir fournir uniquement une liste de critères auxquels les résultats doivent répondre. La plate-forme doit : mettre en place des patrons de requêtes permettant l'extraction de propriétés, respectivement d'identifiants et de repères de localisation.

En ce qui concerne la tâche d'extraction du code, la plate-forme de réutilisation doit être capable d'évolution. Le nombre de type d'encodages et de langages supportés par la plate-forme influe de manière certaine sur le degré d'utilisation et le succès de celle-ci. Ainsi, nous considérons que la plate-forme doit aborder ce problème en mettant en place des mécanismes permettant d'envisager facilement l'évolution des formats supportés. La délégation de l'activité d'extraction de fragments aux modules externes qui peuvent s'enregistrer dynamiquement auprès de la plate-forme semble une approche prometteuse.

L'inclusion du fragment au sein d'un nouveau document exige une attention particulière car une grande variété de situations peut être envisagée :

- la position dans le graphe de scène est connue à l'avance ;
- les coordonnées de la position absolue dans la nouvelle scène de l'objet sont connues à l'avance ;
- le fragment fait partie d'une scène de présentation des résultats d'une requête ;
- le fragment est à insérer dans la scène en fonction des relations sémantiques et topologiques entretenues avec les autres entités, etc.

Dans cette proposition nous explorons uniquement les trois premières situations. La dernière situation demande à elle seule, un travail conséquent sur la traduction des relations sémantiques en relations spatiales, l'adoption des métaphores de visualisation [Averbukh *et al.*, 2007], l'analyse de leur pertinence, etc. Cependant, la proposition de descriptions sémantiques que nous défendons ici apporte certaines réponses à ce problème.

L'analyse des besoins et des types de solutions envisageables oriente nos recherches vers une architecture formée d'un noyau de réutilisation autour duquel s'agence un ensemble de modules externes. Le noyau coordonne les activités autour du processus de réutilisation, tandis que les modules capables de répondre à des besoins spécifiques liés à un domaine d'application particulier remplissent des tâches auxiliaires.

Dans la section suivante, nous présentons l'architecture générale de notre approche, en insistant sur le rôle de chacun des modules du noyau et sur le protocole de communication avec les modules externes.

7.2. Architecture logicielle de 3DSDL

L'architecture de 3DSDL est présentée dans la Figure 7.1. Comme indiqué dans la section précédente, la plate-forme est composée d'un noyau central et d'un ensemble de répertoires référençant l'ensemble des modules externes contribuant au bon fonctionnement de la plate-forme.

Les principaux composants du noyau sont :

- l'*interface de communication* qui fait le lien entre les clients et le noyau d'exécution ;
- les modules d'interrogation *Récupération de propriétés* et *Récupération d'objets* caractérisés par leur identifiant et par leur repère de localisation ;
- le module *Extraction d'objets 3D* qui récupère le *code 3D* d'un objet défini par un repère de localisation 3DSEAM ;
- le module *Attachement de la sémantique* – dont le rôle est d'injecter de la sémantique dans les fragments 3D ;
- le module *Assemblage de scène* qui décide de l'organisation spatiale des objets au sein de la nouvelle scène ;
- le module *Contrôleur de réutilisation* – dont le rôle est de diriger le processus de réutilisation.

Cette séparation entre les fonctionnalités offre une grande capacité d'évolution de la plate-forme de réutilisation de scènes 3D. Des fragments issus de documents de natures différentes peuvent être intégrés au sein d'un même document résultat. De nouveaux types d'assemblage de résultat peuvent être proposés sans que les autres modules de noyaux en soient affectés. De même, la prise en compte de nouveaux critères de recherche n'affecte en rien l'extraction de contenu.

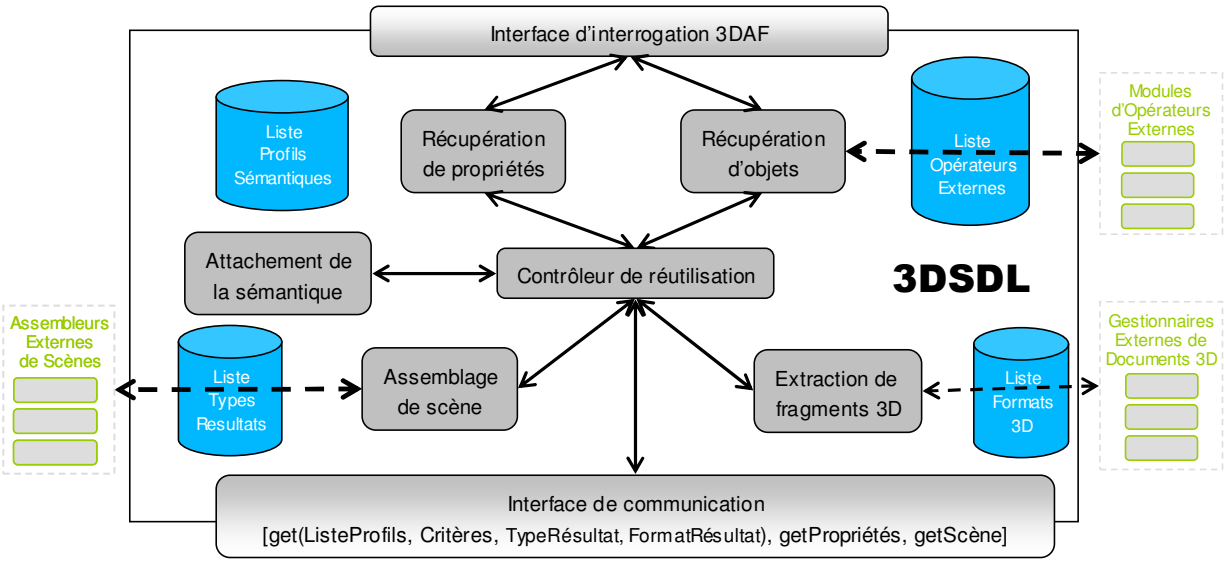


Figure 7.1 Architecture générale de 3DSDL.

Nous adoptons le standard X3D comme langage pivot de représentations internes 3D. Ce choix est motivé par la facilité de transformer les documents disposant d'un encodage XML à l'aide de langages tels que XSLT.

Le lien entre les modules du noyau et les modules externes se fait grâce aux informations stockées dans les répertoires. Ces liens sont illustrés dans la Figure 7.1 en pointillés. Nous avons intentionnellement dessiné les traits à travers les répertoires afin de symboliser le fait que ce sont les informations qu'ils contiennent et qui permettent de faire cette association. Chaque type de module externe (assembleur de scènes – *Assembleurs externes de scènes*, interpréteur de critères – *Modules d'opérateurs externes*, gestionnaire de documents – *Gestionnaires externes de*

documents 3D) doit implémenter une interface et un protocole spécifique de communication afin de pouvoir s'enregistrer dans l'architecture 3DSDL.

Dans la suite de cette section, nous présentons en détail les fonctionnalités des modules composant l'architecture proposée. Pour chaque module du noyau reliant un module externe, nous présentons les exigences de descriptions que nous imposons pour l'enregistrement du module externe auprès du système.

7.3. Le module d'interface de communication

Lors d'une requête de scène sémantique, l'utilisateur indique :

a) les objets qu'il souhaite inclure dans la scène, et

b) les profils sémantiques qu'il souhaite associer aux objets 3D assurant le rendu visuel de l'application.

L'*interface de communication* analyse la fonctionnalité de la bibliothèque demandée par l'utilisateur. Suivant la nature de la requête, le module d'interface :

- renvoie la liste des profils sémantiques disponibles dans la plate-forme lorsque le client émet une requête de type *getListeProfils* ;
- renvoie la liste des critères externes supportés par les modules d'interrogation lorsque le client émet une requête de type *getCritères* ;
- renvoie la liste des types de résultats supportés par le module d'assemblage de scène lorsque le client émet une requête de type *getTypeRésultat* ;
- renvoie la liste de format de sorties des résultats supportés par le module d'assemblage de scène lorsque le client émet une requête de type *getFormatRésultat* ;
- renvoie la liste des propriétés requises par l'utilisateur par rapport à un identifiant d'objet connu lorsque le client émet une requête de type getPropriétés ;
- enfin, le module vérifie tout d'abord que les critères, les propriétés utilisées pour formuler le critère de sélection, les propriétés sémantiques à inclure dans le résultat, le type et le format du résultat sont supportés par la plate-forme lorsque le client émet une requête de type *getScène*. Si oui, la demande est transférée au module dirigeant le processus de réutilisation (*Contrôleur de réutilisation*). Dans le cas contraire, en l'état de nos développements, une réponse négative est renvoyée à l'utilisateur. Cependant, il est envisageable de mettre en œuvre des mécanismes de recherche d'équivalence entre les propriétés.

Dans la suite de cette section, nous présentons les choix faits en ce qui concerne la publication des profils, des critères, des types et des formats de résultats supportés par la plate-forme 3DSDL. Ensuite, nous présentons des patrons permettant de décrire les critères de réutilisation. Ces patrons servent à générer automatiquement des requêtes envoyées à la plate-forme 3DAF. Du même, nous présentons une notation permettant d'indiquer les types de propriétés sémantiques à inclure dans la scène finale par rapport aux différentes catégories d'objets satisfaisant les critères de réutilisation.

7.3.1. Publication des fonctionnalités de la plate-forme

La diffusion des informations relatives aux profils, aux critères, aux types et aux formats des résultats est réalisée au moyen de fichiers XML. Dans la Figure 7.2 nous illustrons la description d'un profil sémantique (`Geospatial`), d'un opérateur (`contains`), de la liste des formats (X3D en encodage XML et X3D en encodage binaire) et des types de méthodes d'assemblage de nouvelles scènes supportées.

La description d'un profil sémantique comporte la définition de la liste des propriétés lui appartenant. Chaque propriété est décrite, conformément au modèle des descripteurs de 3DSEAM (voir section 5.9), par l'alias au niveau du profil, par son nom, par son lien avec une ontologie et par le type de la valeur.

Les opérateurs sont définis de manière similaire à la description des propriétés. Nous indiquons un alias au niveau de la plate-forme, le nom entier de l'opérateur, le lien avec une ontologie le décrivant et la valeur résultat. Chaque opérande est introduit par une balise `element` qui indique l'alias de l'opérande et le type qu'il doit avoir. Pour l'opérateur présenté dans la partie droite de la Figure 7.2, les opérandes doivent être de type `Object3D`.

```
<profiles>
    <profile name="Geospatial">
        <property alias="gpsCoord"  name="geo:gpsCoord"
            ontoRes="http://lig.imag.fr/Ontos/Geospatial#gpsCoord"
             valueType="default::StructureType">
                <property alias="latitude"  name="geo:latitute"
                    ontoRes="http://lig.imag.fr/Ontos/Geospatial#latitude"
                     valueType="default:: DoubleType"/>
        </property>
    </profile>
    …
</profiles>
```

Description des profils sémantiques

```
<operators>
    <operator alias="contains"  name="geo:contains3d"
        ontoRes="http://lig.imag.fr/Ontos/Geospatial#contains"
         value="default::Boolean">
            <description>true if x contains y </description>
            <element alias="x"
                requiredType="_3dseam:Object3D" />
            <element alias="y"
                requiredType="_3dseam:Object3D"/>
    </operator>
    …
</operators>
```

Description des opérateurs supportés

```
<formats>
    <format name="X3D_XML" >
        <description>XML encoding of the X3D standard</description>
    </format>
    <format name="X3D_Binary"/>
        <description>binary encoding of the X3D standard</description>
    </format>
    …
</formats>
```

Liste des formats 3D supportés

```
<reuseTypes>
    <reuseType name="IN_PLACE">
        <description>keeping absolute coordinates</description>
    </reuseType>
    <reuseType name="COLLECTION">
        <description>exposing objects</description>
    </reuseType>
    …
</reuseTypes>
```

Liste des méthodes d'assemblage supportées

Figure 7.2 Exemple de description XML de profils, opérateurs, formats et méthodes d'assemblage de scène supportés par la plate-forme de réutilisation.

Les deux méthodes d'assemblage décrites dans la Figure 7.2 correspondent à deux techniques basiques de réutilisation :

- la conservation des coordonnées absolues des objets dans la scène générée (*IN_PLACE*), et
- la génération d'un espace d'exposition de chaque objet correspondant aux critères (*COLLECTION*).

Ce type de description permet la génération automatique d'interfaces visuelles en vue d'offrir aux utilisateurs novices un moyen d'accès aux outils de réutilisation. Une illustration de ceci est réalisée dans le chapitre consacré à la validation expérimentale de notre proposition (chapitre 0). Après avoir décrit la manière dont les clients peuvent s'informer sur le type d'opérateurs et profils sémantiques supportés, dans la sous-section suivante nous présentons les moyens de formuler des requêtes de réutilisation.

7.3.2. Formulation des requêtes de réutilisation

Afin de connaître les éléments à inclure dans la scène demandée par le client, celui-ci doit indiquer obligatoirement, les critères que les objets à inclure dans la scène doivent satisfaire, et de façon optionnelle, la sémantique qu'il veut reporter dans la scène, le type de résultat et le format de diffusion. Nous adoptons une méthode de formulation simplifiée des requêtes afin de faciliter leur construction par des utilisateurs disposant de faibles connaissances sur l'organisation de l'entrepôt et du langage de requête de 3DAF. Nous pouvons faire cela car toutes les requêtes présentées ici portent sur l'extraction des identifiants et des repères spatiaux associés aux fragments multimédia.

Les critères servent à définir les catégories d'objets à inclure dans la scène finale. Une catégorie d'objets est décrite par un ensemble de critères qui portent sur les valeurs des propriétés associées aux objets 3D. Les propriétés sont désignées par leur *nom long* comme indiqué dans la section 6.3.2, à l'exception des propriétés issues de la classe `Semantics`. Le modèle comprend deux types d'information sémantique (locale, générale) associés à un fragment média. Nous préfixons les propriétés locales par le nom `LocalSemantics,` pour désambiguïser la notation.

Lorsque des contraintes doivent être imposées entre les instances des diverses catégories, il est nécessaire de séparer leur définition à l'aide de critères, en introduisant explicitement de nouvelles catégories d'objets. Par rapport à ces catégories, nous pouvons définir des critères de sélection similaires à ceux définis au niveau des objets de chaque catégorie. L'évaluation du critère se fait entre chaque n-uplet composé d'instances. Les n-uplets validant le critère sont gardés dans le résultat final.

<ReuseQuery>	::= "{" <ObjectCategories> "}" | <BoolFunct> ("{" <ObjectCategories> "}" | <Funct> ("{" <ObjectCategories> "}") <arithmOp> <Expr>
<ObjectCategories>	::= ObjectCategory, ObjectCategories | ObjectCategory
<ObjectCategory>	::= "["<Criteria>"]"
<Criteria>	::= (<Criterion> <boolOp> <Criteria>) | <Criterion>
<Criterion>	::= (<Expr> <arithmOp> <Expr>) | <BooleanFunct> (<Expr>)
<Expr>	::= <LongName> | <Funct> "(" <LongName> ")" | <Litteral>

Figure 7.3 Grammaire BNF [Backus, 1959] pour la construction d'une requête de réutilisation.

La grammaire définissant les possibilités de formulation des requêtes est disponible dans la Figure 7.3. La clause `ReuseQuery` définit le critère global de sélection. Entre accolades, on retrouve l'ensemble des catégories d'objets (`ObjectCategories`) de la requête, précédé des éventuels opérateurs qui portent sur les individus de chaque catégorie. La définition de chaque catégorie (`ObjectCategory`) se fait par rapport à un ensemble de critères (`Criteria`) portant sur les valeurs des propriétés entourées par des parenthèses droites. Les critères sont formulés en suivant les mêmes règles de description (voir section 6.4.2.4) que dans le cas de la plate-forme 3DAF, exception faite de l'absence des moyens permettant de décrire des jointures et de faire référence aux variables.

Afin d'illustrer les modalités de construction d'une requête de réutilisation, nous présentons dans la Figure 7.4 trois requêtes illustrant les différents types de requête supportées :

- la première considère une seule catégorie d'objets et vise l'obtention d'une scène contenant l'ensemble des bâtiments et des routes ;
- la deuxième présente l'intégration d'un opérateur externe (issu du domaine géospatial) ;

- et la troisième illustre l'utilisation d'un critère au niveau des catégories d'objets afin d'en retenir uniquement quelques-uns.

{[(Semantics!Catégorisation.catégorie=« bâtiment ») || (Semantics!Catégorisation.catégorie=« route »)]}
=> sélection de tous les bâtiments et routes enregistrés dans l'entrepôt

{[(Semantics!Catégorisation.catégorie=« bâtiment ») || (inRange(Semantics!Geospatil.gpsCoord,'7.32E 40.3N','1km')]]}
=> sélection de tous les bâtiments dans un rayon de 1km autour du point de coordonées de 7.32E et 40.3N

distanceM({[(Semantics!Catégorisation.catégorie=« bâtiment »)],[(Semantics!Catégorisation.catégorie=« arbre »)]]}) < 20
=> sélection des bâtiments et arbres se trouvant à une distance inférieure de 20 m, l'un de l'autre

Figure 7.4 Exemples de requêtes de réutilisation.

7.3.3. Description des propriétés sémantiques

Par l'intermédiaire de cette plate-forme, nous visons la réutilisation de scènes enrichies d'informations sémantiques incluses en tant que métadonnées dans le document correspondant à la nouvelle scène générée. Comme nous l'avons souligné dans la sous-section précédente, nous pouvons définir plusieurs catégories d'objets. De la même façon, un client de la plate-forme peut demander à ce que l'on personnalise les types d'information sémantique pour chaque catégorie introduite dans la requête de réutilisation.

Une notation similaire à celle choisie pour la requête de réutilisation à base de listes, est adoptée (voir Figure 7.5). Entre accolades, nous introduisons les différents ensembles de propriétés sémantiques (`SemanticGroups`) associées aux catégories de la requête d'utilisation. Ensuite chaque groupe de propriétés (`SemanticGroup`) est défini comme une liste de propriétés (`PropList`), chacune d'entre-elles étant identifiée par le *nom long* (`LongName`). La correspondance entre les catégories d'objets et les listes de propriétés introduites ici, se fait directement par rapport à la position dans la liste : les objets de la première catégorie intègrent l'ensemble des propriétés décrites à la première position dans la liste des groupes de propriétés sémantiques. Par exemple, si nous associons l'expression `{[Semantics!Geospatial.gpsCoord, Semantics!Geospatial.altitude]}` à la dernière requête de la Figure 7.4, chaque bâtiment introduit dans la scène est décrit par ses coordonnées GPS et son altitude, tandis que les arbres ne portent aucune information sémantique additionnelle.

<SemanticInfos>	::= "{" <SemanticGroups> "}"
<SemanticGroups>	::= <SemanticGroup>, <SemanticGroups> | <SemanticGroup>
<SemanticGroup>	::= "["<PropList>"]" | "[]"
<PropList>	::= <LongName>, <PropList> | <LongName>

Figure 7.5 Grammaire BNF [Backus, 1959] pour la définition de propriétés sémantiques à associer aux catégories d'objets.

Dans la suite, nous présentons séparément les modules responsables de la mise en œuvre des différentes fonctionnalités exigées, pour aboutir à la construction d'une nouvelle scène sémantique. Nous commençons par ceux assurant les opérations de base (recherche, extraction du code, injection de la sémantique, assemblage) pour présenter à la fin le module orchestrant la réutilisation.

7.4. Les modules de recherche d'information

Les principaux rôles des deux modules sont de traduire la description des *pseudo* requêtes – que nous avons présentées dans la section précédente concernant la réutilisation et la sémantique – en requêtes interprétables par le langage 3DAF et de les transférer au gestionnaire de requêtes 3DAF afin de les exécuter.

7.4.1. Le module de récupération d'objets

Le module *Récupération d'objets* permet de retrouver les repères de localisation d'un objet 3D à partir d'une description signalétique couvrant les dimensions sémantique, géométrique, topologique et l'apparence de l'objet.

La requête d'utilisation est transformée automatiquement en requête 3DSEAM en suivant le patron de la Figure 7.6. Pour chaque catégorie d'objets (`obj1`, …, `objN`), nous demandons l'extraction des `id` et des repères de localisation `MediaLocator` des `Objets3D`. À partir des *noms longs* des propriétés (par exemple, `Semantics!Geospatial.gpsCoord`) utilisés dans les critères de sélection pour chaque catégorie d'objets, nous introduisons des alias (`obj1->Entity->Semantics sem1_1`) pour référencer les instances du modèle en rapport avec les objets 3D qui contiennent les propriétés requises (`Geospatial.gpsCoord`). L'ensemble des critères relatifs à chaque catégorie est traduit en remplaçant les préfixes des propriétés (`Semantics!Geospatial.gpsCoord`) par l'alias de jointure (`sem1_1!Geospatial.gpsCoord`). Finalement, nous reportons dans la clause `WHERE`, les critères portant sur l'ensemble des objets.

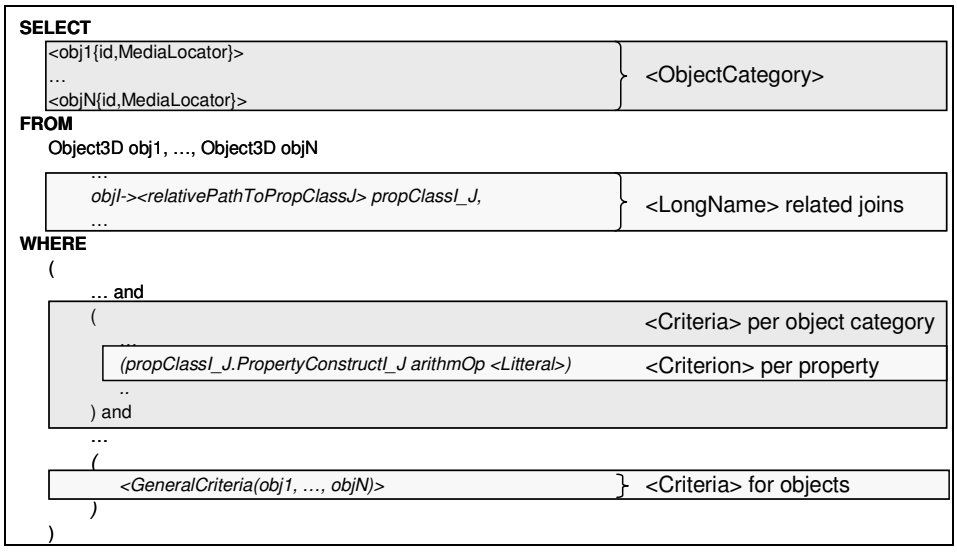


Figure 7.6 Patron de transformation de requêtes de réutilisation en requêtes 3DSEAM.

Dans la Figure 7.7, nous illustrons la traduction des trois requêtes de sélection décrites dans la Figure 7.4. Les deux premières comportent uniquement une catégorie d'objets. Par conséquent une seule variable (`obj1`) est introduite dans la clause `SELECT`. Du fait que les critères portent uniquement sur la sémantique dans les deux requêtes, nous introduisons un seul alias qui permet d'accéder aux propriétés relatives à l'objet (`obj1->Entity->Semantics sem1_1`). La même approche de traduction est adoptée dans la troisième requête, à la différence que dans ce cas, deux variables sont introduites (`obj1`, `obj2`), respectivement deux alias (`sem1_1`, `sem2_1`). Le critère portant sur les instances de chaque catégorie est reporté en fin de requête (`distanceM(obj1,obj2)<20`).

1. Sélection de tous les bâtiments et routes enregistrés dans l'entrepôt

```
SELECT obj1{id,MediaLocator}
    FROM Object3D obj1, obj1->Entity->Semantics sem1_1
    WHERE
        sem1_1!Catégorisation.catégorie=« bâtiment » or
        sem1_1!Catégorisation.catégorie=« route »
```

2. Sélection de tous les bâtiments dans un rayon de 1km autour du point de coordonnées géographiques 7.32E et 40.3N

```
SELECT obj1{id, MediaLocator}
    FROM Object3D obj1, obj1->Entity->Semantics sem1_1
    WHERE
        sem1_1!Catégorisation.catégorie=« bâtiment ») and
        inRange(sem1_1!Geospatil.gpsCoord,'7.32E 40.3N','1km')
```

3. Sélection des bâtiments et arbres se trouvant à une distance inférieure de 20 m, l'un de l'autre

```
SELECT obj1{id, MediaLocator}, obj2{id, MediaLocator}
    FROM Object3D obj1, Object3D obj2,
            obj1->Entity->Semantics sem1_1, obj2->Entity->Semantics sem2_1
    WHERE
            (sem1_1!Catégorisation.catégorie=« bâtiment ») and
            (sem2_1!Catégorisation.catégorie=« arbre »)
            distanceM(obj1,obj2) < 20
```

Figure 7.7 Construction de requêtes 3DSEAM à partir de couples propriété-valeur.

7.4.2. Le module de récupération de propriétés

Le module *Récupération de propriétés* est chargé de la publication des informations sémantiques relatives à un objet ou à un ensemble d'objets. La liste des noms de propriétés, dont les valeurs sont exigées par l'utilisateur, est fournie selon la notation introduite dans la Figure 7.5. Une requête par catégorie d'objets retrouvés est générée car chaque catégorie peut être associée à une description sémantique différente.

Liste des propriétés

[…, *ClassNameI{PropertyConstructs}*, …]

Requête 3DSEAM

```
SELECT
    obj.id,
    …, varI{<PropertyConstruct>, …}, …
    FROM
        Object3D obj,
        …,
        obj->PathToClassNameI varI,
        …
WHERE
    (obj.id in ($OBJ_IDs))
```

Figure 7.8 Patron d'extraction d'informations sémantiques.

Dans la Figure 7.8, nous présentons le patron d'extraction utilisé pour traduire les demandes d'information sémantique d'un client en requête 3DSEAM interprétable par la plate-forme 3DAF. Le squelette de la requête est composé des éléments soulignés dans la Figure 7.8. Le squelette implique l'association des identifiants des objets (`obj.id`) avec les propriétés sémantiques demandées. Dans la clause `WHERE`, un critère limite l'extraction des propriétés par rapport à la liste des identifiants des objets spécifiés par le paramètre `$OBJ_IDs`. Chaque propriété de la liste est introduite dans la requête à travers une variable (`varI`) associée à l'instance qui

contient la propriété (`ClassNameI`). Le chemin de navigation (`PathToClassNameI`) permettant d'atteindre cette instance, est construit automatiquement en considérant la structure fixe du modèle.

Liste des propriétés

Semantics{!Geospatial.gpsCoord,!Geospatial.altitude}
Appearance.dominantColor

Requête 3DSEAM

```
SELECT
    obj.id,
    sem{!Geospatial.gpsCoord,!Geospatial.altitude}, app.dominantColor
FROM
    Object3D obj,
    obj->Entity->Semantics sem, obj->Appearance app
WHERE
    and obj.id in ($OBJ_IDs)
```

Figure 7.9 Construction d'une requête SQL 3DSEAM pour l'extraction des couples propriété-valeur à partir d'une liste d'identifiants d'objets.

La Figure 7.9 montre un exemple concernant la récupération des coordonnées GPS (`Semantics!Geospatial.gpsCoord`), de l'altitude (`Semantics!Geospatial.altitude`) et de la couleur dominante (`Appearance.dominantColor`) de chaque objet dont l'identifiant a été extrait par le module de sélection d'objets et désigné par le paramètre `$OBJ_IDs`. Les *noms longs* des propriétés sont analysés afin d'extraire les classes correspondantes. La variable `s` est ainsi associée à la classe `Semantics`. La variable `a` est associée à la classe `Appearance`. La clause `SELECT` contient pour chaque variable l'ensemble des propriétés visées : `sem{!Geospatial. gpsCoord, !Geospatial.altitude}` et `app.dominantColor`. Du fait que les propriétés demandées sont associées aux différents concepts 3DSEAM, plusieurs chemins de navigation, ayant comme cible les instances de chaque classe, sont introduits : `obj->Entity->Semantics sem` et `obj->Appearance app` (voir la dernière ligne de la Figure 7.9).

7.5. Le module d'extraction de fragment 3D

Le module *Extraction de fragments 3D* est capable, à partir d'un repère de localisation, d'extraire le fragment du document correspondant et de le convertir dans le format X3D. Afin d'aboutir à cet objectif, le module repose sur des modules externes capables de prendre en charge les différents types d'encodage de fichiers et qui implémentent l'ensemble des types de repères de localisation proposé par 3DSEAM.

Chaque repère utilisé pour la localisation (voir section 5.2) comporte l'URL du document. À partir de cette information, et notamment de l'extension du document, le module d'extraction peut déduire le type d'encodage et le format de représentation. En conséquence, le module externe, capable de traiter ce type d'information, parcourt le document en recherchant les éléments géométriques sous l'emprise du repère de localisation.

Une deuxième étape de transformation vers un codage X3D est réalisée afin de disposer d'un moyen commun de représentation lorsque des fragments issus de plusieurs types de documents sont mis en commun. Tout fragment ainsi obtenu est ajouté au nœud `Group` auquel on associe en utilisant l'attribut `DEF` l'identifiant de l'objet dans le cadre de l'entrepôt d'annotation. Toute information concernant la position des objets dans la scène initiale est filtrée. En effet, à ce

niveau l'objet extrait correspond à une entité de présentation indépendante. La position de cet objet dans la nouvelle scène, est définie par le module d'assemblage de scène.

Avant de présenter la manière dont les scènes sont assemblées, nous montrons dans la section suivante la manière dont nous attachons l'information sémantique aux objets de la scène.

7.6. Le module d'attachement de la sémantique

Le module *Attachement de la sémantique* réalise l'annotation automatique des fragments documentaires représentant les objets 3D extraits par le module *Extraction de fragments 3D*. Les informations sémantiques à injecter dans chaque fragment sont celles fournies par le module *Récupération de propriétés* en rapport avec chaque objet et en concordance avec les souhaits formulés par les clients.

Le module s'appuie sur les éléments du standard X3D permettant de représenter des métadonnées au sein de scènes 3D, présentées dans la section 2.2 de l'état de l'art. L'ensemble des propriétés associées aux objets dans l'entrepôt de description est traduit en un ensemble d'éléments de métadonnées dont le choix (`MetadataSet`, `MetadataFloat`, `MetadataString`, `MetadataInteger`) est guidé par le type de valeur associé à chaque propriété. Les éléments `MetadataSet` sont utilisés pour représenter des propriétés dont les valeurs sont représentées par des données complexes. Les éléments `MetadataFloat`, `MetadataString`, `MetadataInteger` sont utilisés pour représenter des propriétés basiques dont les valeurs sont représentées par des types primaires. L'attribut `name` des nœuds `Metadata[*]` représente les noms longs de chaque propriété. L'attribut `reference` des noeuds `Metadata[*]` permet de définir le standard de métadonnées par rapport auquel sont définis le nom de la propriété et la signification de la valeur associée. Ainsi nous rangeons dans cet attribut, la valeur de l'attribut `ontoRes` présent dans la définition de la propriété (voir Figure 7.2 sur la description des profils supportés par le système). Le rôle de cette propriété est de faire le lien avec un concept présent dans une ontologie de domaine.

La transformation entre les deux moyens de représentation (format XML auto-descriptif de la plate-forme 3DAF et les éléments du standard X3D) se fait facilement grâce à une collection de feuilles de transformation XSLT [Clark, 1999], dont le détail est présenté à l'adresse `http://www-lsr.imag.fr/users/Marius.Bilasco/these/annexes/ATTACH_SEM.xsl`. Les éléments de description X3D ainsi construits sont attachés au nœud `Group` correspondant à la définition de l'objet.

Afin d'illustrer ces propos, nous proposons dans la Figure 7.10 un scénario d'annotation automatique d'un objet (`BâtimentNoir`) avec l'ensemble des propriétés (`Semantics!Geospatial.gpsCoords`, `Appearance.dominantColor`) extraites de l'entrepôt de connaissances 3DSEAM. Initialement, on dispose de la représentation X3D du fragment associé à l'objet `BâtimentNoir` et la liste des propriétés sémantiques. Suite à l'application des feuilles de style de transformation des propriétés en métadonnées X3D, nous obtenons le nœud `MetadataSet name="3dseam_annotations"` regroupant l'ensemble des propriétés. Nous décidons d'organiser les métadonnées sous ce nœud chapeautant les propriétés afin de ne pas interférer avec les annotations qui sont contenues ou qui peuvent être apportées manuellement au nœud `Group`. Dans cette illustration nous n'avons pas inclus les attributs `reference` afin de simplifier la complexité du rendu visuel et de faciliter la compréhension.

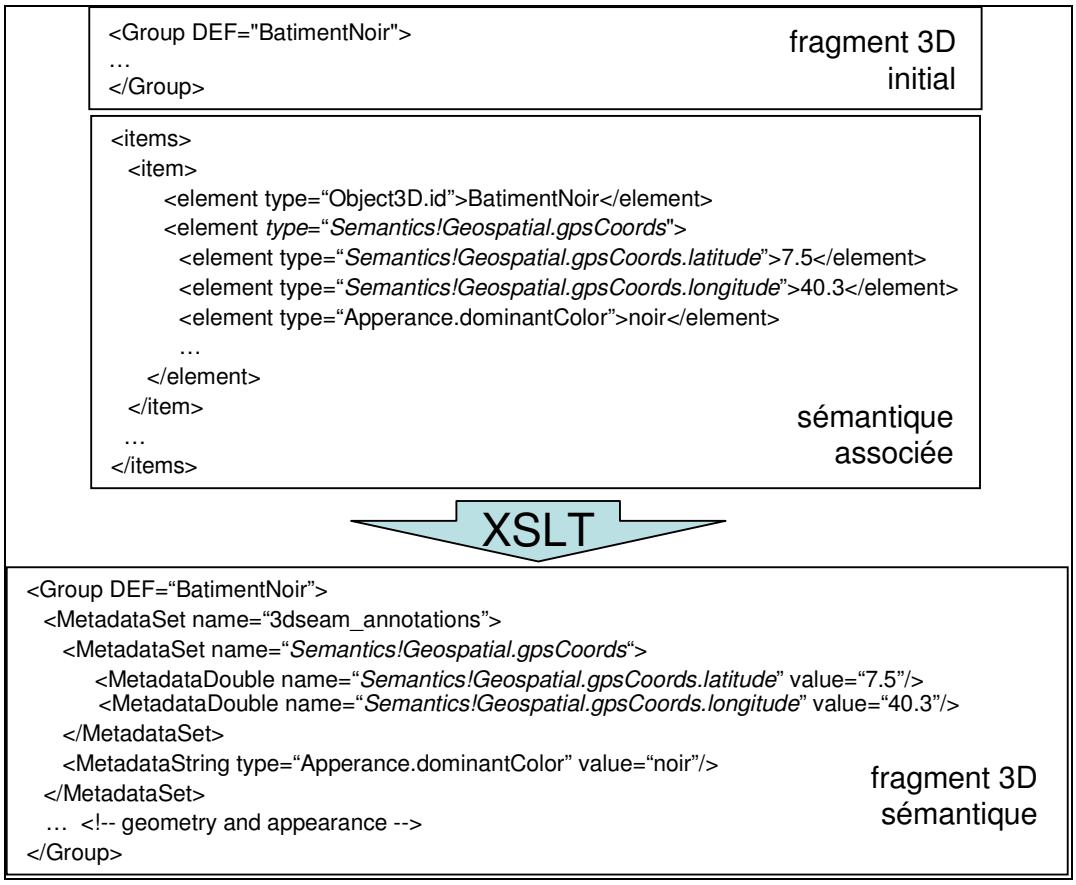


Figure 7.10 Injection des propriétés sémantiques dans les fragments 3D à l'aide de métadonnées.

7.7. Le module d'assemblage

Le module *Assemblage de scène* organise les fragments construits par le module *Attachement de la sémantique* au sein d'une scène qui est, par la suite, renvoyée au client de la plate-forme par le biais du module d'interface.

Indépendamment de l'approche d'assemblage choisie, il est fondamental de déterminer la position des objets au sein de la scène, en précisant les éventuels regroupements des objets au sein d'objets composites, et les positions de chacun au sein de la scène. Cette description peut être assimilée à un patron de visualisation qui dirige le processus de mise en scène des objets.

Nous décidons de formaliser ce type de description sous la forme d'un fichier XML qui comporte des informations sur l'organisation logique de la scène et qui précise la position, l'orientation et la mise à l'échelle de chaque objet. Nous choisissons d'adopter un schéma de description similaire à X3D.

Chaque objet résultant de la requête est introduit à la description de la pseudo scène sous la forme d'un nœud `Inline`. L'attribut `url` du nœud indique un URI dont le préfixe `_3dseam` annonce l'identifiant de l'objet correspondant. Nous avons fait ce choix pour permettre l'intégration des objets issus de la requête de réutilisation avec des éventuels objets 3D *décoratifs/statiques* qui enrichissent l'aspect visuel de la scène. Tout objet dont le préfixe de l'URL ne commence pas par `_3dseam` est considéré comme statique par le module d'assemblage.

La position, l'orientation et la mise à l'échelle d'un objet sont contrôlées par un élément `Transform` qui contient la définition de l'`Inline` introduisant l'objet. La position, la rotation et la

mise à l'échelle de chacun des sous objets d'un élément `Transform` sont définies par rapport aux systèmes de coordonnées attachés à l'objet parent.

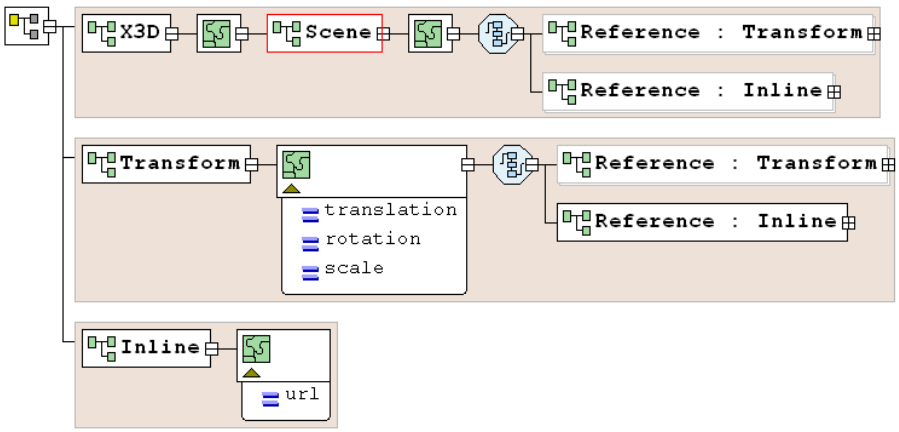


Figure 7.11 Définition XML Schema pour les éléments d'une pseudo scène.

Dans la Figure 7.11, nous présentons le *XML Schema* définissant les règles de description de la scène. Une scène (`Scene`) contient une séquence d'objets (`Inline`) entourée éventuellement d'éléments définissant leur position précise (`Transform`). Chaque élément `Transform` comporte trois attributs (`position`, `rotation`, `scale`) et éventuellement, une séquence de sous-élémentss `Inline` ou `Transform`. Le choix de considérer les nœuds `Transform` dans ce pseudo langage de description, a été fait pour laisser au moteur externe de disposition spatiale le soin de définir un graphe de scène de manière à ce que le rendu soit optimisé. En effet, à chaque scène on peut associer une représentation ayant un graphe de scène à un niveau. Tous les objets sont définis dans le repère global de la scène. Ceci représente une augmentation de calcul de visibilité au moment du rendu.

L'intérêt d'une telle approche de description intermédiaire, est que cela permet, au module externe de calcul de position des différents éléments, de personnaliser la disposition des objets et de l'environnement qui entoure les éléments, en utilisant directement les constructions disponibles dans le langage X3D.

```
<X3D profile="Immersive" version="3.0">
<Scene>
  <Background skyColor="1 1 1"/>
  <Viewpoint position="0 0 50" description="face"/>
  <Transform translation="20 0 20">
    <Transform scale="20 0 20">          <Inline url="parcelleUnitaire.x3d"/>      </Transform>
    <Transform translation="0   1.5 0">  <Inline url="_3dseam://rdcGris">          </Transform>
    <Transform translation="0   4.6 0">  <Inline url="_3dseam://1erGris">          </Transform>
    <Transform translation="0   7.7 0">  <Inline url="_3dseam://2eGris">          </Transform>
    <Transform translation="0 10.8 0">   <Inline url="_3dseam://3eGris">          </Transform>
    </Transform>
  <Transform translation="-30 0 20">
    <Transform scale="20 0 20">          <Inline url="parcelleUnitaire.x3d"/>      </Transform>
    <Transform translation="0   7 0">    <Inline url="_3dseam://BatimentNoir">     </Transform>
  </Transform>
</Scene>
</X3D>
```

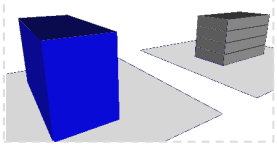


Figure 7.12 Pseudo description d'une scène 3D.

Dans la Figure 7.12, nous présentons une *pseudo* description de scène correspondant à une scène 3D incluant deux bâtiments extraits de l'entrepôt. Le terme *pseudo* indique le fait que

la scène n'est pas envoyée en l'état à l'utilisateur car elle fait référence aux notions du modèle 3DSEAM (les `Inline` en italique). La pseudo scène indique que le premier bâtiment est représenté par ses étages (`rdcGris`, `1erGris`, `2eGris`, `3eGris`) et le second est représenté en tant qu'objet atomique (`BatimentNoir`). Deux éléments décoratifs sont introduits par l'URL relatif `parcelleUnitaire.x3d`. Chaque objet est introduit par une construction `Transform` – `Inline`. D'autres éléments, propres au langage X3D, sont contenus dans la pseudo description de la scène (`Background`, `Viewpoint`). Les éléments en italique sont ceux qui sont remplacés par la représentation des objets extraits par le module *Extraction de fragments 3D* et annotés ultérieurement par le module *Attachement de la sémantique*.

En utilisant le langage XSLT, nous transformons la *pseudo scène* en une scène X3D où les nœuds `Inline`, qui font référence aux URL internes au modèle de la requête, sont remplacés par les nœuds `Group` encodant la géométrie, l'apparence et la sémantique.

Dans la Figure 7.13, nous présentons une partie de document X3D obtenu suite aux transformations de la description de la pseudo scène présentée dans la Figure 7.12. Nous concentrons notre attention sur l'objet `BatimentNoir`, dont la représentation effective remplace le nœud `Inline url="_3dseam://BatimentNoir"` dans la pseudo scène (voir Figure 7.12). L'identifiant indiqué après le préfixe `_3dseam://` permet au module *Assemblage de scène* de chercher parmi les fragments extraits et annotés, celui qui correspond au nœud `Inline` en cours de traitement et de le remplacer en conséquence.

La dernière tâche à remplir par le module d'assemblage est l'encodage de la scène suivant les souhaits indiqués par l'utilisateur. Cette tâche est déléguée aux modules externes capables de traduire une scène X3D dans un format spécifique.

```
<X3D profile="Immersive" version="3.0">
<Scene>
  <Background skyColor="1 1 1"/>
  <Viewpoint position="0 0 50" description="face"/>
  <Transform translation="20 0 20">
    …
  </Transform>
  <Transform translation="-30 0 20">
    <Transform scale="20 0 20">        <Inline url="parcelleUnitaire.x3d"/>        </Transform>
    <Transform translation="0   7 0">
      <Group DEF="BatimentNoir">
        <MetadataSet name="3dseam_annotations">
          <MetadataSet name="Semantics!Geospatial.gpsCoords">
            <MetadataDouble name="Semantics!Geospatial.gpsCoords.latitude" value="7.5"/>
            <MetadataDouble name="Semantics!Geospatial.gpsCoords.longitude" value="40.3"/>
          </MetadataSet>
          <MetadataString type="Apperance.dominantColor" value="black"/>
        </MetadataSet>
        <Transform scale="12 7 5"/>
          <Shape><Box/><Appearance><Material diffuseColor="0 0 0"/></Appearance></Shape
        </Transform>
    </Transform>
  </Transform>
</Scene>
</X3D>
```

Sémantique

Géometrie et Apparence

Figure 7.13 Fragment de scène sémantique construite par le module Assemblage de scène.

7.8. Le module de contrôle de la réutilisation

Comme nous l'avons présenté tout au long de ce chapitre, le processus de réutilisation suppose l'enchaînement de plusieurs étapes afin d'aboutir au résultat final : une nouvelle scène annotée convenablement selon la demande des clients.

Pour donner de la flexibilité au système, nous décidons de construire un module dont la première responsabilité est d'orchestrer les actions indépendantes de chacun des modules fonctionnels de l'architecture : *Récupération d'objets*, *Récupération de propriétés, Extraction de fragments 3D*, *Attachement de la sémantique* et *Assemblage de scène*.

Afin d'illustrer le rôle du contrôleur au sein de la plate-forme, nous présentons dans la Figure 7.14 les flux de données entre ses différentes entités. Dans un premier temps, le module de contrôle demande (1) au module *Récupération d'objets* de récupérer les repères dans la scène 3D qui correspondent aux fragments indiqués et de les déposer (2) dans l'entrepôt : *ID et repères d'objets/Catégorie*. Ces repères sont ensuite lus (3) par le module de contrôle et transférés (4) aux modules d'extraction de fragments et de recherche de propriétés. Ces deux actions peuvent se dérouler en parallèle (5). Le module *Récupération de propriétés* alimente l'entrepôt des *Propriétés sémantiques* avec les propriétés sémantiques demandées par le client pour chaque catégorie d'objets. Le module *Extraction de fragments 3D* alimente l'entrepôt de *Fragments X3D* local à la plate-forme. Une fois les deux opérations réalisées, le module de contrôle demande l'implantation des métadonnées dans les fragments extraits (6). Le module *Attachement de la sémantique* récupère (7) la liste des propriétés et le fragment 3D correspondant avant d'appliquer la transformation et la mise à jour dans l'entrepôt de fragments 3D (8). Ensuite, le module de contrôle demande l'assemblage des fragments (9) pour lesquels l'ensemble des propriétés sémantiques a été intégré. Ce dernier module *Assemblage de scène* extrait (10) les caractéristiques susceptibles d'intéresser les modules d'assemblage externes, et une fois la disposition spatiale calculée par ceux-là, il génère la nouvelle scène qu'il transfère au module d'interface.

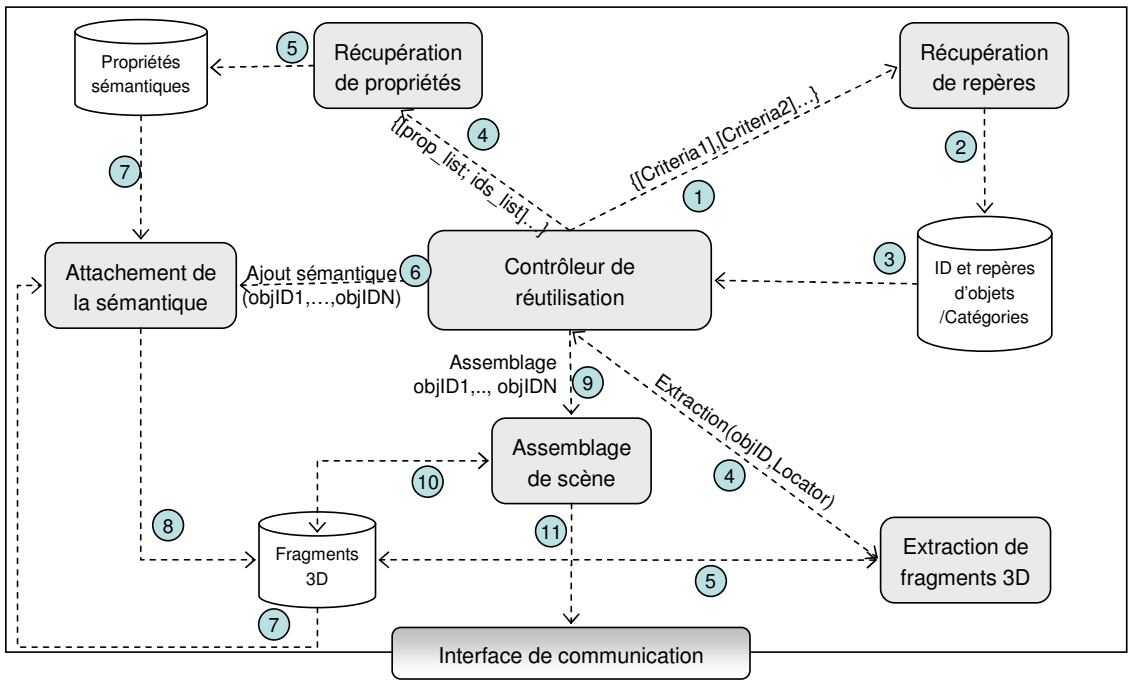


Figure 7.14 Flux de données au sein de la plate-forme de réutilisation.

Dans la section suivante, nous présentons une étude de cas concernant la prise en compte des opérateurs externes issus du domaine de la géomatique. Nous présentons également un profil géospatial construit à partir des métadonnées géospatiales supportées par le standard X3D.

7.9. Une extension géospatiale de 3DSDL

Le but de cette section est d'illustrer les possibilités d'extension de notre modèle et de la plate-forme de gestion de l'information qui lui est associée. À cette fin, nous proposons une extension visant la gestion et l'analyse de contenus géoréférencés [Bilasco *et al.*, 2007a]. Nous définissons un contenu géoréférencé en tant que collections de fragment multimédia disposant chacun directement, ou à travers l'entité qu'il représente, d'un profil sémantique géospatial.

Cette extension vise les scènes X3D car elles supportent l'association d'information sémantique ou géospatiale avec les différents éléments de la scène grâce aux métadonnées (voir la section 2.2 dédiée à la sémantique dans les scènes X3D). La sémantique et l'information géospatiale sont stockées dans les noeuds de la famille de métadonnées (`Metadata[Set|Int|Float|String]`) qui sont associés aux primitives géométriques (`Cube`, `Box`, `IndexedFaceSet`, etc.) représentant le contenu de la scène. L'information géospatiale peut être incluse directement dans le document X3D à travers les primitives géolocalisées proposées dans la composante *Geospatial component*[38] du standard X3D. Cependant, ce cas de figure se retrouve plus souvent dans le cadre des scènes qui sont géolocalisées dès leur création. L'utilisation des primitives géolocalisées fige la position des éléments par rapport à un repère global. Ainsi, lorsque l'on veut utiliser un fragment géolocalisé dans une autre scène, le fragment garde sa position initiale. Cependant, si un concepteur veut rassembler dans une même région (un musée virtuel) les copies d'objets 3D (la Tour de Londres, la Tour Eiffel, les Pyramides) éparpillés sur tout le globe, il est plus judicieux de les avoir stockées à l'aide de primitives 3D simples et de rajouter des métadonnées spécifiques au profil géospatial (`GeoMetadata`).

Dans cette proposition nous nous orientons vers ce type de solution, où l'utilisateur veut réutiliser des objets initialement georéférencés sans forcément vouloir les placer au même endroit dans la nouvelle scène. L'information géospatiale des objets initiaux est portée par les répliques en tant que métadonnées.

La suite de cette section est consacrée à la présentation du profil géospatial qui regroupe l'ensemble des propriétés géospatiales que l'on rencontre au niveau de la composante *Geospatial* du standard X3D. Une fois les bases du profil géospatial posées, nous présentons l'ensemble des opérateurs de sélection géographique que nous retenons pour cette version de la bibliothèque. Les opérateurs sont construits sur le modèle de ceux proposés par l'OpenGIS Consortium[39] et concernent la distance et la position relative de deux objets géoreférencés. La troisième partie de cette section illustre comment ces nouveaux opérateurs s'intègrent à ceux supportés par défaut dans la bibliothèque 3DSDL. Finalement, nous détaillons un scénario d'exécution et le résultat attendu.

7.9.1. Un profil géospatial

Afin de répondre aux besoins des applications s'intéressant à la nature géospatiale des données 3D, nous proposons un profil sémantique, nommé *X3DGeospatial*. Ce profil regroupe le

[38] http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/Part01/components/geodata.html
[39] http://www.opengeospatial.org/

minimum d'information (voir Tableau 6) en rapport avec un objet 3D dont une application doit disposer afin de prétendre à la mise en œuvre d'opérations d'analyse géospatiale.

Nom	Description
coordinateSystem	type de système de coordonnées utilisé pour localiser l'objet (GD, GC, UTM)
ellipsoid	nom de l'ellipsoid géodésique considéré
extent	l'étendue spatiale des objets définie par rapport au système de coordonnées
resolution	la résolution en unités/m
originator	la source des données
Date	un instant ou intervalle définissant la validité de la donnée
latitude	la latitude de l'objet par rapport au système de coordonnées choisi
longitude	la longitude de l'objet par rapport au système de coordonnées choisi
elevation	l'altitude de l'objet par rapport au géoïde utilisé

Tableau 6 Liste des propriétés du profil X3DGeospatial.

Avant de passer en revue les opérateurs géospatiaux, nous illustrons l'instanciation d'un profil géospatial par rapport aux objets contenus dans une scène représentant le site des *Grandes Pyramides* (voir Figure 7.15). La scène contient une modélisation basique des trois pyramides : *Kheops*, *Chepren* et *Mykorinus*, ainsi que du *Sphynx*.

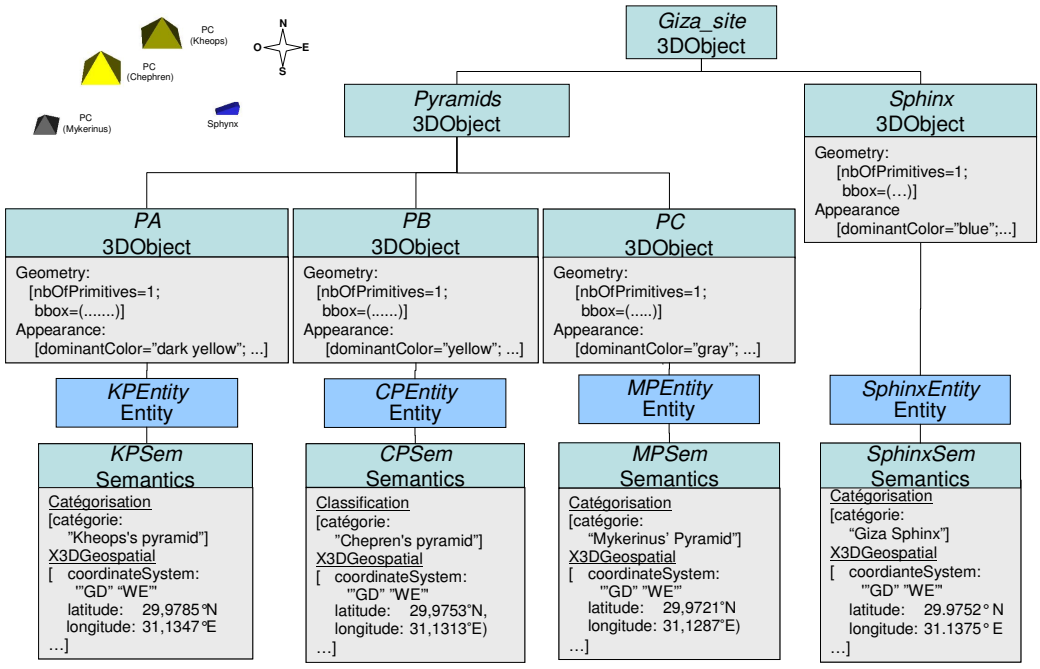


Figure 7.15 Scène aux éléments géométriques géolocalisés représentant le site de Giza.

Chaque objet est associé à une entité homonyme dont la sémantique regroupe des propriétés issues du profil géospatial. Ainsi, par exemple, l'objet PA est associé à l'entité KPEntity qui est caractérisée par un profil *X3DGeospatial*. Les informations de ce profil indiquent que l'entité KPEntity est localisée à 29,9785°N (latitude) et 31,1347°E (longitude) par rapport à un système de coordonnées géodésique (GD). Nous utilisons les degrés décimaux pour indiquer la latitude et la longitude afin de se conformer à la spécification de la composante

géospatiale du standard X3D qui préconise l'utilisation de nombres réels pour représenter chaque composante d'une coordonnée géographique. Des outils logiciels tels que l'API *Chaeron GPS Library*[40] *ou GeoTools*[41] permettent le passage entre les différentes modalités de représentation (degrés-minutes DM, degrés-minute-secondes DMS, degrés décimaux DD) de la latitude et la longitude.

7.9.2. Opérateurs géospatiaux

Dans cette section nous présentons les opérateurs géospatiaux qui permettent de formuler des requêtes spatiales complexes. Ils sont implémentés par le module d'*Analyse Géospatiale* qui se propose comme un composant externe à notre architecture (la partie droite de la Figure 7.16).

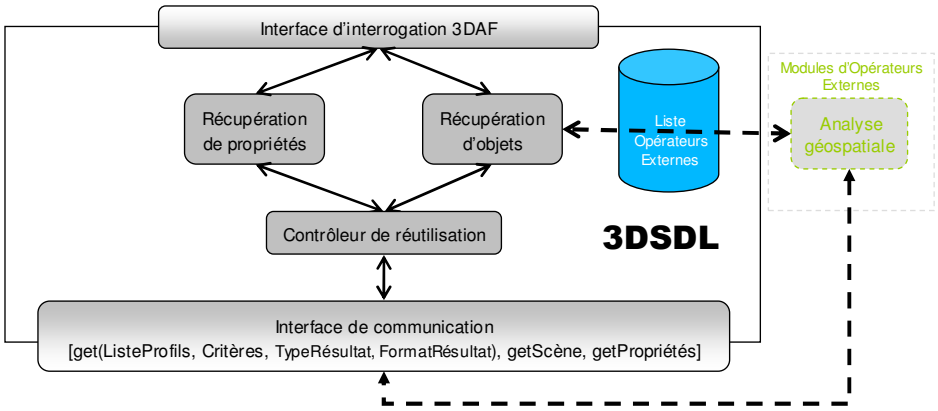


Figure 7.16 Lien avec le module externe d'analyse géospatiale.

Les modules externes enregistrent auprès de la plate-forme 3DSDL l'ensemble des opérateurs (identifiés par leur nom) dont ils disposent. Ces liens permettent à la plate-forme de savoir quels modules traitent quels opérateurs spécifiques. Plus particulièrement, le module *Analyse géospatiale* est mis en relation avec le module *Récupération d'objets*. L'analyse géospatiale requiert des informations relatives à la nature géographique des objets 3D. Ainsi, le module externe se connecte à 3DSDL afin d'accéder aux informations nécessaires à l'évaluation des requêtes impliquant divers opérateurs géospatiaux.

Le module propose trois opérateurs quantitatifs : *equals, distance, inRange*. L'opérateur *equals* évalue si deux entités localisées de manière indépendante au sein du système de références spatiales, sont situées à la même coordonnée géographique absolue. Cet opérateur est similaire à l'opérateur *equals* de l'OpenGIS qui teste l'égalité spatiale dans un système de référence géométrique unique. L'opérateur *distance* calcule la distance réelle entre deux entités géolocalisées indépendamment des systèmes de références. L'opérateur *inRange* évalue si un objet se trouve à l'intérieur d'un rayon indiqué par rapport à un point de l'espace.

Deux opérateurs qualitatifs : *contains* et *inside* sont également considérés. Leur évaluation demande l'analyse des propriétés géométriques ou des relations topologiques des données 3D, en plus du profil sémantique géospatial. Suivant la nature des informations disponibles, l'opérateur *contains* est calculé :

a) soit en vérifiant l'inclusion de la surface géospatiale des deux régions associées aux objets opérandes,

b) soit en vérifiant l'existence d'une relation topologique `contains` entre les objets,

[40] http://www.chaeron.com/gps.html#JavaLibrary
[41] http://geotools.codehaus.org/2.3.1

c) soit en vérifiant l'inclusion partielle de deux objets dans le système cartésien de coordonnées non nécessairement géolocalisées.

Pour illustrer ces situations, nous supposons qu'à l'intérieur de la scène présentée dans la Figure 7.15, seul l'objet modélisant le plateau (`Giza_site`) est géolocalisé. Une requête qui demande une pyramide située dans la région correspondant au plateau `Giza_site` peut être résolue uniquement si on combine l'information topologique avec le profil géospatial.

Dans un premier temps, il est nécessaire d'identifier les objets qui sont contenus dans la région désignée. Ensuite pour chaque objet, ici l'objet Giza_site, on procède à la recherche d'objets inclus d'un point de vue topologique dans l'objet principal. Le processus de recherche peut continuer afin d'explorer les objets sous-jacents. Le moteur de résolution de la requête décide du nombre de pas avant que le processus d'exploration soit terminé.

D'une manière semblable, nous gérons l'instanciation d'une famille d'opérateurs évaluant les positions relatives des deux objets par rapport aux axes NS-EO du globe terrestre. Les opérateurs en question sont : *geo_rel_north*, *geo_rel_south*, *geo_rel_east*, *geo_rel_west*, et ils évaluent si la première opérande est au nord, respectivement au sud, à l'est ou à l'ouest de la deuxième opérande. Si les géocoordonnées sont disponibles, un calcul direct peut produire le résultat souhaité. Dans le cas contraire, l'algorithme d'évaluation explore d'autres propriétés des objets/entités concernés (par exemple, on cherche des objets géolocalisés qui contiennent spatialement les objets concernés et on reporte le calcul sur les objets géolocalisés).

7.9.3. Plan d'exécution pour les requêtes géospatiales

Dans la Figure 7.17, nous présentons le plan d'exécution d'une requête géospatiale. Nous supposons que toutes les coordonnées des objets sont connues. Nous nous focalisons sur la scène correspondant à la description sémantique faite dans la Figure 7.17. La requête vise à identifier les couples d'objets associés respectivement à la catégorie sémantique *pyramid* ou *sphinx*, et dont la disposition spatiale satisfait la contrainte suivante : la pyramide recherchée est située au nord du sphinx.

I. Requête initiale transférée au module *Récupération d'objets*

geo_rel_north(
["Semantics!Catégorisation.catégorie", "Pyramid"],
["Semantics!Catégorisation.catégorie", "Sphinx"])

II. Décomposition en requêtes ne faisant pas appel aux opérateurs externes et récupération des couples d'objets opérandes

(A) ["Semantics!Catégorisation.catégorie", "Pyramid"]
(B) ["Semantics!Catégorisation.catégorie", "Sphinx"]

III. Transfert des couples d'objets (A x B) au module d'*Analyse géospatiale*

1. récupération de propriétés géospatiales des objets candidat
List of 3D object identifiers
$A_1, \ldots, A_N, B_1, \ldots, B_M,$
List of properties:
Semantics!Geospatial.coordinateSystem
Semantics!Geospatial.latitude, Semantics!Geospatial.latitude

2. vérification de la relation *geo_rel_north* pour chaque couple (A x B)

IV. Transmission au module de *Récupération d'objets* les couples qui valident le critère

Figure 7.17 Requête géospatiale et le plan d'exécution associé.

Lorsque le module *de Récupération d'objets* de 3DSDL reçoit la requête (I), il analyse tout d'abord le type d'opérateurs présents dans la requête. Quand un opérateur externe est

identifié, la requête est décomposée en plusieurs sous requêtes qui visent la récupération des valeurs des opérandes de l'opérateur en question (II). Les résultats partiaux de ces requêtes sont transmis au module concerné (ici le module d'*Analyse Géospatiale*) (III). Le module d'analyse géospatiale a besoin des informations concernant les coordonnées de chaque entité et le système de référence par rapport auquel les coordonnées sont exprimées. Le module *Analyse Géospatiale* extrait ces informations pour chaque item de la collection des résultats partiels en communiquant avec le module de *Récupération de propriétés* à travers l'*interface de communication* de 3DSDL (III.1). Après cela, tous les tuples obtenus en combinant les items de la première et de la seconde collection de résultats sont validés par rapport à l'opérateur *geo_rel_north* (III.2). Dans notre cas, uniquement le couple (PA, Sphinx) est retenu (en IV) car l'objet désignant la pyramide de Kheops (associée à l'entité PA) est le seul localisé au nord de l'objet "Sphinx" (comme on peut l'observer à partir des profils géospatiaux présentés dans la Figure 7.15).

Le plan d'exécution que nous avons décrit ci-dessus est relativement simple car toutes les informations nécessaires sont disponibles. Cependant, dès qu'une des entités pyramide n'est pas annotée avec les informations géospatiales nécessaires, des algorithmes plus complexes doivent être mis en place afin d'obtenir une approximation des coordonnées manquantes.

7.9.4. Assemblage de scène contenant des informations géospatiales

L'assemblage d'une scène annotée avec des informations géolocalisées est le même que dans le cadre général. Le processus de sélection d'objets et de la sémantique à leur associer dans la scène X3D générée, est transparent pour l'assembleur, car le module assemblant les scènes est indépendant.

```
 1:<X3D profile="Core" ...>
 2: <Scene>
 3:  <Transform>
 4:   <Transform DEF="PA" translation="6 0 -10.3" scale="2.3 1.46 2.3">
 5:    <MetadataSet name="3dseam_annotations">
 6:     <MetadataString name="Semantics!Catégorisation.nom" value="Kheops' Pyramid"/>
 7:     <MetadataString name="Semantics!X3DGeospatial.latitude" value="29.9785°N 31.13472°E"/>
 8:     <MetadataString name="Semantics!X3DGeospatial.longitude" value="31.13472°E"/>
 9:     <MetadataString name="Semantics!X3DGeospatial.coordinateSystem" value='"GD" "WE"'/>
10:    </MetadataSet>
11:    <Shape>
12:     <IndexedFaceSet>...</IndexedFaceSet>
13:     <Appearance>
14:      <Material diffuseColor="0.6 0.6 0"/>
15:     </Appearance>
16:    </Shape>
17:   </Transform>
18:   <Transform DEF="Sphinx" translation=" 9.2 0 -3 ">
19:    <MetadataSet name="3dseam_annotations">
20:     <MetadataString name="Semantics!Catégorisation.nom" value="Kheops' Pyramid"/>
21:     <MetadataString name="Semantics!X3DGeospatial.latitude" value="29.9752°N"/>
22:     <MetadataString name="Semantics!X3DGeospatial.longitude" value="31.1374°E"/>
23:     <MetadataString name="Semantics!X3DGeospatial.coordinateSystem" value='"GD" "WE"'/>
24:    </MetadataSet>
25:    <Shape>
26:     <IndexedFaceSet>...</IndexedFaceSet>
27:     <Appearance>
28:      <Material diffuseColor="0 0 1"/>
29:     </Appearance>
30:    </Shape>
31:   </Transform>
32: </Scene>
33:</X3D>
```

Figure 7.18 Scène comportant des annotations géospatiales générée par 3DSDL.

Le résultat du processus d'assemblage correspondant à la requête géospatiale de la Figure 7.17 est présenté dans la Figure 7.18. Pour chaque noeud qui correspond aux résultats de la requête (lignes 5 et 19), le module d'assemblage inclut dans le document un nœud (`MetadataSet`) qui regroupe l'ensemble des informations sémantiques et géospatiales indiquées par le concepteur lors de la formulation de la requête (lignes 6-9, 20-23). Chaque propriété sémantique ou géospatiale est décrite par le type de la propriété au sein du modèle 3DSEAM ainsi que par sa valeur.

Nous avons utilisé les métadonées disponibles en X3D afin de construire notre profil *X3DGeospatial*. Cependant, nous attirons l'attention sur le fait que, lorsque l'on génère des scènes annotées avec des informations géospatiales, ces informations ne sont pas incluses en tant que nœud géospatial, mais comme faisant partie d'un profil sémantique d'annotations. Ainsi, les définitions géométriques et l'information géospatiale que nous leur attachons n'interfèrent pas les unes avec les autres.

7.10. Synthèse

Dans ce chapitre, nous avons, illustré au moyen d'une plate-forme de réutilisation, les manières dont les connaissances peuvent être exploitées afin de simplifier la mise en place des procédés de réutilisation sémantique.

Par rapport aux travaux liés à la réutilisation que nous avons présentés dans l'état de l'art, **la plate-forme 3DSDL est extensible du point de vue des outils de sélection de contenu** (possibilité d'enregistrer des opérateurs externes, tel que présenté dans la dernière section), **des types d'encodage de données 3D** (module d'extraction de contenu extensible) et **des méthodes d'assemblage de résultats** (module d'assemblage de scène extensible). Ainsi, à la différence des solutions sur mesure présentées dans l'état de l'art, 3DSDL s'adapte facilement à un large éventail de domaines d'application.

Par ailleurs, **la plate-forme s'appuie sur une réutilisation à base de catégories d'objets construites sur critères sémantiques**. L'usage des **patrons de conception de scènes** confère aux utilisateurs de la plate-forme la possibilité de personnaliser la manière dont les contenus réutilisés sont assemblés. Les patrons (ou *pseudo* descriptions de scènes) sont décrits en tant que documents X3D dont le contenu extrait à partir de 3DSEAM est introduit en utilisant les éléments de base de X3D. Ceci donne la possibilité aux concepteurs d'éditer le patron dans leur logiciel 3D favori afin de le personnaliser à leur guise. Cette approche document confère des possibilités d'évolution plus importante que celles proposées en [Bosca *et al.*, 2007] ou [Sommaruga *et al.*, 2007], où la génération des nouvelles scènes 3D est contrôlée de manière exclusive par l'application.

Dans le chapitre suivant, nous présentons une solution qui s'appuie sur la plate-forme 3DAF et réutilise quelques modules de la plate-forme 3DSDL afin de mettre en œuvre l'adaptation de scènes annotées sémantiquement.

8. Vers une adaptation différenciée des données de scènes 3D

Par rapport à l'adaptation de documents multimédia 2D où le nom et les caractéristiques (largeur, hauteur, taille en mémoire, etc.) de chaque ressource média sont partiellement connus, dans un document 3D nous devons traiter des unités d'information géométrique qui sont généralement éparpillées dans l'ensemble du document sans noms, ni frontières précises. À notre connaissance, il n'existe aucune approche qui détermine automatiquement les objets représentant des entités du monde réel et leurs frontières dans un document 3D. Pour cette raison, la plupart des approches d'adaptations existantes, et notamment celles s'intéressant à diminuer la complexité géométrique d'une scène [Hoppe, 1996][Cohen *et al.*, 1997][To *et al.*, 1999], considèrent la scène 3D dans son ensemble et applique des transformations uniformes (par exemple des dégradations géométriques ou de texture) à la scène entière. Par conséquent, le même procédé d'adaptation est appliqué à un arbre, à un bâtiment ou à une surface 3D (par exemple un Modèle Numérique de Terrain) contenus dans la même scène 3D. Ceci présente le risque d'aboutir à une scène incohérente, où de grandes quantités d'informations pertinentes pour l'utilisateur sont perdues à cause de la dégradation uniforme.

Nous défendons l'idée que **le processus d'adaptation doit combiner différentes techniques comme la dégradation de la géométrique ou de la texture, le filtrage d'objets, la substitution, l'accentuation de propriétés de certains objets (modification du niveau de transparence, etc.) afin de produire des résultats adéquats et sur mesure**.

La capacité de combiner plusieurs techniques d'adaptation pour la même scène est cruciale parce que tous les objets ne peuvent être adaptés d'une manière appropriée suivant une technique unique. Par exemple, une technique d'adaptation conçue pour la simplification du MNT dégrade excessivement un objet représentant un bâtiment inclus dans la même scène que le MNT.

Des méthodes qui privilégient une adaptation sélective, suivant les besoins en termes d'information de l'utilisateur, doivent être mises en place. Notre proposition est un pas vers cela.

Nous proposons une plate-forme d'adaptation offrant aux concepteurs des méthodes logicielles et des formalismes leur permettant d'**assembler des scènes ou de transformer des scènes en identifiant des classes d'objets et en leur associant des méthodes spécifiques d'adaptation.**

Nous présentons également la manière dont, au sein de notre approche, on peut relier les éléments du contexte de diffusion d'une scène au choix à faire dans le processus d'adaptation afin de satisfaire les besoins en information des utilisateurs ainsi que les contraintes de diffusion.

La localisation puis la caractérisation des objets d'une scène 3D, que nous avons présentés dans les chapitres précédents, représentent deux étapes nécessaires pour la mise en place d'adaptations adéquates.

Ce type d'information de localisation donne la possibilité d'indiquer au moteur d'adaptation comment adapter une partie spécifique d'une scène 3D en associant sa localisation à une transformation spécifique. En utilisant cette approche directe, où les paramètres d'adaptation doivent être définis individuellement pour chaque objet ou région, exige des efforts conséquents. Cependant, à l'intérieur de la même scène 3D, **chaque catégorie « sémantique » d'objets 3D est généralement destinée à une adaptation homogène de ses représentants** (par exemple tous les arbres sont dégradés uniformément). Nous voulons ainsi œuvrer pour la création d'un moyen permettant de décrire les adaptations à appliquer à certaines classes d'objets, classes définies sur des critères sémantiques.

Dans ce but, nous pensons que des informations additionnelles, qui complètent la représentation brute des objets, sont nécessaires afin de pouvoir désigner convenablement des catégories d'objets construites sur des critères sémantiques au sein d'une scène 3D. Comme, nous l'avons indiqué dans le chapitre précédent, cette information peut être soit extraite directement à partir des noeuds de métadonnées liés aux objets dans les scènes 3D, soit ajoutée ultérieurement au moyen d'annotations sémantiques. Dans les deux cas, dès que l'information est présente dans le modèle 3DSEAM, nous pouvons envisager de **formuler des adaptations exploitant les propriétés sémantiques** des objets concernés.

Nous distinguons deux notions pour l'adaptation : **des règles d'adaptation** et **des stratégies d'adaptation**. Une règle d'adaptation est construite en deux parties : la première définit le critère de choix des objets sous l'emprise de la règle et la seconde correspond à la technique d'adaptation qui s'applique aux objets qui répondent aux critères de choix. Une stratégie d'adaptation correspond à un ensemble de règles d'adaptation et un ordre d'exécution précis.

Dans ce qui suit, nous insistons premièrement sur la définition des règles d'adaptation, puis sur la définition des stratégies d'adaptation. Ensuite, nous présentons une architecture fonctionnelle, nommée Adapt3D [Bilasco *et al.*, 2007c], supportant l'interprétation des règles et des stratégies d'adaptation. Avant d'en faire une synthèse, nous présentons dans ce chapitre un exemple d'adaptation à base de règles.

8.1. Définition des règles d'adaptation

Comme mentionné plus haut, **une règle d'adaptation se compose d'un critère de sélection et d'une technique d'adaptation**. Nous précisons que les règles d'adaptation ne sont pas considérées comme des constructions impératives. L'association d'une règle à une scène n'implique pas directement que tous les objets de cette scène soient adaptés selon cette règle. La règle participe plutôt à l'élaboration d'une liste de méthodes d'adaptation qui sont cohérentes avec la nature sémantique de l'objet.

Le critère de sélection correspond à une clause WHERE dans une requête 3DSEAM SQL. Les autres parties de la requête sont implicites, car ce que l'on cherche, ce sont des fragments multimédia, ce qui correspond à la clause SELECT. La clause FROM indique la scène sur laquelle porte le critère de sélection. Ainsi, il est relativement facile de construire une requête 3DSEAM SQL afin d'extraire les objets soumis à la règle. Le critère de choix est formulé en utilisant une expression logique contenant un ensemble de couples de propriété-valeur (nom de propriété, valeur de propriété). La désignation du nom de la propriété suit les conventions de nom présenté dans le chapitre précédent.

La technique d'adaptation est dénotée par son identifiant et un ensemble de paramètres régissant le processus d'adaptation. Ici, nous discutons principalement de la définition des règles d'adaptation : nous n'avons pas l'intention d'offrir ou établir un ensemble exhaustif de techniques d'adaptation. Chaque plate-forme d'adaptation construite au-dessus du modèle de règles que nous proposons, doit être libre de définir un ensemble de techniques d'adaptation spécifique. En définissant une règle d'adaptation, le concepteur de la règle doit tenir compte des types d'objets sous l'emprise du critère de sélection formulé afin de définir la technique d'adaptation la plus appropriée.

Un autre aspect important est la prise en compte des informations extraites du contexte de diffusion afin d'aligner les paramètres du processus d'adaptation (tel que le niveau de détail ou autre) avec les informations contextuelles ou les préférences utilisateur.

À partir de ces constats, nous proposons une notation pour définir les règles d'adaptation (`AdaptationRule`). Dans la Figure 8.1, nous présentons la première clause de la grammaire des règles d'adaptation. Une règle est identifiée par son nom (`RuleName`), la désignation des objets potentiellement concernés par cette règle d'adaptation (`AdaptationScope`), la désignation de la méthode d'adaptation (`AdaptationMethod`) applicable aux objets d'une scène sous l'emprise de la règle, et les paramètres de la méthode d'adaptation (`AdaptationParams`).

<AdaptationRule> :=
"Rule" <RuleName> ":=" "(" <AdaptationScope>, <AdaptationMethod>, <AdaptationParams> ")"

Figure 8.1 Notation BNF [Backus, 1959] pour une règle d'adaptation.

Dans les sous-sections suivantes nous analysons et précisons les modalités de description de chaque partie d'une règle d'adaptation. Nous commençons par discuter des modalités de construction de l'emprise d'une règle sur les éléments d'une scène.

8.1.1. L'étendue et l'applicabilité d'une règle

Une catégorie d'objets est définie par une série de critères qui portent sur les dimensions sémantique, géométrique, d'apparence, topologique et média. Afin qu'un objet fasse partie d'une catégorie il doit répondre aux critères associés à la définition d'une catégorie. La formulation d'un critère de sélection permettant de désigner une catégorie d'objets reprend les mêmes conventions de notation que celles présentées dans le chapitre précédent (voir section 7.3.2).

Un problème, que nous devons traiter avec attention, est la situation dans laquelle un objet satisfait les critères associés à plusieurs catégories. Dans le chapitre précédent, consacré à la réutilisation et à l'annotation sémantique de scènes, le choix d'ajouter à un objet l'ensemble des métadonnées associé aux catégories dont l'objet fait partie, nous a paru naturel. Cependant, dans le cas de la mise en place de l'adaptation, il nous semble inadéquat d'appliquer directement à un objet l'ensemble des techniques d'adaptation associées aux catégories dont l'objet fait partie. En contrepartie, il doit être possible de spécifier des règles portant sur les résultats

obtenus en formalisant des opérations ensemblistes sur l'ensemble des objets issus de diverses catégories.

Afin de faciliter la formulation de ce genre de critères portant sur les objets appartenant à plusieurs catégories, nous donnons des noms à chaque catégorie. Ainsi, nous pouvons définir facilement des opérations ensemblistes (intersection, union, différence, différence symétrique) définissant des sous-ensembles d'éléments auxquels nous associons des techniques d'adaptation spécifique. Dans la Figure 8.2, nous synthétisons les règles définissant l'étendue (`<AdaptationScope>`) d'une règle d'adaptation. L'étendue est indiquée soit par le nom d'une catégorie (`<CategoryName>`), soit par une expression (`<CategoryExpr>`) constituée par des opérations ensemblistes (`<SetOp>`) entre les catégories d'objets. Les quatre opérations ensemblistes sont introduites par les termes suivants : `inter` pour l'intersection, `union` pour l'union, `diff` pour la différence et `sym_diff` pour la différence symétrique. Le nom d'une catégorie est introduit au moyen d'une déclaration (`<CategoryDef>`) qui associe au nom l'ensemble des critères (`<Criteria>` comme défini dans la Figure 6.14 du chapitre précédent). Cette déclaration se fait à l'extérieur de toute règle d'adaptation.

```
<AdaptationScope> := <CategoryName> | <CategoryExpr>
<CategoryExpr> := "("<CategoryExpr> <SetOp> <CategoryExpr>")" | <CategoryName>
<SetOp> := "inter" | "union" | "diff" | "sym_dyff"

<CategoryDef> := "Category" <CategoryName> ":=" {[<Criteria>]}
```

Figure 8.2 Définition des catégories d'objets et de l'étendue d'une règle d'adaptation.

Dans la Figure 8.3, nous présentons un exemple de définition de trois catégories sémantiques (`Arbres`, `Bâtiments` et `UFR`), ainsi que la définition partielle de deux règles d'adaptation. L'étendue de la première règle couvre les objets associés à la catégorie `Arbres`. L'étendue de la deuxième règle concerne les bâtiments qui n'accueillent pas d'UFR. Les termes Ti et Pi correspondent à la définition de la technique et des paramètres qui lui sont associés pour chacune des règles.

```
Category Abres      := {[Semantics!Catégorisation.catégorie="arbre"]}
Category Batiments  := {[Semantics!Catégorisation.catégorie="bâtiment"}
Category UFR        := {[Semantics!Catégorisation.catégorie="UFR"]}

Rule règle1 := (Arbres, T1, P1)
Rule règle2 := (Bâtiments diff UFR, T2, P2)
```

Figure 8.3 Exemple de définition partielle des catégories et des règles d'adaptation.

Dans la sous-section suivante, nous présentons la manière dont les techniques d'adaptation sont associées à une règle d'adaptation.

8.1.2. Technique d'adaptation

Chaque technique d'adaptation est identifiée par son nom et le type d'adaptation qu'elle met en œuvre. La typologie d'adaptation que nous avons identifiée en étudiant les travaux de la communauté 3D sur l'adaptation présentés dans le chapitre 0, montre une large palette de techniques d'adaptation.

Afin d'être utilisée correctement, chaque technique doit être caractérisée par l'ensemble des paramètres qui influencent le résultat du processus. De même, la définition d'une technique

d'adaptation doit indiquer le type de modélisation et le type des éléments supportés, ainsi que la méthode de modélisation associée aux résultats.

Nous proposons une description de ces caractéristiques au moyen d'un document XML représentant le profil d'adaptation. Le schéma XML décrivant l'organisation de ce document est présenté dans la Figure 8.4. Ainsi, la description d'une méthode est réalisée en indiquant les paramètres du processus (`Parameter`), la liste des formats supportés en entrée (`SupportedFormat`) et les types de formats supportés en sortie (`OutputFormat`) du processus d'adaptation. La description inclut également le nom de la méthode (`name`) et une `description` donnant, en langage naturel, les spécificités de la méthode d'adaptation. Les paramètres sont décrits par leur nom, par une référence à un concept d'une ontologie associée au paramètre, et par le type de valeur du paramètre.

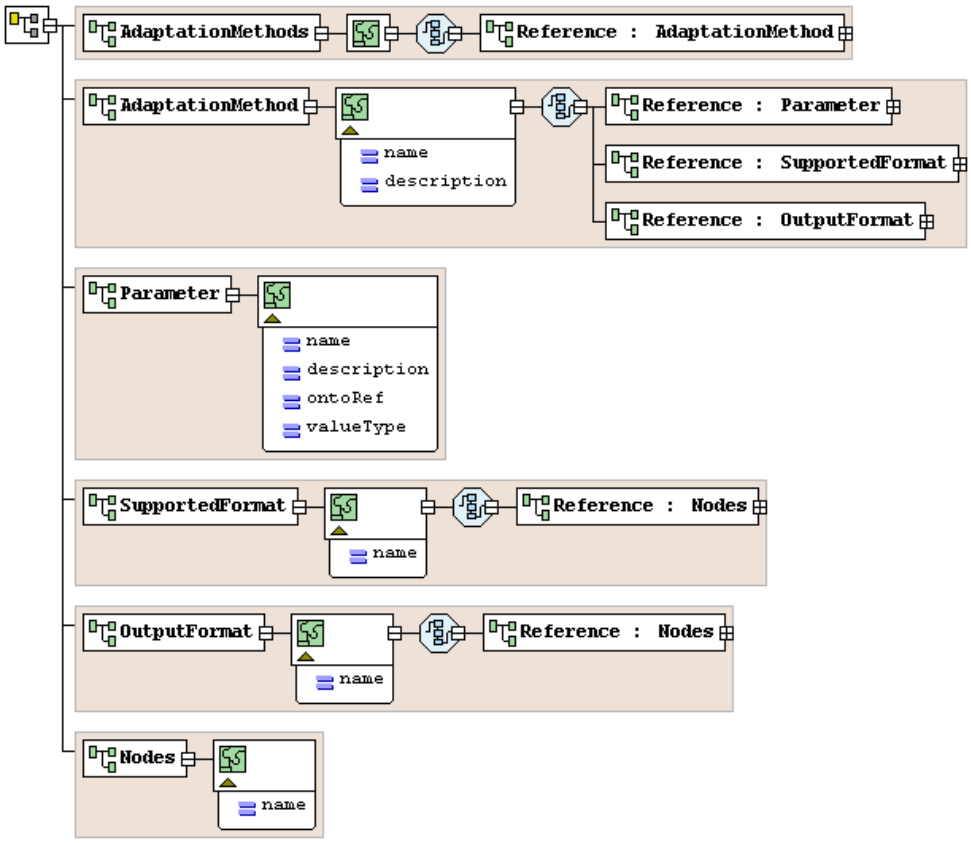


Figure 8.4 Schéma XML pour la définition d'un profil d'adaptation associé à une méthode d'adaptation.

Cette description comporte le minimum d'informations nécessaires pour l'intégration de la méthode au sein d'une stratégie d'adaptation. D'autres informations comme par exemple, des relations d'équivalence entre méthodes, les dimensions des données 3D affectées par la méthode, etc., peuvent être ajoutées à cette description car elles peuvent amener des réponses certaines dans le calcul de la meilleure stratégie d'adaptation. Toutefois, comme nous nous concentrons sur la description des stratégies et non pas leur calcul, nous ne détaillons pas ces aspects.

8.1.3. Les paramètres d'adaptation

Certains paramètres d'une technique de diffusion doivent être mis en correspondance avec les souhaits du concepteur de l'adaptation ou bien avec des informations extraites du contexte de diffusion.

Le contexte de diffusion est vu comme une collection de catégories de propriétés, chaque catégorie s'intéressant à un aspect (élément) de la diffusion d'une scène 3D. Nous retenons quatre dimensions de caractérisation du contexte : l'utilisateur, les contraintes matérielles, les contraintes logicielles, les conditions réseaux et la configuration de la plate-forme d'adaptation. Cependant, il est possible que les intérêts, en termes d'information contextuelle, varient en fonction des spécificités de l'application dans le cadre de laquelle les adaptations sont effectuées. L'ajout de nouvelles dimensions ou de nouvelles propriétés au sein des dimensions existantes est envisageable.

Afin de répondre à cette variabilité, nous considérons une formalisation du contexte à deux niveaux. Le premier niveau correspond à la définition des dimensions du contexte. Le deuxième recueille les listes de propriétés de chacune des dimensions. L'accès à un élément précis du contexte se réalise en indiquant le nom de la dimension et le nom long de la propriété, dans le cas où la propriété fait partie d'une propriété composée. Un système spécifique de gestion de contexte peut considérer la représentation interne qu'il souhaite. La formalisation proposée ici et basée sur deux niveaux d'information contextuelle ne constitue, en effet, qu'une vue sur le contexte, indépendante de la représentation sous-jacente.

Nous avons vu dans la sous-section précédente que le seul moyen de contrôler le processus d'adaptation est le choix des valeurs des paramètres de celui-ci. Chaque paramètre peut être associé à une valeur du contexte de diffusion ou bien à une fonction portant sur plusieurs éléments du contexte de diffusion (voir Figure 8.5).

```
<AdaptationParams>   :=  "[" <AdaptationParam> "]" |
                         "["<AdaptationParam>"] , ["<AdaptationParams>"]"
<AdaptationParam>    :=  <AdaptationParamName> "=" <Expr>
<Expr>               :=  <Numeric> | $<ContextElement>.<PorpertyConstruct> |
                          <Funct>(<ExprList>) | <Expr> <BinaryOp> <Expr>
<ExprList>           :=  <Expr> | <Expr>,<ExprList>
<ContextElement>     :=  User | Software | Hardware | Network | Application | …
<PropertyConstruct>  :=  <PropertyName>.<PropertyConstruct> | <PropertyName>
<PropertyName>       :=  literal
```

Figure 8.5 Notation BNF pour la mise en correspondance des paramètres d'adaptation avec l'information contextuelle.

8.1.4. Exemple de règles d'adaptation

Dans cette section, nous illustrons la formulation des règles d'adaptation en utilisant deux techniques d'adaptation : le filtrage, mentionné en utilisant l'identificateur EXCLUDE ; et la dégradation de la géométrie, dont l'identificateur est DEGRADE_GEOM. La deuxième technique d'adaptation implique l'utilisation d'un paramètre qui indique le type de dégradation voulu : BBOX ou FLAT. La valeur BBOX, qui désigne la boîte englobante de l'objet, exige le remplacement de l'objet par sa boîte englobante. La valeur FLAT demande le remplacement de l'objet par une surface plane correspondante à la projection de l'objet au sol.

Deux règles d'adaptation sont illustrées dans la Figure 8.6. Ces règles peuvent servir à préparer le déploiement d'une scène de type ville virtuelle sur un dispositif disposant de capacités limitées appartenant à un utilisateur dont le besoin principal, en termes d'information, est d'obtenir une vue générale de la scène dans le but d'évaluer la densité des bâtiments dans une région spécifique de la scène.

Catégories d'objets

Category Abres := {[Semantics!Catégorisation.catégorie="arbre"]}
Category Batiments := {[Semantics!Catégorisation.catégorie="batiment"}

Règles d'adaptation pour l'exclusion des *arbres* (a) et la substitution géométrique par la boîte englobante des *bâtiments* (b)

a) {[*Arbres*], [**EXCLUDE**], [ ]}

étendue **méthode** **paramètres**

b) {[*Batiments*], [**DEGRADE_GEOM**], [type="BBOX"]}

Figure 8.6 Exemple de règles d'adaptation.

Afin d'illustrer également la prise en compte du contexte de diffusion, nous présentons dans la Figure 8.7 deux règles d'adaptation. La première autorise la substitution des objets appartenant à la catégorie *Arbres* avec le code X3D contenu dans le fichier `arbres_simple.x3d`. La deuxième règle régit la personnalisation (`CUSTOM_APPEARANCE`) de la couleur (`color`) des bâtiments n'étant pas des UFR en fonction de la couleur préférée renseignée dans le profil utilisateur (`$User.preferredColor`)

Rule règle1 := (Arbres, SUBSTITUTE, [url = "arbres_simples.x3d"])
Rule règle2 := (Batiments *diff* UFRs, CUSTOM_APPEARANCE, [color =$User.preferredColor])

Figure 8.7 Exemple de règle d'adaptation avec prise en compte du contexte.

8.2. Stratégies d'adaptation

Une stratégie d'adaptation correspond à un ensemble de règles d'adaptation et à l'ordonnancement de ces règles. La stratégie d'adaptation peut être définie de manière statique par le concepteur ou calculée dynamiquement par un planificateur d'adaptation qui analyse les besoins de l'utilisateur et les caractéristiques de son dispositif d'accès.

Dans ce travail, nous nous intéressons uniquement à la description et la mise en œuvre d'une telle stratégie. Le calcul de celle-ci relève des caractéristiques de chaque domaine d'application.

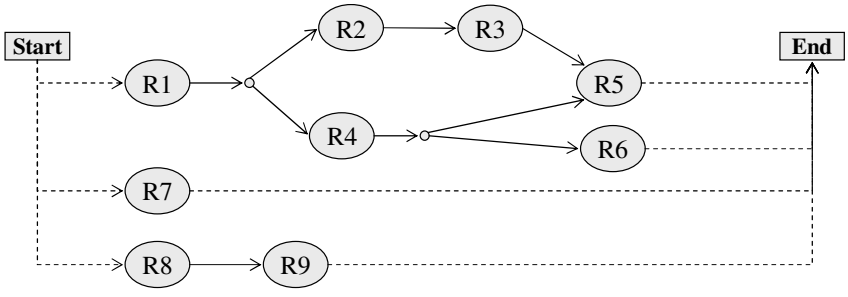


Figure 8.8 Stratégie d'adaptation formalisée sous la forme d'un graphe d'états.

Le formalisme adopté pour décrire l'ordonnancement des règles d'adaptation est constitué d'un graphe d'états (voir Figure 8.8). Chaque état correspond à l'application d'une règle. Deux états spéciaux sont introduits dans le système afin de traiter, de façon homogène, le début et la fin du processus associé à l'application de la stratégie d'adaptation. Un lien entre deux états représente le fait que la cible du lien attend la fin du processus associé à l'état source.

L'exécution d'un état ne s'effectue pas avant que tous les liens entrants soient traversés. L'exécution de la stratégie est accomplie lorsque l'état spécial *End* est atteint par tous les fils d'exécution. Les liens en pointillés sont introduits automatiquement par le moteur d'exécution de stratégies.

Le schéma XML illustré dans la Figure 8.9 rend compte de la description d'une stratégie d'adaptation sous la forme d'un document XML. Une stratégie (`Strategy`) est définie comme un ensemble de règles effectives d'adaptation (`EffectiveRule`). Une règle effective, à la différence d'une règle présentée dans la première section de ce chapitre, permet de décrire de manière explicite la portée de l'adaptation par l'intermédiaire d'une liste de repères de localisation (`LocatorList`) où des transformations de la scène sont nécessaires. De même, la liste de paramètres contient directement les valeurs précises qui sont transmises au processus d'adaptation associé.

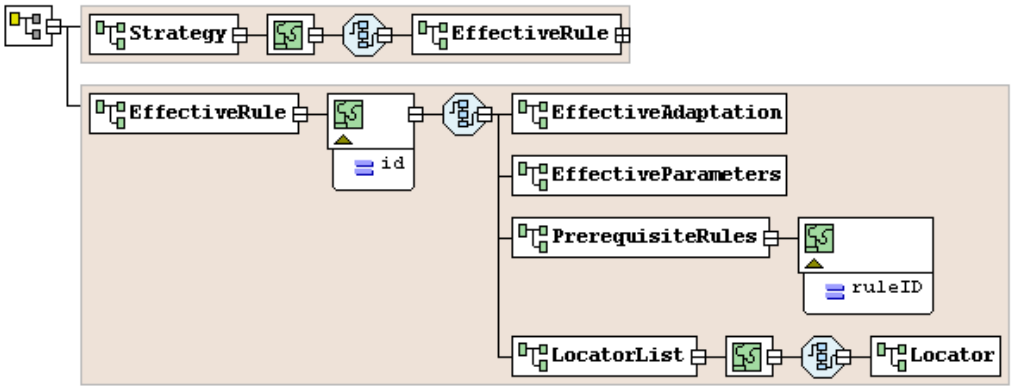


Figure 8.9 Schéma XML de description de stratégies d'adaptation.

La Figure 8.10 illustre le processus d'instanciation de règles effectives d'adaptation à partir des règles d'adaptation générales telles que présentées dans la section précédente. En partant de la règle d'adaptation générale, on identifie l'ensemble des repères de localisation d'instances conformes à la catégorie associée à la règle. Dans cet exemple, deux repères structurels sont identifiés comme étant sous l'emprise de la règle d'adaptation. Ces deux repères remplacent la première partie de la règle générale qui devient ainsi effective car désormais le sujet de l'adaptation est identifié. Le paramètre `$User.preferredColor` est également remplacé par sa valeur au moment de la génération de la règle.

Règle d'adaptation

Rule règle2 := (Batiments *diff* UFR, CUSTOM_APPEARANCE, [color =$User.preferredColor])

Repères de localisation des instances de la catégorie Bâtiments *diff* UFR

StructuralLocator(campus.x3d,{"xpath:///Group[DEF='BU']") et
StructuralLocator(campus.x3d, {"xpath:///Group[DEF='RU']"})

La couleur préférée de l'utilisateur (User.preferredColor) est le GRIS

Règles effectives

({
StructuralLocator(campus.x3d, {"xpath:///Group[DEF='BU']"),
StructuralLocator(campus.x3d, {"xpath:///Group[DEF='RU']")
}, CUSTOM_APPEARANCE, [color=GRIS])

Figure 8.10 Exemple de règle effective d'adaptation

Afin de construire le graphe d'ordonnancement, nous introduisons pour chaque règle un nœud dans le graphe. Les liens entre nœuds sont introduits après analyse des éléments `PrerequisiteRules` de chaque règle. Ces éléments contiennent les identifiants des règles qui doivent être appliquées en amont. Ainsi, pour chaque élément `PrerequisiteRules`, nous construisons un lien entre le nœud correspondant à la règle référencée par l'élément (`ruleID`) et le nœud associé à la règle courante.

Dans la section suivante nous présentons l'architecture de la plate-forme *Adapt3D* qui permet d'implémenter la chaîne d'opérations nécessaires à la formulation et à la mise en œuvre des adaptations.

8.3. Architecture Adapt3D

Au-dessus de la plate-forme 3DAF présentée dans le chapitre 0, nous construisons une série de modules (voir Figure 8.11), capables d'interpréter et d'exécuter des règles d'adaptation. L'ensemble de ces modules composent la plate-forme Adapt3D [Bilasco *et al.,* 2007c]. Nous décrivons ici comment le plan d'exécution des règles d'adaptation est établi. Le plan découle d'une stratégie d'adaptation. Nous focalisons notre discours sur la mise en œuvre d'une règle d'adaptation à la fois. Nous construisons un moteur d'adaptation capable d'adapter une scène à partir d'un ensemble de règles d'adaptation appliquées de manière séquentielle.

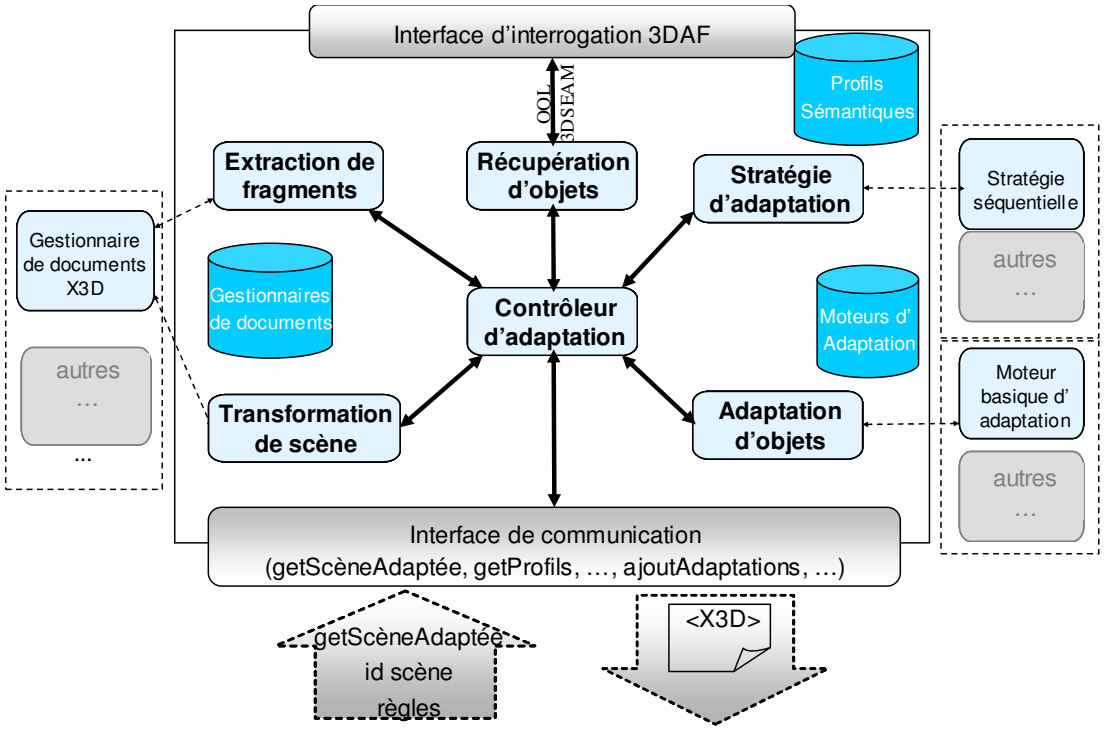


Figure 8.11 L'architecture de la plate-forme Adapt3D.

Les adaptations sont réalisées dans le cadre d'un processus en cinq étapes. Chaque étape est sous la responsabilité d'un module qui lui est dédié au sein du moteur d'adaptation. La collaboration entre modules est réalisée par le module *Contrôleur d'adaptation.*

La première étape, implémentée par le module *Récupération d'objets*, correspond à la construction de requêtes 3DSEAM qui visent la recherche d'objets qui répondent au critère de

sélection et la récupération des repères spatiaux ou structurels les définissant. Un ensemble de repères de localisation d'objets 3D est ainsi construit.

Dans un deuxième temps, pour chaque repère nous extrayons, à l'aide des repères récupérés, les objets dans leur état d'origine dans la scène en question. Le module d'*Extraction de fragments 3D* repose sur une série de gestionnaires de documents 3D enregistrés dans le système dans un registre spécifique (registre de *Gestionnaires de documents*). Chaque gestionnaire de document est associé à un format 3D et à un type d'encodage spécifique. Un gestionnaire exhibe deux fonctionnalités principales :

- l'interprétation de repères de localisation contenus dans le modèle 3DSEAM (repères spatiaux, repères structurels, repères spatio-structurels) ;
- le remplacement d'une partie de la scène par de nouveaux contenus.

Actuellement, nous supportons exclusivement l'extraction d'objets dans leur état initial à partir de documents X3D (*Gestionnaire de documents X3D*). Des gestionnaires additionnels peuvent être enregistrés auprès du système à travers les fonctionnalités (`addManager`/`deleteManager`) de l'*Interface de communication*.

En troisième lieu, le module *Stratégie d'adaptation* décide de la manière dont les transformations sont appliquées aux fragments 3D de la scène. Une bonne stratégie d'adaptation prend en compte les propriétés des objets, le contexte de diffusion, le domaine d'application, ainsi que les préférences de l'utilisateur. Cependant, nous focalisons notre attention sur la plate-forme permettant de mettre en œuvre les stratégies d'adaptation et non pas sur leur calcul. Afin de valider notre approche, nous proposons néanmoins une stratégie basique qui correspond à l'application en séquence de toutes les règles d'adaptation associées à la requête d'adaptation.

En quatrième lieu, chaque copie locale de l'objet dans l'état d'origine est transformée selon la méthode d'adaptation correspondant au critère de sélection. Si un objet est assujetti à plusieurs règles, les adaptations sont exécutées de manière séquentielle. Le module *Adaptation d'objets* s'appuie, pour la mise en place effective des transformations d'objets 3D, sur des programmes d'adaptation externes. Le registre *Moteurs d'adaptation* informe le module d'*Adaptation d'objets* 3D quel module d'adaptation externe prend en charge la méthode d'adaptation imposée. Le registre *Moteurs d'adaptation* enregistre pour chaque identifiant de méthode d'adaptation, les modules d'adaptation capables d'exécuter l'adaptation en question. L'*Interface de communication* gère la configuration du registre. À présent, le module externe *Moteur basique d'adaptation* que nous avons conçu, supporte les méthodes EXCLUDE – exclusion de l'élément et DEGRADE_GEOM – dégradation de la géométrie de l'objet. Nous illustrons le fonctionnement de ces méthodes dans la section suivante.

Finalement, le module *Transformation de scène* demande au gestionnaire du document de transformer la copie locale de la scène en remplaçant les éléments affectés par les transformations avec leurs variants adaptés. Le *Gestionnaire de documents X3D* que nous avons implémenté utilise XSLT [Clark, 1999] afin de substituer les éléments et de générer la scène adaptée. Le résultat de ce module est envoyé à l'utilisateur par l'*Interface de communication*.

8.4. Exemple d'adaptation

La Figure 8.12 présente le résultat de l'adaptation décrite par les deux règles d'adaptation de la Figure 8.6 appliquées sur une scène de petite taille modélisant une partie du campus universitaire.

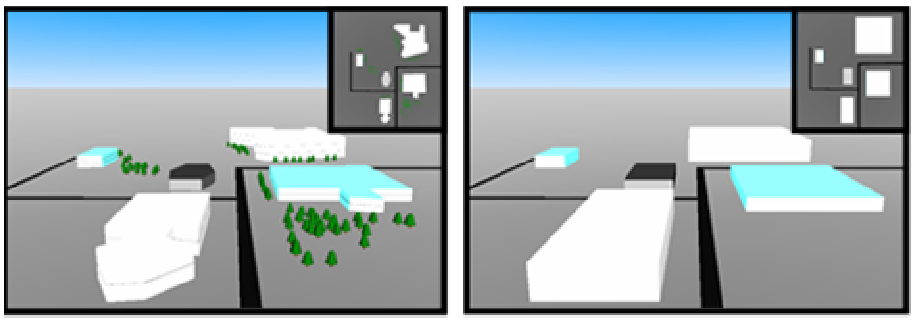


Figure 8.12 Résultat de transformation d'une scène urbaine en appliquant les règles de la Figure 8.6.

À gauche, la scène initiale est illustrée. La scène contient des bâtiments, des arbres et des routes. À droite, nous présentons le résultat du processus d'adaptation. La scène générée est approximativement 16 fois plus petite en termes de nombre de primitives géométriques de base que la scène initiale (uniquement 34 polygones dans la scène adaptée contre 540 polygones dans la scène initiale). Elle est également 23 fois plus petite en termes de taille mémoire sur le disque (uniquement 4kb contre 92kb).

L'application simultanée des deux règles d'adaptation affecte de manière conséquente la scène initiale car l'information relative aux contours des bâtiments est perdue. Si le dispositif d'accès de l'utilisateur dispose de très peu de ressources, cette solution nous semble un bon compromis. Cependant, parfois, l'application d'une partie des règles uniquement peut produire une scène qui satisfait les contraintes du contexte de diffusion. Par exemple, si nous appliquons uniquement la première règle (*EXCLUDE arbres)*, nous obtenons une scène 4 fois plus petite en termes de taille d'espace occupé sur le disque (22K) et 2,5 fois plus petite en termes de primitives géométriques de base (200 polygones) par rapport à la scène initiale. Le résultat est illustré dans la Figure 8.13. Cette scène offre un meilleur rendu visuel car l'information concernant les contours des bâtiments est maintenue. D'autres exemples d'adaptation sont présentés dans le chapitre suivant.

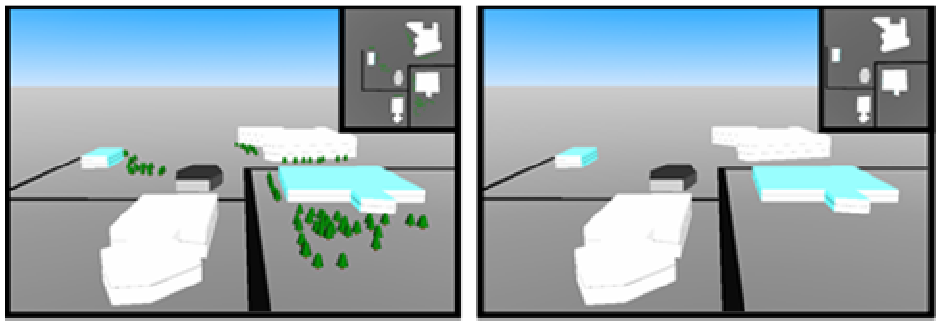


Figure 8.13 Résultat de transformation d'une scène urbaine en appliquant uniquement la première règle d'adaptation de la Figure 8.6.

Afin de pouvoir mettre en place ce type d'application partielle de règles, il est nécessaire de se munir de moyens qui évaluent clairement la distance entre l'état courant d'une scène et l'état idéal défini par rapport à un contexte de diffusion spécifique. Il faut également décider de ce qu'est l'état d'une scène, quelles sont les caractéristiques qui le décrivent et comment elles sont mesurées. Par exemple, dans le cas de la figure que nous venons de présenter, l'état de la scène est caractérisé soit par le nombre de primitives géométriques de base composant la scène, soit par l'espace disque occupé par la scène. Cependant, ces aspects ne sont pas traités dans le cadre de ce travail, qui visent principalement la prise en compte de la dimension sémantique des données 3D dans le processus d'adaptation.

8.5. Synthèse

La plate-forme Adapt3D que nous avons proposée permet de **définir des méthodes d'adaptation en relation avec les catégories sémantiques d'objets** que l'on retrouve dans une scène. Par rapport aux systèmes d'adaptations présentés dans l'état de l'art, notre solution offre la possibilité de **décrire de manière explicite au niveau documentaire les techniques d'adaptation applicables** au niveau d'une scène 3D.

Ainsi, indépendamment d'une implémentation spécifique de la plate-forme Adapt3D, le client (utilisateur ou application) – lorsqu'il demande la visualisation d'une scène – peut décrire les types d'adaptation qu'il souhaite appliquer à une catégorie d'objets particulière. Ce privilège n'est plus réservé uniquement au concepteur du moteur d'adaptation comme c'est le cas par exemple dans [Chittaro *et al.*, 2002] ou [Dachselt *et al.*, 2006].

En s'appuyant sur une architecture ouverte, les techniques d'adaptation qui sont supportées par la plate-forme évoluent en fonction des besoins d'adaptation pressentis pour les différents types de scènes (d'intérieur, espace ouverts, objets CAD, etc.), gérées par la plate-forme. **L'ajout de nouvelles techniques d'adaptation est indépendant des autres dimensions du processus d'adaptation.** Ainsi, nous pouvons introduire dans le système une palette de techniques d'adaptation qui couvrent l'ensemble des types de méthode d'adaptation (portant sur la géométrie, l'apparence, l'organisation logique, etc.) que nous avons illustrés dans l'état de l'art.

En effet, les fonctionnalités mises en œuvre dans le processus d'adaptation – *la localisation des objets, l'extraction des objets*, *le calcul de stratégie d'adaptation*, *l'adaptation individuelle des objets*, *la transformation de la scène* – ne dépendent pas directement l'une de l'autre. La communication entre les modules regroupant ces fonctionnalités repose sur l'échange de documents respectant les conventions de description des règles ou des stratégies d'adaptation.

Dans le chapitre suivant nous abordons la construction d'une application visant la réutilisation et l'adaptation de données 3D. Cette application repose sur un entrepôt 3DSEAM dont les données sont représentées au moyen de schémas MPEG-7.

9. Une expérimentation basée sur MPEG-7 pour la réutilisation et l'adaptation de données 3D

Dans ce chapitre, nous présentons les résultats expérimentaux obtenus en implémentant une application qui respectent les principes d'organisation et de fonctionnement des plates-formes 3DSDL et Adapt3D. L'expérimentation que nous proposons ici vise la réutilisation et l'adaptation des objets 3D modélisant le campus universitaire de Grenoble.

À ce jour, nous disposons d'une première version de ces plates-formes nous permettant de valider les principes qu'elles défendent à un niveau conceptuel : placer la sémantique au cœur des processus d'adaptation et de réutilisation, réutiliser conjointement des données 3D et leur sémantique, rendre explicite la formulation des règles d'adaptation.

Nous présentons tout d'abord une extension de MPEG-7 pour localiser des contenus 3D au sein d'une scène en nous appuyant sur les repères spatio-structurels que nous avons proposés dans le chapitre relatif au modèle 3DSEAM. La deuxième section présente les spécificités de la plate-forme 3DAF pour la gestion d'entrepôts 3DSEAM représentés en MPEG-7. La dernière section relate les expériences que nous avons menées autour d'une application couvrant le cycle de vie relatif à une modélisation basique du campus universitaire de Grenoble. L'application propose des interfaces graphiques pour faciliter la construction, l'annotation, la réutilisation et l'adaptation de différents objets 3D composant le campus universitaire.

9.1. Une extension MPEG-7 pour la localisation de contenus 3D

En raison de sa grande flexibilité et extensibilité, nous avons décidé de placer MPEG-7 au centre de nos travaux de réutilisation. Or, dans son état actuel, MPEG-7 ne couvre pas les objets 3D. Afin de faire face à ce problème, nous avons défini des descripteurs et des schémas de

description spécifiques pour les données 3D. Ces schémas de description ont été initialement publiés dans [Bilasco *et al.*, 2005b].

Afin d'intégrer la description de contenu 3D dans MPEG-7, nous avons analysé les *Descripteurs* (D) et les *schémas de description* (DS) de localisation prédéfinis. Nous avons proposé une extension au standard afin de localiser à l'aide des repères structuraux et spatiaux des objets dans les scènes 3D.

MPEG-7 supporte deux types principaux de descripteurs de localisation : le *MediaLocatorType* (similaire à un repère structurel) et le *RegionLocatorType* (similaire aux repères spatiaux).

Le *MediaLocatorType* pourrait être employé directement pour localiser structurellement les objets 3D. Cependant, le fait que le `MediaLocator` puisse seulement contenir un seul URI, ne répond pas aux exigences de localisation du modèle 3DSEAM (voir 5.2.4). Il ne permet pas non plus d'ajouter plusieurs chemins locaux (expressions XPATH) identifiant les parties d'un objet éparpillées dans le document. Plusieurs chemins locaux sont parfois nécessaires pour localiser un même objet (voir les exemples de la section 5.2.1). Pour répondre à ce besoin, nous avons proposé le `StructuralLocatorType` comme extension du *MediaLocatorType*.

```
<complexType name="StructuralLocatorType">
 <complexContent >
  <extension base="mpeg7:MediaLocatorType">
   <element name=”MediaLocator” type="anyURI" minOccurs="1" maxOccurs="1"/>
   <element name="LocalPath" type="anyURI" minOccurs="1" maxOccurs=”unbounded”/>
  </extension>
 </complexContent >
</complexType >
```

Figure 9.1 Définition d'un StructLocatorType.

Le schéma de description *RegionLocatorType* construit des repères de localisation en combinant des boîtes et des polygones rectangulaires. Nous proposons un `_3DRegionLocatorType` comme une extension pour définir des régions 3D. La localisation des objets 3D peut être obtenue en intégrant la description des régions 3D (la localisation géométrique) avec les repères structuraux (choix des unités du contenu) dans un schéma de description `_3DLocatorType` générique. La définition de ces nouveaux types se fait de manière similaire à la définition du *StructuralLocatorType.*

Afin de rendre possible l'annotation et l'indexation des instances de la classe `Object3D` du modèle 3DSEAM en MPEG-7, nous étendons, à l'aide de DDL, le DS *MultimediaSegmentType*. Un nouveau DS `_3DObjectType` correspondant à un segment (objet simple ou composite) dans une scène 3D est introduit dans la Figure 9.2.

Un *_3DObjectType* inclut ainsi un ou plusieurs repères de localisation (ligne 5 de la Figure 9.2). Les caractéristiques de l'apparence d'un objet 3D sont introduites par l'élément *Appearance* qui est constitué d'une liste de descripteurs visuels (*VisualDType*) de MPEG-7 (lignes 7-11). De la même façon, en utilisant des listes de propriétés nous introduissons la géométrie et le profil média. Des types de descripteurs spécifiques sont définis pour la géométrie (*GeometryDType* dérivé de *DType* – le type de base) et le profil média (*MediaProfileDType* dérivé également de *DType*). Les relations topologiques sont introduites comme une liste de descripteurs relationnels *TopologyRelType* (dérivés de *RelationType*). Ces *DType et RelationType* dérivés correspondent aux classes du modèle de descripteurs de 3DSEAM (voir Figure 5.20). Lorsqu'un nouveau descripteur est introduit dans le modèle de descripteurs, la construction d'un *DType* dérivé lui étant associé est de rigueur. L'attribut `entity_id` assure le

lien entre l'objet 3D et l'entité du monde réel qu'il représente. Le lien avec la sémantique locale à la scène se fait à travers l'élément `SemanticRef` hérité du type `MultimediaSegmentType`. Les éléments désignant la structure logique des objets 3D sont également hérités du *MultimediaSegmentType* qui prévoit un élément *MediaSourceDecomposition* pour cela.

```
 1: <complexType name="_3DObjectType">
 2:  <complexContent>
 3:   <extension base="mpeg7:MultimediaSegmentType">
 4:    <sequence>
 5:     <element name="MediaLocator" minOccurs="1" maxOccurs="unbounded" type="_3DLocatorType"/>
 6:     <element name="Appearance" minOccurs="0" maxOccurs="1">
 7:      <complexType>
 8:       <sequence>
 9:        <element type="mpeg7:VisualDType" maxOccurs="unbounded"/>
10:       </sequence>
11:      </complexType>
12:     </element>
13:     <element name="Geometry"  minOccurs="0" maxOccurs="1">...</element>
14:     <element name="Topology"  minOccurs="0" maxOccurs="1">...</element>
15:     <element name="MediaProfile"minOccurs="0" maxOccurs="1">...</element>
16:    </sequence>
17:    <attribute name="entity_id" type="anyURI"/>
18:   </extension>
19:  </complexContent>
20: </complexType>
```

Figure 9.2 Définition de 3DObjectType

La partie correspondante à l'instanciation des entités et de la sémantique repose entièrement sur le schéma sémantique (*SemanticBaseDS*) existant en MPEG-7.

Dans la section suivante, nous illustrons les choix faits lors de l'instanciation du modèle 3DSEAM en utilisant MPEG-7.

9.2. 3DAF mis en œuvre au-dessus d'un entrepôt MPEG-7

La première sous-section est consacrée à l'exposé des conventions de représentation des instances d'un entrepôt 3DSEAM exploitant les schémas de description disponibles dans MPEG-7. La traduction des requêtes 3DSEAM OQL en requêtes XQuery conformes à l'organisation des schémas de descriptions MEPG-7 est présentée ensuite.

9.2.1. Organisation de l'entrepôt 3DSEAM en MPEG-7

L'instanciation à l'aide du standard MPEG-7, que nous décrivons ici, vise à maintenir autant que possible l'indépendance entre les dimensions d'une entité (objet réel) : sa sémantique et ses matérialisations dans les documents multimédia.

Afin de garantir cette indépendance, pour chaque scène ou objet 3D, nous employons deux schémas de description indépendants : le schéma de description sémantique *Semantic Description Schemes* (pour `Semantics`) et le schéma de description de contenus multimédia *Multimedia Content Description Schemes* (pour `MultimediaFragment`).

La classe `Entity` et la classe `Semantics` de 3DSEAM sont instanciées en utilisant les schémas sémantiques de description MPEG-7. La sémantique associée à une entité est introduite par les éléments sémantiques de MPEG-7 regroupés dans le schéma de description *Semantic Base DS*. Les éléments sémantiques disponibles dans ce schéma de description sont : les libellés, les définitions, les propriétés et les relations.

Le schéma de description de contenu multimédia assure la décomposition de la scène en objets 3D indépendants caractérisés chacun en utilisant *_3DObjectType*.

Afin d'illustrer par un exemple les choix d'instanciation nous considérons la scène et les annotations présentées dans la Figure 6.4 (page 143).

La Figure 9.3 et la Figure 9.4 exposent l'instanciation XML du modèle 3DSEAM en utilisant l'extension que nous proposons pour MPEG-7. La description sémantique est illustrée par la Figure 9.3, tandis que la description des fragments multimédia est donnée par la Figure 9.4. L'exemple porte sur l'indexation de la scène décrite dans la Figure 6.3. Dans la Figure 9.3, une première description (sémantique) présente les entités principales suivantes : le corps D de l'ENSIMAG *ENSIMAG_D* (lignes 5-15), le corps E de l'ENSIMAG *ENSIMAG_E* (lignes 27-28) et les *passerelles* (lignes 29-38). *Le bâtiment gris* et *les passerelles* sont divisés en sous-entités. Seules les définitions des *étages* du bâtiment gris sont présentées dans la Figure 9.3 : le rez-de-chaussée *ENSIMAG_D0* (lignes 16-24), le premier étage *ENSIMAG_D1* (lignes 25-26), etc.

Nous employons les libellés MPEG-7 (*Label*) pour introduire et attacher les profils sémantiques (le profil *Catégorisation* lignes 6, 17 et 30 dans la Figure 9.3) aux entités. Pour chaque profil sémantique un élément *Label* est instancié. Afin de faciliter l'utilisation (et plus particulièrement, la formulation de requêtes), un alias est défini par l'élément fils *Name* de *Label*. Les propriétés sémantiques d'un profil sont présentées par des éléments *Term* (lignes 8, 19 et 32 dans la Figure 9.3). Le nom de la propriété (spécifique au profil sémantique défini) est déclaré par le sous-élément *Name* (lignes 9, 20, 33). La valeur de la propriété est indiquée par le sous-élément *Definition* (lignes 10, 21, 34). Bien que non illustrées ici, les relations sémantiques peuvent être introduites en utilisant les attributs de *termId* et *relation* de l'élément *Term*. Le *termId* pointe vers la propriété associée et l'attribut *relation* indique la nature de la relation.

```
 1:<Mpeg7 ...>
 2: <Description xsi:type="SemanticDescriptionType">
 3:  <Semantics>…
 4:   <SemanticBase xsi:type="ObjectType">…
 5:    <Object id="ENSIMAG_D">
 6:     <Label>
 7:      <Name>Catégorisation</Name>
 8:       <Term>
 9:        <Name>catégorie</Name>
10:        <Definition>bâtiment</Definition>
11:       </Term>
12:     </Label>
13:     <ObjectRef refid="ENSIMAG_D0"/>
14:     <ObjectRef refid="ENSIMAG_D1 "/>…
15:    </Object>
16:    <Object id="ENSIMAG_D0">
17:     <Label >
18:      <Name>Catégorisation</Name>
19:      <Term>
20:       <Name>catégorie</Name>
21:       <Definition>étage</Definition>
22:      </Term>…
23:     </Label>
24:    </Object>
25:    <Object id="ENSIMAG_D1">
26:    …</Object> …
27:    <Object id="ENSIMAG_E">
28:    …</Object>
29:    <Object id="Passerelles">
30:     <Label >
31:      <Name>Catégorisation</Name>
32:      <Term>
33:       <Name>catégorie</Name>
34:       <Definition>passerelle</Definition>
35:      </Term>
36:     </Label>
37:     <ObjectRef refid="Niveau1"/>
38:     <ObjectRef refid="Niveau2"/>…
39:    </Object>
40:    …
```

Figure 9.3 Représentation MPEG-7 d'entités et d'annotations sémantiques.

Quelques descriptions de fragments multimédia (instance de *_3DObjectType*) sont présentées dans la Figure 9.4. Nous nous concentrons sur les fragments liés à l'objet *ENSIMAG_D* : le segment *BâtimentGris* (lignes 4-17) et le segment correspondant au ENSIMAG_D1 – *1erGris* (lignes 18-26). Le bâtiment entier est défini en utilisant un repère structurel simple (lignes 6-8), alors que le premier étage est référencé avec un repère composé : structurel (lignes 20-22) et géométrique (lignes 24-25). La structure logique (*LogicalStructure*) de la scène est présentée en utilisant l'élément prédéfini *MediaSourceDecomposition* (ligne 14).

```
 1: <Mpeg7 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 2:   <Description xsi:type="ContentEntityType">
 3:     <MultimediaContent xsi:type="MultimediaType">
 4:       <Multimedia xsi:type="_3DObjectType" id="BâtimentGris"
 5:         entity_id="ENSIMAG_D">
 6:       <Locator xsi:type="StructuralLocatorType">
 7:         <MediaLocator>buildings.x3d</MediaLocator>
 8:         <LocalPath> xpath://child::*[@DEF='GRIS'] </LocalPath>
 9:       </Locator>
10:     <MediaProfile>...</MediaProfile>
11:     <Geometry>...</Geometry>
12:     <Appearance>...</Appearance>
13:     <MediaSourceDecomposition>
14:       <MultimediaRef idref="RDCGris" />
15:       <MultimediaRef idref="1erGris />
16:     </MediaSourceDecomposition>
17:   </Multimedia>
18:   <Multimedia xsi:type="_3DObjectType" id="RDCGris"
19:         entity_id="ENSIMAG_D0">
20:     <Locator xsi:type="_3DLocatorType">
21:       <MediaLocator>buildings.x3d</MediaLocator>
22:       <LocalPath> xpath://child::*[@DEF='GRIS'] </LocalPath>
23:       <SpatialLocator xsi:type="Box3D" x="0" y="-3"
24:                   z="0" l="12" h="1" w="2" />
25:     </Locator>
26:   </Multimedia>
27:   <Multimedia xsi:type="_3DObjectType" id="1erGris"
28:                 entity_id="ENSIMAG_D1">…
29:   </Multimedia>
30:   </MultimediaContent>
31: </Description>
32: </Mpeg7>
```

Figure 9.4 Représentation MPEG-7 de l'entrepôt de description de fragments multimédia de 3DSEAM.

Le lien entre les objets 3D et les entités correspondantes est fait en indiquant dans l'attribut `entity_id`, qui apparaît dans la description de chaque objet 3D (ligne 4 pour le `BâtimentGris`), l'identifiant de l'entité associée. La sémantique peut alors être mise en relation avec chacun des segments multimédia.

9.2.2. Interrogation de l'entrepôt avec XQuery

Dans cette section, nous discutons de la mise en œuvre des modules du 3DAF en utilisant une représentation MPEG-7 pour l'entrepôt d'annotations. Comme illustré précédemment, dans cette implémentation nous ne retenons que deux bases de connaissances MPEG-7 : une pour les caractéristiques multimédia et une pour les entités et la sémantique. La représentation des entités et de la sémantique au sein d'une même base n'est pas conforme à celle préconisée dans la plate-forme 3DAF. De plus, les profils sémantiques ne partagent pas leurs propriétés communes ce qui peut mener à des doublons potentiellement générateurs d'incohérence. Déstinés à être corrigés prochainement, ces faits n'ont pas une importance capitale pour cette première implémentation qui valide l'utilité de la plate-forme 3DAF (garantir l'indépendance vis-à-vis des choix spécifiques de représentation des connaissances).

Afin de pouvoir interroger les instances MPEG-7, le module *Gestionnaire de requêtes* et le module *Gestionnaire d'annotations* de 3DAF doivent traduire les requêtes 3DSEAM en requêtes conformes à MPEG-7. Des langages d'interrogation spécifiques pour MPEG-7 existent [Fatemi *et al.,* 2003], mais ils ne ciblent que la dimension audio-visuelle du contenu multimédia et, à notre connaissance, ils ne sont pas extensibles.

Puisque MPEG-7 est un langage XML, nous pouvons exploiter XQuery afin de manipuler les instances 3DSEAM. Dès lors, les conventions d'adressage des concepts et des propriétés 3DSEAM doivent être traduites en expressions XPath [Clark *et al.*, 1999]. Les expressions XPath pointent vers les éléments ou attributs dans les fichiers d'instanciation ou le concept ou la propriété 3DSEAM sont stockés. Par exemple, l'expression `Semantics!Catégorisation.catégorie` (la propriété `catégorie` du profil sémantique de `Catégorisation`) est traduite en `//Object/Label[Name/text()='Catégorisation']/Term[Name/text()='catégorie']`. Ensuite les constructions SELECT FROM WHERE doivent être traduites en expressions FLWR. Dans la Figure 9.5, nous montrons la réécriture d'une requête 3DSEAM OQL en requête XQuery. Les numéros montrent les correspondances entre l'écriture OQL et les conventions de représentation des différentes classes 3DSEAM en MPEG-7 décrite à l'aide des expressions XQuery.

Trouver la localisation des objets qui appartiennent aux catégories "bâtiment" ou "étage".

3DSEAM OQL

```
SELECT loc FROM Object3D obj, obj->MediaLocator loc,
       obj->Entity.Semantics s
       WHERE s!Catégorisation.catégorie="bâtiment"  or s!Catégorisation.catégorie="étage"
```

XQuery

```
for $obj in doc("fragments.mpeg7.xml")//Multimedia["_3DObjectType"= @xsi:type)]
let $loc : = $obj/MediaLocator
let $s := doc("entities.mpeg7.xml")//Object[@id=$obj/@entity_id]
let $catégorie:=$s/Label[contains(Name/text(),"Catégorisation")]/Term[contains(Name/text(),"catégorie")]
where (contains($catégorie/Definition/text(),'bâtiment')  or contains($catégorie/Definition/text(),'étage'))
return
  <item type="_3dseam::Object3D.MediaLocator">{$loc}</item>
```

Figure 9.5 Requêtes XQuery sur le modèle 3DSEAM instancié avec MPEG-7.

9.3. Gen3D pour la génération adaptée des environnements urbains

L'expérimentation que nous proposons ici vise la réutilisation et l'adaptation dans le cas des applications de type SIG (Systèmes d'Information Géographique) et notamment celles visant la gestion des environnements 3D urbains comme les villes virtuelles. En effet, ce type d'application devient de plus en plus populaire. À l'image d'Aix en Provence, Marseille, Paris, Rennes et Toulouse[42], le nombre de villes disposant d'une modélisation 3D du bâti ne cesse de croître. Les modèles 3D des villes correspondent à de très grands espaces physiques qui contiennent une masse conséquente d'information. Ceci nous permet d'anticiper des besoins émergents en termes d'adaptations et de réutilisation car, généralement, lorsque l'utilisateur navigue dans un espace 3D, il cible ses recherches d'un point de vue sémantique (cherche une banque) et spatiale (au centre-ville).

L'application Gen3D permet de générer des scènes 3D comportant des objets 3D issus d'une modélisation simplifiée du campus universitaire de Grenoble.

L'application fournit deux principales interfaces utilisateurs permettant respectivement de rajouter des nouveaux objets 3D à la collection des objets modélisant les entités de campus et de générer des scènes 3D dont le contenu et le niveau de détail des objets sont conformes aux attentes des utilisateurs.

Nous organisons la présentation autour des fonctionnalités des interfaces de création et de génération adaptée en présentant pour chacune d'entre elles les choix techniques réalisés et le processus d'interaction avec les plates-formes 3DAF, 3DSDL et Adapt3D.

9.3.1. Interface de construction de nouveaux éléments

Nous facilitons la création de scènes 3D représentant un environnement urbain, en proposant aux utilisateurs une approche de création en deux étapes :

- le choix d'un type d'objet prédéfinis, et
- la définition des contours des objets.

[42] http://v3d.pagesjaunes.fr/

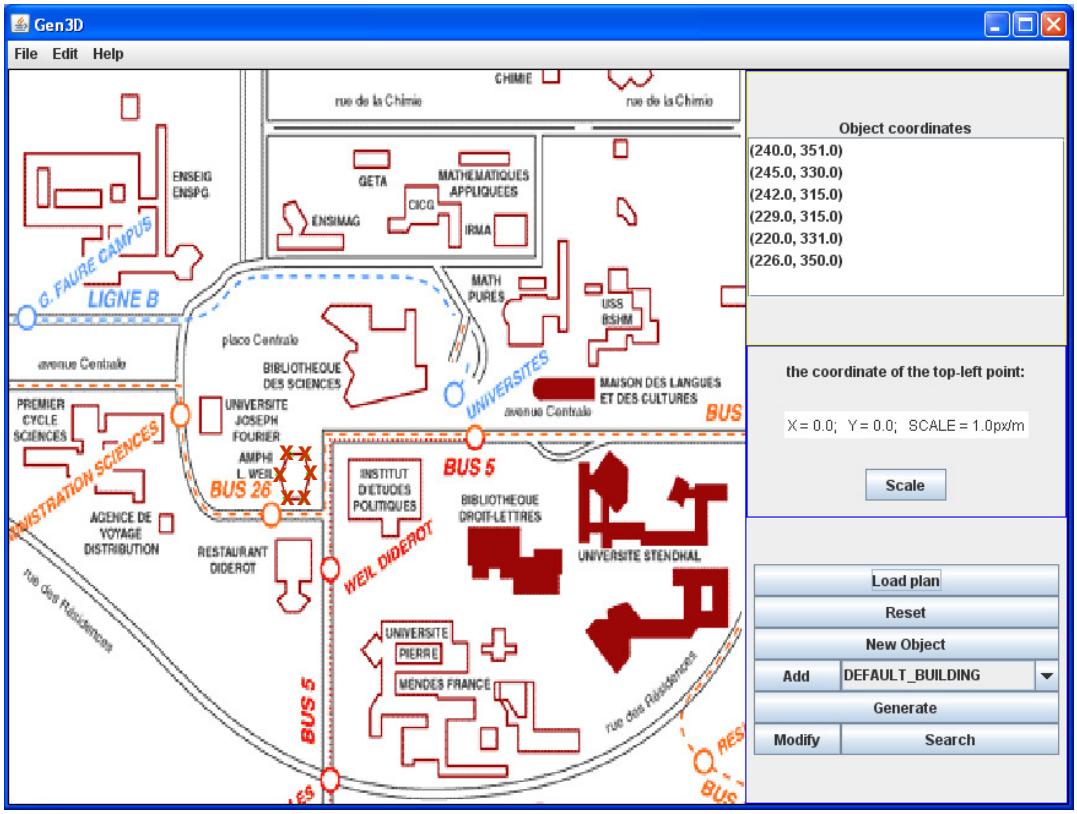


Figure 9.6 Interface de construction de nouveaux objets 3D de l'application Gen3D.

L'interface de construction est présentée dans la Figure 9.6. La partie gauche présente le plan de l'espace réel que l'on souhaite modéliser et qui est un support pour la disposition spatiale des objets que l'on veut créer. La partie droite recueille des informations concernant les coordonnées du polygone associé à l'objet 3D en cours de création. Également la partie droite permet de décider du type d'objet 3D que l'on souhaite créer.

Types d'objets supportés

Dans la Figure 9.7 nous présentons les types d'objets que nous avons retenus pour la construction des scènes urbaines. Pour chaque type d'objet nous avons construit un prototype X3D qui permet de faciliter son instanciation.

De fait que l'application contrôle les propriétés géométriques et d'apparence des divers objets 3D instanciés, l'alimentation de la base de connaissances relative aux fragments multimédia est réalisée automatiquement. En ce qui concerne la dimension sémantique, lors de la création d'un objet 3D, nous laissons le choix aux utilisateurs de rajouter des informations sémantiques.

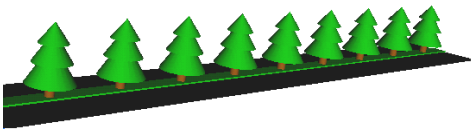
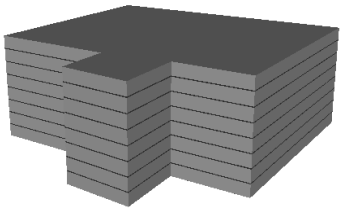
	DEFAULT BUILDING : bâtiment construit à partir d'au moins trois coordonnées, de la hauteur et de la couleur.
	DEFAULT ROAD : route construite à partir de deux coordonnées (point gauche et point droit) et de la largeur. La route est présentée sous forme de boîte noire.
	DEFAULT TREE : arbre construit à partir d'une coordonnée et de la hauteur. Un arbre se présente sous la forme d'un sapin représenté par un cylindre marron et par trois cônes verts.
	DEFAULT ROADTREE : route avec des arbres au centre construite à partir de deux coordonnées (point gauche et point droit) et sa largeur. Elle est présentée sous forme d'une boîte de couleur noire recouverte au centre d'une boîte verte et d'arbres.
	DEFAULT STOREBUILDING : bâtiment à plusieurs étages construit à partir d'au moins 3 coordonnées, de la hauteur, du nombre d'étages et de la couleur. Chaque étage est construit à partir de DEFAULT BUILDING et les séparations entre étages sont des DEFAULT BUILDING noir de très petite taille (0.2).
	DEFAULT TREES : plusieurs arbres construits en saisissant les coordonnées de chaque arbre et une hauteur commune à tous les arbres. Les arbres sont représentés comme DEFAULT TREE.

Figure 9.7 Types d'objets construits avec l'interface Gen3D

Interface de saisie d'annotations

L'interface de saisie d'information sémantique est présentée dans la Figure 9.8.

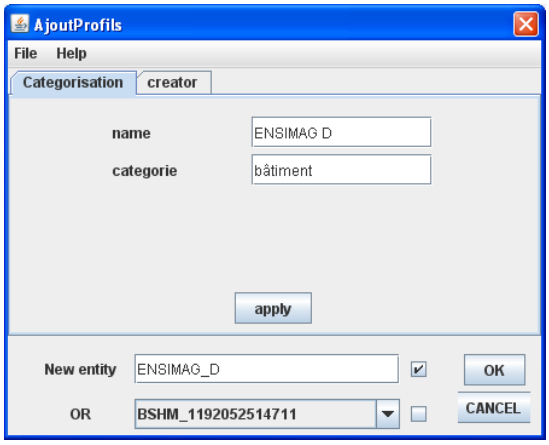


Figure 9.8 Interface de saisie d'annotations de Gen3D.

Les différents onglets de l'interface correspondent aux différents profils sémantiques. Chaque propriété peut être saisie à l'aide des champs texte. La version actuelle ne supporte pas la spécification de propriétés structurées. La partie basse de l'interface permet de saisir le nom de l'entité ou bien d'associer l'objet que l'on vient de créer à une entité existante dans l'entrepôt de description.

La génération de cette interface se fait dynamiquement en analysant les profils sémantiques disponibles inclus dans le modèle de descripteurs 3DSEAM. Nous nous appuyons sur une représentation XML de la composition de profils sémantiques à l'image de la description de profils sémantiques réalisée dans le cadre de la plate-forme 3DSDL (voir Figure 7.2). En parcourant ce fichier nous instancions de nouveaux éléments graphiques `JTabbedPane` afin d'avoir un onglet pour chaque profil sémantique.

Lorsque l'utilisateur valide la saisie des profils sémantiques l'entité est introduite dans l'entrepôt MPEG-7 en faisant appel aux méthodes proposées par le module *Gestionnaire d'annotations* (voir section 6.3).

9.3.2. Interface de génération automatique de scènes 3D personnalisées

L'interface de génération automatique de scènes (voir Figure 9.9) permet aux utilisateurs de définir les catégories sémantiques d'objets à inclure dans la scène.

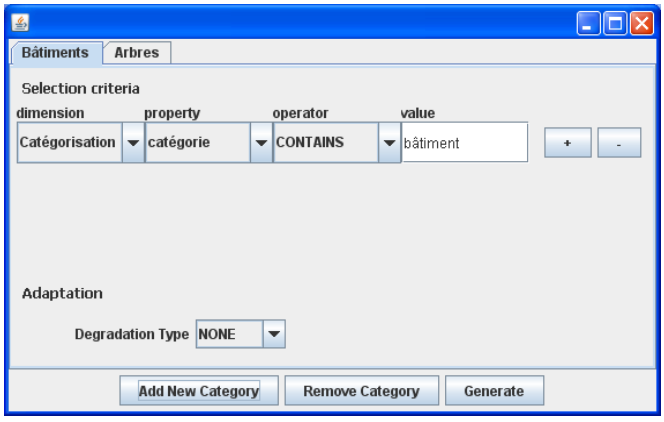


Figure 9.9 Interface de génération adaptée de scènes.

L'utilisateur indique la liste de propriétés et les valeurs qui caractérisent chaque catégorie d'objets. Les propriétés sont définies par rapport aux divers profils sémantiques et dimensions de caractérisation relatives aux fragments multimédia (géométrie, apparence).

À chaque catégorie d'objets l'utilisateur peut associer un type de dégradation géométrique. L'application supporte quatre types de dégradations : BBOX, SPHERE, FLAT_2D, FLAT_3D illustrées dans la Figure 9.10.

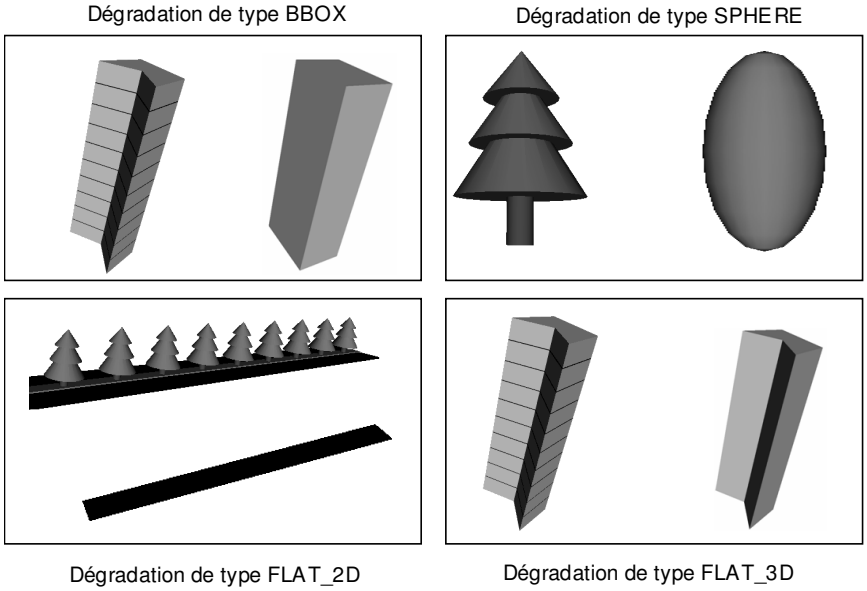


Figure 9.10 Types de dégradations supportés.

9.3.3. Scénario de génération adaptée

Lorsque l'utilisateur demande la génération d'une nouvelle scène, dans un premier temps l'application récupère les informations relatives aux différentes catégories d'objets et nous construisons une requête de réutilisation (à l'image de celles de la Figure 7.4). Cette requête est transmise à 3DSDL afin d'assembler une scène qui inclut l'ensemble des objets appartenant aux différentes catégories, chacun comportant comme annotation sémantique l'identifiant de l'entité sémantique à laquelle il est associé.

Cette scène intermédiaire est stockée temporairement dans son état initial, avant de demander à la plate-forme Adapt3D d'appliquer les règles d'adaptation saisies par l'utilisateur. Afin de se conformer aux usages de la plate-forme Adapt3D, les objets 3D de la scène intermédiaire sont temporairement ajoutés dans l'entrepôt 3DSEAM. Ces nouveaux objets (car appartenant à une nouvelle scène) sont associés aux entités auxquelles les anciens objets étaient initialement associés. Cette association est réalisée automatiquement en tenant compte de l'identifiant de l'entité associée à chaque objet dans la scène intermédiaire.

La scène ainsi obtenue est délivrée sous la forme d'un fichier X3D à l'utilisateur. Les objets et la scène intermédiaire sont retirés du système. Les exemples présentés dans la Figure 8.12 (page 193) et la Figure 8.13 (page 193) ont été générés avec cette application.

Conclusion

Le domaine d'étude de cette thèse est la caractérisation sémantique de données 3D en vue de l'amélioration des techniques de réutilisation et d'adaptation appliquées aux scènes 3D.

- la réutilisation (d'objet 3D) vise à augmenter la productivité lors du développement de données 3D ;
- l'adaptation vise le déploiement adéquat d'une donnée 3D en rapport avec le contexte applicatif.

Nous proposons des solutions aidant les concepteurs d'applications 3D à mettre en place des techniques d'adaptation et de réutilisation.

I Rappel des principaux manques de l'état de l'art

L'étude des modèles et méthodes s'intéressant à l'association d'information sémantique aux données 3D, au rôle de la sémantique dans le cycle de vie des données 3D et notamment à la réutilisation et à l'adaptation, nous a permis de dresser un panorama montrant les acquis et les manques de l'existant.

Nous avons remarqué un engouement certain autour de la représentation sémantique, cependant l'utilisation de celle-ci se réalise le plus souvent dans le cadre d'une application spécifique. Du point de vue de la caractérisation des données 3D, nous avons retenu les observations suivantes :

- les critères de bas niveau (géométriques, d'apparence, topologiques) sont relativement faciles à extraire, cependant ils sont peu exploitables pour la réutilisation et l'adaptation de données 3D ;
- l'association des informations sémantiques à un objet de la scène se fait uniquement à travers l'identifiant de l'objet. Ceci a pour conséquence que les objets n'ayant pas d'identifiant explicitement défini sont exclus ;
- aucune des propositions que nous avons étudiées ne s'intéresse à la caractérisation d'une donnée 3D dans sa globalité.

Les mécanismes de réutilisation sont peu formalisés et l'apport de la sémantique est très peu exploité. En ce qui concerne les principales propositions s'intéressant à la réutilisation de données 3D nous remarquons :

- la réutilisation quasi-inexistante de la sémantique des fragments multimédia ;

- l'utilisation des identifiants des objets pour indiquer les objets à réutiliser ;
- le manque de moyens d'expression des critères de sélection des objets ;
- le manque de description de moyen d'assemblage des objets sélectionnés dans les nouvelles scènes ;
- l'accent mis sur la visualisation des données, en négligeant les aspects relatifs à l'enrichissement sémantique de la scène générée.

En dépit de leur grand nombre et de leur diversité, les techniques d'adaptation sont généralement appliquées de manière ponctuelle, en étroite relation avec une solution logicielle spécifique. Les concepteurs de scènes ne disposent pas d'outils et de modèles permettant de décrire des stratégies d'adaptation indépendamment d'une application spécifique. D'un point de vue adaptation, nous notons :

- la grande variété de techniques d'adaptation ;
- la non prise en compte de la sémantique dans le processus d'adaptation ;
- l'absence d'un moyen de formaliser les techniques permettant de définir la manière dont les objets, ou les classes d'objets au sein d'une scène 3D, sont adaptés ;
- la dépendance des solutions vis-à-vis des choix du concepteur de l'adaptation (listes de variants, identification directe des objets) ;
- l'absence de moyens pour construire des stratégies d'adaptation génériques qui puissent s'appliquer à toute une famille de scènes (par exemple, les scènes modélisant des environnements urbains).

La formalisation d'un tel moyen de description permet aux concepteurs ou aux utilisateurs d'une scène, ne disposant pas forcément des compétences requises, de réaliser des transformations sur la scène. Ces transformations visent à réduire la complexité de la scène, ou bien qui visent à personnaliser certains de ses aspects (apparence de certains objets, filtrage sémantique, etc.).

II Bilan

Par rapport aux manques identifiés dans l'état de l'art, nous proposons nos solutions de réutilisation et d'adaptation sur critères sémantiques. Notre contribution est organisée en quatre parties.

II.1. 3D SEmantic Annotation Model (3DSEAM)

3DSEAM est un modèle générique permettant de caractériser *la géométrie*, *l'apparence*, *la dimension document ou profil média* et les informations relatives aux objets du monde réel modélisés par les objets 3D d'une scène.

Le modèle est organisé en trois parties :

- la partie *fragment multimédia* – qui introduit les caractéristiques géométriques, d'apparence, topologiques et média des données 3D ;
- la partie *entité du monde* – qui fait le lien entre les entités du monde réel et leurs matérialisations (*i.e.* les fragments) dans les documents multimédia en tant que données 3D ;

- la partie *sémantique* – qui couvre la sémantique locale d'un fragment (propre à la scène) et la sémantique générale d'une entité du monde (indépendante de la scène).

3DSEAM se propose comme une solution générique qui ne dépend pas d'un domaine d'application spécifique. Le modèle de descripteurs sous-jacent que nous avons défini permet au modèle de prendre en compte une large variété d'informations issues de domaine d'application divers. Ainsi, à titre d'exemple, les propositions que nous avons étudiées dans l'état de l'art pourraient exploiter le modèle 3DSEAM afin de modéliser leurs besoins en termes de sémantique.

II.2. 3D Annotation Framework (3DAF)

3DAF est une plate-forme qui permet d'exploiter les connaissances renseignées selon le modèle 3DSEAM et organisées dans un entrepôt de données. 3DAF introduit une couche abstraite qui assure l'indépendance entre une représentation particulière des instances du modèle 3DSEAM et le moyen d'accéder à l'information. Les fonctionnalités de la plate-forme sont exprimées à travers des notations capables de tenir compte de l'hétérogénéité qu'il peut y avoir en termes de propriétés au niveau des instances.

Une extension du langage de requête OQL a été proposée pour le modèle 3DSEAM. Afin de simplifier l'écriture des requêtes, nous avons introduit des notations (de navigation dans la structure prédéfinie du modèle, d'accès aux éléments d'un profil sémantique, etc.) en étroite relation avec l'organisation du modèle 3DSEAM.

Par rapport, aux travaux que nous avons présentés dans l'état de l'art, cette plate-forme permet d'utiliser la sémantique et d'autres caractéristiques des objets 3D dans un cadre plus large que celui d'une application spécifique.

Cette plate-forme constitue une base au-dessus de laquelle des solutions spécifiques aux différents domaines d'application de la 3D peuvent alors être utilisées pour exploiter les connaissances sémantiques et les caractéristiques relatives aux données 3D.

II.3. 3D Semantic Data Library (3DSDL)

3DSDL est une plate-forme pour faciliter la *recherche et la réutilisation de contenus 3D sémantiques*. Ces opérations sont mises en œuvre au-dessus de la plate-forme 3DAF. Nous proposons un langage de description de patrons de scènes 3D dont le contenu inclus est en concordance avec les critères sémantiques, géométriques et d'apparence formulés par l'utilisateur.

La réutilisation des fragments suppose deux actions distinctes : l'extraction de fragments de la scène initiale, et l'inclusion de ceux-ci au sein de la nouvelle scène.

Par ailleurs, la plate-forme s'appuie sur une réutilisation à base de catégories d'objets construites sur critères sémantiques. L'usage des patrons de présentation de scènes confère aux utilisateurs de la plate-forme la possibilité de personnaliser la manière dont les contenus réutilisés sont assemblés. Les patrons sont décrits en tant que documents X3D dont le contenu, extrait des données au format 3DSEAM, est introduit en utilisant les éléments de base de X3D.

Par rapport aux travaux liés à la réutilisation que nous avons présentés dans l'état de l'art, la plate-forme 3DSDL est extensible du point de vue des outils de sélection de contenu, des types d'encodage de données 3D et des méthodes d'assemblage de résultats.

II.4. Adapt3D

Adapt3D est une plate-forme qui supporte la mise en place des adaptations sur les documents 3D. Nous proposons une solution qui permet, pour chaque catégorie d'objets au sein d'une scène, d'indiquer la technique d'adaptation la plus appropriée. Les catégories sont définies à l'aide de critères portant sur les caractéristiques des objets 3D (sémantique, géométrie, …).

Le processus d'adaptation doit combiner différentes techniques comme la dégradation de la géométrie ou de la texture, le filtrage d'objets, la substitution, l'accentuation de propriétés de certains objets (modification du niveau de transparence, etc.) afin de produire des résultats adéquats et sur mesure.

La capacité de combiner plusieurs techniques d'adaptation pour la même scène est cruciale parce que tous les objets ne peuvent pas être adaptés d'une manière optimale dès lors qu'on a recours à une technique unique.

Les étapes du processus d'adaptation – *la localisation des objets, l'extraction des objets*, *le calcul de stratégie d'adaptation*, *l'adaptation individuelle des objets*, *la transformation de la scène* – sont indépendantes d'un point de vue conceptuel.

Ainsi, nous proposons un ensemble de modèles et d'architectures conceptuelles associées qui convergent vers le but fixé par ce travail de thèse : explorer la dimension sémantique des données 3D afin de simplifier les techniques de réutilisation et rendre plus adéquates les adaptations portant sur une scène 3D.

III Perspectives

Les bases posées par le modèle sémantique et les plates-formes construites autour de ce modèle nous laissent envisager des perspectives intéressantes pour la suite de nos travaux. Nous classons ces futurs travaux en plusieurs catégories concernant respectivement : l'implémentation, la sémantique, la réutilisation et l'adaptation.

III.1. Axe implémentation – futures évolutions

Nous envisageons de continuer les efforts d'implémentation autour de notre solution d'adaptation et de réutilisation. Pour les plates-formes que nous proposons au niveau conceptuel dans cette thèse, l'implémentation est en cours. L'application présentée dans le chapitre 9, utilise des versions monolithiques des plates-formes. Une première perspective est donc de modifier les implémentations existantes afin de leur assurer la flexibilité mise en avant par les architectures conceptuelles que nous proposons.

Nous souhaitons enrichir le modèle de descripteurs de 3DSEAM associé aux instances 3DSEAM représentées en MPEG-7, avec l'ensemble des descripteurs disponibles en MPEG-7. Ainsi, au moment de l'interrogation nous pouvons bénéficier de l'ensemble des critères portant sur la caractérisation des objets multimédia.

Actuellement nous n'avons pas pris en considération les problèmes liés au passage à l'échelle de nos entrepôts de description. L'utilisation d'index construits sur les critères géométriques, d'apparence, spatiaux ou sémantiques doit être considérée afin d'assurer un accès rapide aux informations. L'organisation distribuée des bases des connaissances est également à l'étude. La mise en place de ces transformations n'aura d'impact que sur la plate-forme 3DAF qui gère l'accès direct à l'information.

De même, nous envisageons également la construction au-dessus de 3DSDL et Adapt3D, d'une application qui facilite la génération de scènes adaptées, annotées sémantiquement. À présent, par souci de généricité les deux solutions 3DSDL et 3DAF sont indépendantes. La génération d'une scène adaptée est réalisée en deux temps : la génération de la scène à base de critères de réutilisation et l'adaptation de cette scène conformément aux règles d'adaptation. Cependant certaines de leurs fonctionnalités sont communes : la récupération de repères de localisation et de propriétés sémantiques, l'assemblage de scène, l'extraction de fragments, etc.

D'un point de vue applicatif, nous poursuivons le développement de l'application expérimentale consacrée à la gestion des modèles urbains pour valider l'ensemble des approches sur un cas d'étude spécifique à un domaine. L'étape suivante consistera à tester la robustesse de l'approche en termes de réutilisation et d'adaptation en cherchant à ré-exploiter les données 3D issues du milieu urbain dans le cadre d'un autre domaine applicatif, comme par exemple les systèmes visant la génération de visites virtuelles thématiques.

III.2. Axe sémantique – utilisation d'un raisonneur sémantique

Grâce au modèle de descripteurs, le modèle 3DSEAM est capable d'accommoder la variabilité en termes de propriétés utilisées pour décrire telle ou telle dimension du modèle. Ceci est un trait important de notre modèle car cela permet à chaque utilisateur qui alimente le système, de définir ses priorités en termes de descriptions. Selon ses intérêts, certaines propriétés sont plus adéquates que d'autres pour décrire une dimension. Le fait d'avoir inclus dans la définition d'un descripteur de propriété, l'ontologie de référence par rapport à laquelle il est défini, nous permet d'envisager de définir des synonymies entres descripteurs fournis par plusieurs acteurs. Nous considérons que ceci relève d'un domaine d'application spécifique où un expert du domaine a la responsabilité de caractériser les descripteurs.

Ainsi, nous envisageons de rattacher un raisonneur au module d'interprétation de requêtes de la plate-forme 3DAF. Ce raisonneur a pour rôle de trouver des règles d'équivalence entre les termes fournis par l'utilisateur et les règles associées aux fragments et aux entités de l'entrepôt d'annotations. Des mécanismes de réécriture de requêtes, tels que ceux présentés dans [Hammiche *et al.*, 2007], peuvent être mis en œuvre afin d'interroger convenablement les bases de connaissances de l'entrepôt 3DSEAM.

III.3. Axe réutilisation – approche déclarative pour l'assemblage de scène

En ce qui concerne, la réutilisation de scènes 3D nous souhaitons explorer l'utilisation de descriptions déclaratives de haut niveau pour indiquer la manière dont l'assemblage d'une scène est réalisé. Actuellement, la solution que nous proposons est de type déclaratif, cependant elle requiert des connaissances sur le langage X3D et l'identification précise de chaque objet dans la scène.

Nous considérons qu'il est souhaitable de proposer des moyens de descriptions de plus haut niveau où des relations qualitatives sont définies entre les objets composant la scène. L'intérêt de spécifier la disposition spatiale à l'aide de relations qualitatives est que cela rend le processus de génération sensible aux variations qu'il peut y avoir en termes de dimensions d'objets. Si dans un *patron de scène* des positions fixes sont retenues pour chaque objet, dans le cas où l'instance retrouvée est plus grande que celle prévue dans le *patron de scène*, des incohérences d'ordre spatial (par exemple, le chevauchement) peuvent apparaître dans la scène. Dans des approches qualitatives, telles que celles présentées dans [Larive *et al.,* 2004], la variabilité en termes de taille est prise en compte. L'utilisation d'une approche qualitative

suppose la réécriture des relations qualitatives en relations quantitatives au moment de la génération de la scène en fonction des propriétés spatiales des objets récupérés.

Un autre point que nous souhaitons résoudre avec ce type de formulation de règles d'assemblage est la variabilité en termes de nombre d'instances que l'on obtient pour une requête ouverte. Généralement, il est impossible de prévoir à l'avance le nombre exact d'objets se conformant aux critères d'une requête. Des constructions (métaphores de visualisation [Averbukh *et al.*, 2007]) permettant de faire face à cette variabilité doivent être intégrées dans la description de *patrons de scène*. Des pistes pour aborder ce type de solutions sensibles à la variabilité du nombre de résultats sont présentées dans [Walczak *et al.*, 2003].

III.4. Calcul de stratégies d'adaptation

Une dernière perspective d'étude que nous présentons ici, correspond à la quantification des différentes dimensions d'une scène 3D afin de mettre en place des stratégies d'adaptation qui modifient le moins possible la scène initiale. Ceci passe par une meilleure prise en compte du contexte de diffusion et par une évaluation quantitative de chaque dégradation subie par la scène par rapport aux différentes techniques d'adaptation disponibles.

Nous envisageons d'explorer une approche similaire à celle défendue par [Lemlouma, 2004]. Nous pensons construire un graphe dont les nœuds représentent la scène à différents états. Les arêtes correspondent aux techniques d'adaptation appliquées à la scène pour transiter d'un état à l'autre. Le calcul d'une stratégie se réduit au chemin le plus court entre le nœud source correspondant à l'état initial de la scène et le nœud destination correspondant à l'état souhaité, conforme au contexte de diffusion.

Afin que ce calcul soit en concordance avec les souhaits de l'utilisateur, la quantification de la dégradation apportée par chaque transformation doit être évaluée en fonction du profil utilisateur et de ses préférences.

Les données 3D sont de plus en plus présentes dans les systèmes d'information, notamment dans les SIG avec par exemple la visualisation de villes 3D. Si l'émergence du 3D entraîne une profusion d'informations, elle génère également de nouvelles possibilités d'applications.

Les propositions faites dans le cadre de cette thèse constituent une base de travail pour le développement d'applications 3D dotées de capacités d'adaptation et facilement réutilisables. Ceci va dans le sens d'une démocratisation des processus de création et de déploiement universel de données 3D.

Bibliographie

[3DIF, 2006]

3D INDUSTRY FORUM (3DIF). 2006. *Universal 3D File Format (Third Edition),* ECMA Standard 363, June 2006, disponible en ligne sur : http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-363.pdf.

[Abasolo *et al.*, 2006]

ABASOLO, M.J., Y PERALES, F.J. 2003. *Wavelet analysis for a new multiresolution model for large-scale textured terrains,* in Proc. of International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision - WSCG, disponible en ligne sur : http://wscg.zcu.cz/wscg2003/Papers_2003/K61.pdf.

[Airey *et al.*, 1990]

AIREY J.M., ROHLF J.H., BROOKS JR., FREDERICK.P.1990.*Towards image realism with interactive update rates in complex virtual building environment,* in ACM SIGGRAPH Special Issue on 1990 Symposium on Interactive 3D Graphics, Vol. 24 (2), pp. 41-50.

[Albertoni *et al.*, 2005]

ALBERTONI R., PAPALEO L., PITIKAKIS M., ROBBIANO F., SPAGNUOLO M., VASILAKIS G. 2005. *Ontology-based searching framework for digital shapes,* in OTM Workshops, LNCS 3762, pp. 896-905.

[Attene *et al.*, 2005]

ATTENE M., MOCCOZET L., HASSNER T., LEON J.C., SAYEGH R., TAL A., PAPALEO L., ROBBIANO F., GUTIERREZ M., ANDRESEN O., MARINI S., BIASOTTI S., CATALANO C. CHEUTET V., ALBERTONI R., BELAYEV A., HAMMANN S., ALLIEZ P., CIGNOGNI P., PITIKAKIS M. 2005. *Metadata for digital shape models – second version,* Delivrable D1.5 - IST Network of Excelence (NoE) n° 506766 AIM@SHAPE, July 2005, disponible en ligne sur : http://www.aimatshape.net/Groups/WP1/ ontologyDesign/d1-5_v2-0_final.doc.

[Autodesk, 2007]

AUTODESK LTD. 2007. *Autocad DXF Reference,* Autodesk Limited, disponible en ligne sur : http://images.autodesk.com/adsk/files/acad_dxf0.pdf.

[Averbukh *et al.*, 2007]

AVERBUKH V., BAKHTEREV M., BAYDALIN A., ISMAGILOV D., TRUSHENKOVA P. 2007. *Interface and Visualization Metaphors,* In Proc. of Human-Computer Interaction Conference, Beijing, pp. 13-22.

[Baader *et al.*, 2002]

BAADER F., CALVANESE D., MCGUINNESS D., NARDI D., PATEL-SCHNEIDER P. 2002. *The Description Logic Handbook Theory: Implementation and Applications*, Cambridge University Press.

[Backus, 1959]

BACKUS, J.W. 1959. *The syntax and semantics of the proposed International Algebraic Language of the Zürich ACM-GAMM Conference,* in Proc. of International Conference on Information Processing, UNESCO, pp.125-132.

[Baeza-Yates, 1999]

BAEZA-YATES B. 1999. *Modern Information Retrieval,* Addison Wesley, 1999.

[Barni *et al.*, 2001]

BARNI M.; BARTOLINI F.; PIVA A. 2001. *Improved wavelet-based watermarking through pixel-wise masking,* in IEEE Trans. on Image Processing, Vol. 10 (5), pp.783-791.

[Beck, 2005]

BECK H. W. 2005. *Lyra Ontology Management System,* University of Florida, http://orb.at.ufl.edu/ObjectEditor.

[Bekaert *et al.*, 2003]

BEKAERT, J. HOCHSTENBACH, P., VAN DE SOMPEL, H. 2003. *Using MPEG-21 DIDL to Represent Complex Digital Objects in the Los Alamos National Laboratory Digital Library,* in D-Lib Magazine, November 2003, Vol 9 (11), disponible en ligne sur : http://www.dlib.org/dlib/november03/bekaert/11bekaert.html.

[Benbernou et *al.*, 2005]

BENBERNOU S., HACID M.S., MAKHOUL A., MOSTEFAOUI A. 2005. *A spatio-temporal Adaptation model for multimedia presentations*, in Proc. of IEEE International Symposium on Multimedia, Irvine, California, pp. 143-150.

[Bentley, 1975]

BENTLEY J.L. 1975. *Multidimensional binary search trees used for associative searching,* in Communications of ACM, Vol. 18, pp. 509-517.

[Berners-Lee *et al.*, 2001]

BERNERS-LEE T., HENDLER J., LASSILA O. 2001. *The Semantic Web,* in Scientific American 284, pp. 34–43.

[Bilasco *et al.*, 2005a]

BILASCO I.M., GENSEL J., VILLANOVA-OLIVER M. 2005. *STAMP: a Model for Generating Adaptable Multimedia Presentations,* in Multimedia Tools and Applications Journal, Vol. 25 (3), pp. 361–375.

[Bilasco *et al.*, 2005b]

BILASCO I.M., GENSEL J., VILLANOVA-OLIVER M., MARTIN H. 2005. *On indexing of 3D scenes using MPEG-7,* in Proc. of ACM International Conference on Multimedia, Singapore, pp. 471-474.

[Bilasco *et al.*, 2005c]

BILASCO I.M., GENSEL J., VILLANOVA-OLIVER M., MARTIN H. 2005. *3DSEAM: a model for annotating 3D scenes using MPEG-7,* in Proc. of IEEE International Symposium on Multimedia, Irvine, California, pp. 310-319.

[Bilasco *et al.*, 2006a]

BILASCO I.M., GENSEL J., VILLANOVA-OLIVER M., MARTIN H. 2006. *Annotating 3D contents with MPEG-7 for reuse purposes*, in Proc. of IS&T/SPIE Symposium on Electronic Imaging, San Jose, California, pp. 60730Y.

[Bilasco *et al.*, 2006b]

BILASCO I.M., GENSEL J., VILLANOVA-OLIVER M., MARTIN H. 2006. *An MPEG-7 framework enhancing the reuse of 3D models,* in Proc. of Web3D Symposium, Columbia, Maryland, pp. 65-74.

[Bilasco *et al.*, 2006c]

BILASCO I.M., GENSEL J., VILLANOVA-OLIVER M., MARTIN H. 2006. *User Localization using a typology of 3D queries,* in Proc. of 1st International Workshop on Mobile Geospatial Augmented Reality, Banff, Alberta, Canada, disponible en ligne sur : http://regard.crg.ulaval.ca/proceedings/01-bilasco_et_al.pdf.

[Bilasco *et al.*, 2007a]

BILASCO I.M., GENSEL J., VILLANOVA-OLIVER M., MARTIN H. 2007. *Towards geospatial queries in semantic digital library for 3D data,* Trans. in GIS, Special Issue on The Geospatial Semantic Web, Vol. 11 (3), pp. 337-353.

[Bilasco *et al.*, 2007b]

BILASCO I.M., VILLANOVA-OLIVER M., GENSEL J., MARTIN H. 2007. *Semantique et modelisation de scenes 3D,* Revue ISI, Ed. Lavoisier, Vol. 12 (2), pp. 121-135.

[Bilasco *et al.*, 2007c]

BILASCO I.M., VILLANOVA-OLIVER M., GENSEL J., MARTIN H. 2007. *Semantic-based rules for 3D scene adaptation,* in Proc. of Web3D Symposium, Perugia, Italy, pp. 97-100.

[Boag *et al.*, 2007]

BOAG S., CHAMBERLIN D., FERNANDEZ M.F., FLORESCU D., ROBIE J., SIMEON J. 2007. *XQuery 1.0 : An XML Query Language,* W3C Recommendation, 23 January 2007, http://www.w3.org/TR/xquery.

[Bogdanovych *et al.*, 2005]

BOGDANOVYCH A., BERGER H., SIERRA C., SIMO S. 2005. *Narrowing the Gap between Humans and Agents in E-commerce: 3D Electronic Institutions,* in Proc. of 6th International Conference on Electronic Commerce and Web Technologies (EC-Web'05), LNCS 3590, Springer-Verlag, pp. 128-137.

[Boll *et al.*, 1999]

BOLL S., KLAS W. ET WANDEL J. 1999. *A Cross-Media Adaptation Strategy for Multimedia Presentations,* in Proc. of 7th ACM International Conference on Multimedia, Orlando, Florida, USA, pp. 37-46.

[Booch *et al.*, 1998]

BOOCH G., RUMBAUCH J, JACOBSON I. 1998. *The Unified Modelling Language User Guide,* Addison Wesley

[Bordersen, 2005]

BORDERSEN A. 2005. *Real-time visualization of large textured terrains,* in Proc. of 3rd International Conference on Computer Graphics and Interactive Techniques, Dunedin, New Zeeland, pp. 439-442.

[Bormans *et al.*, 2003]

BORMANS J., GELISSEN J., PERKIS A. *MPEG-21: The 21st century multimedia framework,* in IEEE Signal Processing Magazine, Vol. 20 (2), pp. 53-62.

[Bosca *et al.*, 2007]

BOSCA A., BONINO D., CORNO F., COMERIO M., GREGA S. 2007. *A reusable 3D visualisation component for the Semantic Web,* in Proc. of 12th International Conference on 3D Web Technology, Perugia, Italy, pp. 89-96.

[Brooks *et al.*, 1986]

BROOKS JR., FREDERICK P. 1986. *Walkthrough - A Dynamic Graphics System for SimulatingVirtual Buildings,* in Proc. of Workshop on Interactive 3D Graphics, Chapel Hill, North Carolina, pp. 9-21.

[Bucur *et al.*, 2005]

BUCUR O., BEAUNE P., BOISSIER O. 2005. *Définition et représentation du contexte pour des agents sensibles au contexte,* Actes de la 2ème Conférence UbiMob'05, Grenoble, France, pp. 13-16.

[Bulterman, 1998]

BULTERMAN D. C. A. 1998. *User-Centered Abstractions for Adaptive Hypermedia Presentations, CWI (Centrum voor Wiskunde en Iformatica),* in Proc. of ACM International Conference of Multimedia, Bristol, UK, pp. 247-256.

[Bulterman, 2005]

BULTERMAN D. C. A. 2005. *Synchronized Multimedia Integration Language (SMIL 2.1),* W3C Recommandation, 13 decembre 2005, disponible en ligne sur : http://www.w3.org/TR/SMIL/.

[Burnett *et al.*, 2003]

BURNETT, I., VAN DE WALLE, R., HILL, K., BORMANS, J., PEREIRA, F. 2003. *MPEG-21: Goals and Achievements,* in IEEE Multimedia, Vol. 10 (4), pp. 60-70.

[Burnett *et al.*, 2005]

BURNETT I., DAVIS S., DRURY G. 2005. *MPEG-21Digital Item Declaration and Identification – Principles and Compression,* in IEEE Trans. on Multimedia, Vol. 7 (3), pp. 400-407.

[Cai *et al.*, 2007]

CAI S., QI Y., SHEN X. 2007. *3D Data Codec and Transmission over the Internet,* in Proc. of Web3D Symposium, Perugia, Italy, pp. 53-56.

[Caligari, 2002]

CALIGARI CORPORATION. 2002. *Caligari trueSpace file format*, Caligari Corporation, disponible en ligne sur : http://www.caligari.com/download/zip/scncob65.zip (checked on 8th February 2006).

[Carroll *et al.*, 2004]

CARROLL J. J., DICKINSON I., DOLLIN C., REYNOLDS D., SEABORNE A., WILKINSON K. 2004. *Jena: Implementing the semantic web recommendations,* in Proc. of 13th International World Wide Web Conference (WWW 2004), New York, USA, pp. 74-83.

[Carson *et al.*, 1999]

CARSON C., THOMAS M., BELONGIE S., HELLERSTEIN J. M., MALIK J. 1999. *Blobworld: A System for Region-Based Image Indexing and Retrieval,* in Proc. of 3rd Conference on Visual Information and Information Systems, pp. 509-516.

[Cattel, 1994]

CATTELL R. 1994. *The Object Database Standard: ODMG-93*. Morgan Kaufmann, San Francisco, CA.

[Celakovski *et al.*, 2005]

CELAKOVSKI S., PREDA M., KALAJDZISKI S., DAVCEV D., PRETEUX F. 2005. *Rendering strategies for displaying MPEG-4 3D graphics objects,* WSEAS Trans. on Communications, New Orleans, Louisiana, Vol. 7 (4), pp. 1851-1858.

[Celentano *et al.*, 2004]

CELENTANO A., NODARI M., PITTARELLO F. 2004. *Adaptive interaction in Web3D virtual worlds,* in Proc. of 9th International Conference on 3D Web technology, Monterey, California, pp. 41-50.

[Certain *et al.*, 1996]

CERTAIN A., POPOVIC J., DEROSE T., DUCHAMP T., SALESIN D., STUETZLE W. 1996. *Interactive multiresolution surface viewing,* in Proc. of 23rd Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH '96, pp. 91-98.

[Chaari *et al.*, 2004]

CHAARI T., LAFOREST F., CELENTANO A. 2004. *Design of context-aware applications based on web services,* Technical Report RR-2004-033, LIRIS, Lyon, octobre 2004, disponible en ligne sur : http://liris.cnrs.fr/publis/rr_html.

[Chandra *et al.*, 2001]

CHANDRA S., GEHANI A., ELLIS C. S., VAHDAT A. 2001. *Transcoding Characteristics of Web Images*, in Proc. of Multimedia Computing and Networking 2001, SPIE 4312, Adelaide, Australia, pp. 135–149.

[Chang, 2002]

CHANG S.F. 2002. *The holy grail of content-based media analysis*, in IEEE Multimedia, vol 9(2), Apr-Jun 2002, pp. 6-10.

[Chen *et al.*, 2003]

CHEN L.Q., XIE X., FAN X., MA W.Y., ZHANG H.J., ZHOU H.Q. 2003. *A visual attention model for adapting images on small displays,* in Multimedia Systems Journal, Springer Berlin, Vol. 9(4), pp. 353-364.

[Chittaro *et al.*, 2002]

CHITTARO L., RANON R. 2002. *Dynamic generation of personalized VRML content: a general approach and its application to 3D e-commerce,* in Proceeding of the Seventh international Conference on 3D Web Technology, Tempe, Arizona, USA, February 24 - 28, pp. 145-154.

[Chittaro *et al.*, 2003]

CHITTARO L., RANON R., IERONUTTI L. 2003. *Guiding visitors of Web3D worlds through Automatically Generated Tours,* in Proceeding of the 8th International Conference on 3D Web Technology, Saint Malo, France, pp. 27-38.

[Chittaro *et al.*, 2004]

CHITTARO L., RANON R. 2004.*Using the X3D language for adaptive manipulation of 3D Web content,* in LNCS 3137, pp. 287-290.

[Chrisment *et al.*, 2000]

CHRISMENT C., SEDES F. 2000. *Le document multimedia,* Actes du Congrès CIFED '2000 : colloque international francophone sur l'écrit et le document, Lyon, France, pp. 10-20.

[Chrisment *et al.*, 2002]

CHRISMENT C., SEDES F. *Annotations de média : Vers une représentation multidimensionnelle,* Revue ISI, Ed. Lavoisier, Vol. 7 (5-6), pp. 45-65.

[Christopoulos *et al.*, 2000]

CHRISTOPOULOS C., SKODRAS A., EBRAHIMI T. 2000.*The JPEG2000 Still Image Coding System: An Overview,* in IEEE Trans. on Consumer Electronics, Vol. 46. pp. 1103–1127.

[Cimprich *et al.*, 2007]

CIMPRICH P., BECKER O., NENTWICH C, JIROUSEK H., BATSIS M, BROWN P., KAY M. 2007. *Streaming Transformations for XML (STX) Version 1.0,* disponible en ligne sur : http://stx.sourceforge.net/documents/.

[Clark, 1999]

CLARK J. 1999. *XSL Transformation (XSLT) Version 1.0,* W3C Recommandation 16 novembre 199, disponible en ligne sur : http://www.w3.org/TR/xslt.

[Clark *et al.*, 1999]

CLARK J., DEROSE S. 1999. *XML Path Language (XPath) Version 1.0,* W3C Recommandation 16 novembre 1999, disponible en ligne sur : http://www.w3.org/TR/xpath.

[Clementini *et al.*, 1997]

CLEMENTINI E., DI FELICE P. 1997. *Approximate topological relations,* in International Journal of Approximate Reasoning, Vol. 16 (2), pp. 173-204.

[Coehlo *et al.*, 2004]

COELHO E.M., MACINTYRE B., JULIER S. 2004. *OSGAR: A scene graph with uncertain transformations,* In Proc. of International Symposium on Mixed and Augmented Reality (ISMAR'04), Arlington, USA, pp. 6-15.

[Cohen *et al.*, 1997]

COHEN J., MANOCHA D., OLANO M. 1997. *Simplifying polygonal models using successive mappings,* in Proc. of IEEE Visualization'97, pp. 395–402.

[Cohen *et al.*, 1993]

COHEN L., COHEN I. 1993. *Finite Element Methods for Active Contour Models and Balloons for 2D and 3D images,* In IEEE Transaction Pattern Analysis and Machine Intelligence, Vol. 15 (11) pp. 1131-1147.

[Cohen-Or *et al.*, 2003]

COHEN-OR D., CHRYSATHOU Y., SILVA C.T., DURAND F. 2003. *A survey of visibility for walkthrough applications,* in Transaction of IEEE on Visualization and Computing Graphics, Vol. 9 (3), pp. 412-431.

[Comaniciu *et al.*, 2002]

COMANICIU D., MEER P. 2002. *Mean shift: A robust approach toward feature space analysis,* in IEEE Transaction. Pattern Analysis and Machine Intelligence, Vol. 24 (5), pp. 603–619.

[Cruz *et al.*, 2005]

CRUZ C., BOOCHS F., NICOLLE C. 2005. *3D Reconstruction based on Semantic Information for Architectural Applications,* in Schriftenreihe Informations- und Messtechnick, Shaker Verlag, Vol. 6, pp. 67-82.

[Cvetkovic *et al.*, 1997]

CVETKOVIC D., ROWLINSON P., SIMIC S. 1997. *Eigenspaces of Graphs,* Cambridge University Press, United Kingdom.

[Dachselt *et al.*, 2002]

DACHSELT R., HINZ M., MEIßNER K. 2002. *Contigra: an XML-based architecture for component-oriented 3D applications,* in Proc. of 7th International Conference on 3D Web Technology, Temple, Arizona, pp. 155-163.

[Dachselt *et al.*, 2003]

DACHSTEL R., DÖRNER R., GRIMM P. 2003. *Adopting and augmenting X3D for efficient 3D content production: concepts and tools,* in Proc. of 8th International Conference on 3D Web Technology, Saint Malo, France, pp. 184-185.

[Dachselt *et al.*, 2006]

DACHSTEL R., HINZ M., PIETSCHMANN S. 2006. *Using the AMACONT architecture for flexible adaptation of 3D web applications,* in Proc. of 11th International Conference on 3D Web Technology, Columbia, Maryland, pp. 75-84.

[Danovaro *et al.*, 2007]

DANOVARO E., PAPALEO L., ATTENE M., SALEEM W. 2007. *Advanced remote inspection and download of 3D shapes,* in Proc. of 12th International Conference on 3D Web Techonology, Perugia, Italy, pp. 57-60.

[De Bra *et al.*, 2003]

DE BRA P., AERTS A., BERDEN B., DE LANGE B., ROUSSEAU B., SANTIC T., SMITS D., STASH N. 2003. *AHA! The Adaptive Hypermedia Architecture,* in Proc. of ACM Hypertext Conference, Nottingham, U.K., pp. 81-84.

[De Keukelaere *et al.*, 2005]

DE KEUKELAERE F., DE ZUTTER S., VAN DE WALLE R. 2005. *MPEG-21 Digital Item Processing,* in IEEE Trans. on Multimedia, Vol. 7 (3), pp. 427-434.

[DeHaan *et al.*, 2003]

DEHAAN D., TOMAN D., CONSENS M., OZSU M. T. 2003. *A comprehensive XQuery to SQL translation using dynamic interval encoding,* in Proc. of ACM SIGMOD, San Diego, California, pp. 623–634.

[Del Bimbo *et al.*, 1998]

DEL BIMBO A. D., VICARIO E. 1998. *Weighting Spatial Relationships in Retrieval by Visual Contents,* in Proc. of IFIP TC2/WG 2.6 4th Working Conference on Visual Database Systems, L'Aquila, Italy, pp. 277-292.

[Devillers *et al.*, 2005]

DEVILLERS S., TIMMERER C., HEUER J., HELLWAGNER H. 2005. *Bitstream syntax description-based adaptation in streaming and constrained environments,* in IEEE Trans. on Multimedia, Vol. 7 (3), pp. 463-470.

[Dey *et al.*, 2001]

DEY A. K. 2001. *Understanding and Using Context,* in Personal and Ubiquitous Computing Journal, Springer-Verlag, Vol. 5 (1), pp. 4-7.

[Di Giacomo *et al.*, 2004]

DI GIACOMO T., KIM H., GARCHERY S., MAGNENAT-THALMANN N., CAILLIERE D., BELAY G., COTARMANACH A., RIEGEL T. 2004. *Benchmark-Driven Automatic Transmoding of 3D to 2D Talking Heads,* in Workshop on Modelling and Motion Capture Techniques for Virtual Environments, disponible en ligne sur : http://www.miralab.unige.ch/papers/328.pdf.

[Dieberger *et al.*, 1998]

DIEBERGER A., FRANK A.U. 1998. *A City Metaphor for Supporting Navigation in Complex Information Spaces,* in Journal of Visual Languages and Computing, Vol. 9 (6), pp. 597-622.

[Divarkan, 2004]

DIVAKARAN A. 2004. *Management of Multimedia Semantics Using MPEG-7,* Technical Report TR2004-116, Mitsubishi Electric Research Laboratories, Octobre 2004, disponible en ligne sur : http://www.merl.com/reports/docs/TR2004-116.pdf.

[Döllner *et al.*, 2002]

DÖLLNER J., HINRICHS K. 2002. *A generic rendering system.* in IEEE Trans. on Visualization and Computer Graphics, Vol. 8 (2), pp. 99–118.

[Duecker *et al.*, 1997]

DUECKER M., GEIGER C., HUNSTOCK R., LEHRENFELD G., MUELLER W. 1997. *Visual-textual prototyping of 4D scenes,* in Proc. of IEEE Symposium on Visual Languages, Isle of Capri, Italy, pp. 328-335.

[Egenhofer *et al.*, 1990]

EGENHOFER, M.J., HERRING, J.R., 1990. *A mathematical framework for the definition of topological relationships,* in Proc. of 4th International Symposium on Spatial Data Handling, Zurich, Switzerland, pp. 803–813.

[Egenhofer *et al.*, 1995]

EGENHOFER M.J., FRANZOSA R.D. 1995. *On the equivalence of Topological Relations,* in International Journal of Geographical Information Systems, Vol. 9 n° 2, pp. 133-152.

[Elad *et al.*, 2001]

ELAD M., TAL A., ANDAR S. 2001. *Content based retrieval of VRML objects - an iterative and interactive approach,* in Proc. of 6th Eurographics Workshop on Multimedia, Manchester, UK, pp. 107–118.

[Estalayo *et al.*, 2004]

ESTALAYO E., SALGADO L., MORAN F., CABRERA J. 2004. *Adapting Multimedia Information Associationin VRML Scenes for E-Learning Applications,* in Proc. of 1st International Workshop LET-Web3D, Monterey, California, pp. 16-22.

[Fallside *et al.*, 2004]

FALLSIDE D.C., WALMSLEY P. 2004. *XML Schema Part 0: Primer Second Edition,* W3C Recommendation 28 October 2004, disponible en ligne sur : http://www.w3.org/TR/xmlschema-0/.

[Fatemi *et al.*, 2003]

FATEMI N., KHALED O. A. AND CORAY G. 2003. *An XQuery adaptation for MPEG-7 documents retrieval.* in Proc. of XML Conference and Exposition, Philadelphia, disponible en ligne sur : http://www.idealliance.org/papers/dx_xml03/papers/05-03-01/05-03-01.pdf.

[Fensel *et al.*, 2001]

FENSEL D., VAN HARMELEN F., HORROCKS I., MCGUINNESS D.L., PATEL-SCHNEIDER P.F. 2001. *OIL: An Ontology Infrastructure for the Semantic Web,* in IEEE Intelligent Systems, Vol. 16 (2), pp. 38-45.

[Foley *et al.*, 1998]

FOLEY P.G., MAMAGHANI F., BIRKEL P.A. 1998. *The Synthetic Environment Data Representation and Interchange Specification (SEDRIS) Development Project*, disponible en ligne sur : http://sedris.org/download/documentation/it103a60.pdf.

[Fonseca *et al.*, 2003]

FONSECA M.J., JORGE J.A. 2003. *Indexing high-dimensional data for content-based retrieval in large databases,* in Proc. of 8th International Conference on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan, pp. 267-274.

[Fonseca *et al.*, 2004]

FONSECA M.J., FERREIRA A, JORGE J.A. 2004. *Towards 3D modeling using sketches and retrieval,* in Proc. of 1st Eurographics Workshop on Sketch–Based Interfaces and Modeling, Grenoble, France, disponible en ligne sur : http://sketch.inesc.pt/sbm04/papers/14.pdf.

[Fox *et al.*, 1998]

FOX A., GRIBBLE S., BREWER E. A., AMIR E. 1996. *Adapting to Network and Client Variability via On-Demand Dynamic Distillation,* in Proc. of 7th Int.. Conf. on Architectural Support for Programming Languages and Operating Systems, Cambridge, USA, pp. 160–170.

[Funkhouser *et al.*, 1992]

FUNKHOUSER T.A., SÉQUIN C. H., TELLER J.S. 1992. *Management of large amounts of data in interactive building walkthroughs,* in Proc. of Symposium on Interactive 3D Graphics, Cambridge, Massachusetts, pp. 11-20.

[Funkhouser *et al.*, 1993]

FUNKHOUSER T.A., SEQUIN C.H. 1993. *Adaptative display algorithm for interactive frame rates during visualization of complex virtual environments,* in Proc. of 20th International Conference on Computer Graphics and Interactive Techniques, pp. 247-254

[Funkhouser *et al.*, 2003]

FUNKHOUSER T.A., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D. 2003. *A search engine for 3D models,* in ACM Trans. of Graphics (TOG), Vol. 22 n° 1, January 2003, pp. 83-105.

[Funkhouser *et al.*, 2004]

FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D. 2004. *Modeling by example,* in ACM SIGGRAPH 2004 Papers (Los Angeles, California, August 08 - 12, pp. 652-663.

[Goldman *et al.*, 2005]

GOLDMAN O., LENKOV D. 2005. *XML Binary Characterisation,* W3C Working Group Note 31 March 2005, disponible en ligne sur : http://www.w3.org/TR/xbc-characterization/.

[Graves *et al.*, 2002]

GRAVES A, LALMAS M. 2002. *Video retrieval using MPEG-7 based inference network,* in Proc. of ACM SIGIR'02, Tampere, Finland, pp. 339-346.

[Gruber, 1995]

GRUBER T.R., 1995. *Toward principles for the design of ontologies used for knowledge sharing,* in International Journal of Human-Computer Studies, Academic Press, Vol. 43 (5-6), pp. 907-928.

[Gudivada *et al.*, 1995]

GUDIVADA V.N., RAGHAVAN V.V. 1995. *Content-Based Image Retrieval Systems,* in IEEE Computer, Vol. 28 (9), pp. 18-22.

[Gutierrez *et al.*, 2005]

GUTIERREZ M., VEXO F., THALMANN D. 2005. *Semantic-based representation of virtual environments,* in Intl. Journal of Computer Applications in Technology, Vol. 23 (2-4), pp. 229-238.

[Haase *et al.*, 2004]

HAASE P., BROEKSTRA J., EBERHART A., VOLZ R. 2004. *A comparison of RDF query languages,* in Proc. of Third International Semantic Web Conference, Hiroshima, Japan, pp. 502-517.

[Halabala, 2003]

HALABALA, P. 2003. *Semantic Metadata Creation,* in Proc. of 7th Central European Seminar on Computer Graphics (CESCG'03), pp. 15-25.

[Hamilton, 1997]

HAMILTON, G. 1997. *JavaBeans API Specification, v 1.01,* Sun Microsystems.

[Hammiche *et al.*, 2007]

HAMMICHE S., LOPEZ B., BENBERNOU S., HACID M.S, VAKALI A. 2007. *Domain knowledge based queries for multimedia data retrieval,* in Journal of Digital Information Management, Vol. 5 (2), pp. 75-82.

[Handschuh *et al.*, 2002]

HANDSCHUH S., STAAB S., CIRAVEGNA F. 2002. *S-CREAM -- Semi-automatic CREAtion of Metadata,* in LNCS 2473, pp. 165-184.

[Hardman *et al.*, 1993]

HARDMAN L, VAN ROSSUM G., BULTERMAN D.C.A. 1993. *Structured Multimedia Authoring,* in Proc. of ACM International Conference on Multimedia, Anaheim, California, pp. 283-289.

[Hasting *et al.*, 1994]

HASTING A., FERGUSON S. 1994. *Lightwave 3D Object File Format*, disponible en ligne sur : http://www.dcs.ed.ac.uk/home/mxr/gfx/3d/LWOB.txt.

[Heipke, 2005]

HEIPKE. 2005. *Why Extracting Feature Is Hard, EOM – AURORA, Vol. 14 (1), pp. 31-31.*

[Hendler *et al.*, 2000]

HENDLER J, MCGUINNES D.L. 2000. *DARPA Agent Markup Language,* in IEEE Intelligent Systems, Vol. 16 (6), pp. 67-73.

[Hernandez *et al.*, 2007]

HERNANDEZ N., MOTHE J., CHRISMENT C., EGRET D. 2007. *Modeling context through domain ontologies,* in Journal of Information Retrieval, Special Issue on Contextual Information Retrieval Systems, Vol. 10 (2), pp. 143-172.

[Hetherington *et al.*, 2004]

HETHERINGTON R.E., SCOTT J.P. 2004. *Adding a fourth dimension to three dimensional virtual spaces,* in Proc. of 9th International Conference on 3D Web Technology, Monterey, California, pp. 163-172.

[Hilaga *et al.*, 2001]

HILAGA M., SHINAGAWA Y., KOHMURA T., KUNII T.L. 2001.*Topology matching for fully automatic similarity estimation of 3D shapes,* in Proc. of 28th International Conference on Computer Graphics and Interactive Techniques, pp. 203-212.

[Hinz *et al.*, 2004]

HINZ M., FIALA Z. 2004. *AMACONT: A System Architecture for Adaptative Multimedia Web Applications,* in Berliner XML Tage, pp. 65-74.

[Hobson *et al.*, 2006]

HOBSON P, KOMPATSIARIS, Y. 2006. *Advances in semantic multimedia analysis for personalised content access*, in Proc. of IEEE Symposium on Circuits and Systems '06 (ISCAS06), Kos, Greece, pp 2085-2088.

[Hoi *et al.*, 2003]

HOI K. K., LEE D. L. ET XU J. 2003. *Document Visualization on Small Displays,* in Proc. of Mobile Data Management, Melbourne, Australia, pp. 262-278.

[Hoppe, 1996]

HOPPE H. 1996. *Progressive meshes,* in Proc. of 23rd ACM International Conference on Computer Graphics and Interactive Techniques, pp. 99-108.

[Horn, 1984]

HORN B. 1984. *Extended gaussian images,* in Proc. of IEEE 72, Vol. 12, pp. 1671–1686.

[Hosseini *et al.*, 2001]

HOSSEINI M., GEORGANAS N.D. 2001. *Suitability of MPEG4's BIFS for Development of Collaborative Virtual Environments,* in Proc. of 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 299-304.

[Hough, 1962]

HOUGH P.V.C. 1962. *Method and means for recognizing complex patterns,* U.S. Patent 3.069.654.

[Hutt, 2005]

HUTT K. 2005. *A Comparison of RDF Query Languages,* in Proc. of 21th Computer Science Seminar, Hartfort, Connecticut, pp. 1-7.

[Inatsuka *et al.*, 2005]

INATSUKA H., UCHINO M., OKUDA M. 2005. *Level of detail control for texture on 3D maps*, In Proc. of 11th International Conference on Parellel and Distributed Systems, Minneapolis, Minnesota, Vol. 2, pp. 206-209.

[Johnson *et al.*, 1999]

JOHNSON A.E., HEBERT M. 1999. *Using spin images for efficient object recognition in cluttered 3D scenes,* IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 21 (5), pp.433-449.

[Karvounarakis *et al.*, 2002]

KARVOUNARAKIS G., ALEXAKI S., CHRISTOPHIDES V., PLEXOUSAKIS D., SCHOLL M. 2002. *RQL: a Declarative Query Language for RDF,* in Proc. of International Conference on World Wide Web, Honolulu, Hawai, pp. 592-603.

[Katz, 2004]

KATZ H. 2004. *XsRQL: an XQuery-style Query Language for RDF,* Submission to the RDF Data Access Working Group (DAWG) June 27, 2004, disponible en ligne sur : http://www.fatdog.com/xsrql.html.

[Kazhdan *et al.*, 2003]

KAZHDAN M., FUNKHOUSER T., RUSINKIEWICZ S. 2003. *Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors,* in Symposium on Geometry Processing, pp. 156-164

[Kim *et al.*, 2004]

Kim D.H., Yun I.D., Lee S.U. 2004. *A comparative study on attributed relational graph matching algorithms for perceptual 3-D shape descriptor in MPEG-7*, in Proc. of 12th Annual ACM International Conference on Multimedia, New York, USA, pp. 700-707.

[Kim *et al.*, 2006]

KIM H., JOSLIN C., DI GIACOMO T., GARCHERY S., MAGNENAT-THALMANN N. 2006. *Device-based Decision-making for Adaptation of Three-Dimensional Content,* in The Visual Computer, Springer, Vol. 22 (5), pp. 332-345.

[Kirsch-Pinheiro *et al.*, 2005]

KIRSCH-PINHEIRO M., VILLANOVA-OLIVER M., GENSEL J., MARTIN H. 2005. *Une formalisation du contexte dans les environnements coopératifs nomades,* Actes de la 2ème Conférence UbiMob'05, Grenoble, France, pp. 1-8.

[Koenen, 1999]

KOENEN R. 1999. *MPEG-4 multimedia for our time,* in IEEE Spectrum, Vol. 36 (2), pp.26-33.

[Kleinermann *et al.*, 2005]

KLEINERMANN F., TROYER O. DE, MANSOURI H., ROMERO R., PELLENS B., BILLE W. 2005. *Designing semantic virtual reality applications,* in Proc. of 2nd INTUITION International Workshop, Senlis, France, disponible en ligne sur : http://wise.vub.ac.be/members/frederic/Papers/paper2005_6.pdf.

[Klyne *et al.*, 2004]

KLYNE G., REYNOLDS F., WOODROW C., OHTO H., HJELM J., BUTLER M.H., TRAN L. 2004. *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0,* W3C Recommandation, disponible en ligne sur : http://www.w3.org/TR/CCPP-struct-vocab/

[Koenderink, 1990]

KOENDERINK J. 1990. *Solid Shape,* MIT Press, Cambridge, Massachusetts.

[Koubaroulis, 2001]

KOUBAROULIS D. 2001. *The Multimodal Neighbourhood Signature for modelling object colour appearance and applications in computer vision,* Thèse de doctorat, University of Surrey.

[Lafruit *et al.*, 2000]

LAFRUIT, G.; NACHTERGAELE, L.; DENOLF, K.; BORMANS, J. 2000. *3D computational graceful degradation,* in Proc. of IEEE International Symposium on Circuits and Systems, Vol. 3, pp.547-550.

[Lamberti *et al.*, 2003]

LAMBERTI F., ZUNINO C., SANNA A., FIUME A., MANIEZZO M. 2003. *An accelerated remote graphics architecture for PDAS,* in Proceeding of the 8th International Conference on 3D Web Technology, St. Malo, France, pp. 55-66.

[Lanthier *et al.*, 1997]

LANTHIER M., MAHESHWARI A., SACK J. 1997. *Approximating Weighted Shortest Paths on Polyhedral Surfaces,* in Proc. of 13th Symposium on Computation Geometry, Nice, France, pp 274-283.

[Larive *et al.*, 2004]

LARIVE M., LE ROUX O., GAILDRAT V. 2004. *Using Meta-Heuristics for Constraint-Based 3D Objects Layout,* in Proc. of 7th International Conference on Computer Graphics and Artificial Intelligence, (3IA'04), Limoges, France, 11-23.

[Lauf *et al.*, 2005]

LAUF S., BURNETT I. 2005. *Implementation of a mobile MPEG-21 peer,* in Proc. of 13th annual ACM International Conference on Multimedia 2005, Singapore, pp. 323-326.

[Lee *et al.*, 2001]

LEE K., CHANG H. S., CHUN S. S., CHOI L. AND SULL S. 2001. *Perception-based Image Transcoding for Universal Multimedia Access.* in Proc. of IEEE International Conference on Image Processing, pp. 475–478.

[Lemlouma, 2004]

LEMLOUMA T. 2004. *Architecture de négociation et d'adaptation de Services Multimedia dans des Environnements Hétérogènes,* Thèse de Doctorat, Institut National Polytechnique de Grenoble.

[Levoy *et al.*, 1990]

LEVOY M. 1990. *Efficient ray tracing of volume data,* ACM Trans. on Graphics, Vol. 9 (3), pp. 245-261.

[Lewis *et al.*, 1992]

LEWIS A.S.; KNOWLES G. 1992. *Image compression using the 2-D wavelet transform,* in IEEE Trans. on Image Processing, Vol.1 (2), pp. 244-250.

[Libsie *et al.*, 2002]

LIBSIE M., KOSCH H. 2002. *Content adaptation of multimedia delivery and indexing using MPEG-7,* in Proc. of 10th ACM International Conference on Multimedia, Juan-les-Pins, France, pp. 644-646.

[Little *et al.*, 2002]

LITTLE S., GEURTS J., HUNTER J. 2002. *Dynamic Generation of Intelligent Multimedia Presentation through Semantic Inferencing,* in LNCS 2458, pp. 158 – 175.

[Liu *et al.*, 2002]

LIU P., SUHSU L. 2002. *Queries of digital content description in MPEG-7 and MPEG-21 XML documents,* in Proc. of XML Europe 2002, Barcelona, Spain, disponible en ligne sur : http://www.idealliance.org/papers/xmle02/dx_xmle02/papers/03-02-01/03-02-01.pdf

[Lorenz *et al.*, 2006]

LORENZ B., OHLBACH H.J. AND STOFFEL E.P. 2006, *A Hybrid Spatial Model for Representing Indoor Environments*, in Proc. of W2GIS, (LNCS 4295), Hong Kong, pp. 102-112.

[MacVittie, 2006]

MACVITTIE L.A. 2006. *XAML in a Nutshell,* O'Reilly Media, Inc, March 27.

[Mahmoudi *et al.*, 2002]

MAHMOUDI S., DAOUDI M. 2002. *3D models retrieval by using characteristic views,* in Proc. of 16th International Conference on Pattern Recognition, Vol. 2, pp. 457-460.

[Manola *et al.*, 2004]

MANOLA F., MILLER E. 2004. *RDF Primer,* W3C Recommendation 10 February 2004, disponible en ligne sur : http://www.w3.org/TR/rdf-primer/.

[Mansouri, 2005]

MANSOURI H. 2005. *Using Semantic Descriptions for Building and Querying Virtual Environments,* Rapport de master, Vrije Universiteit Brussel, disponible en ligne sur : http://wise.vub.ac.be/downloads/theses/mansourih-thesis.pdf

[Martinez, 2002]

MARTINEZ J.M. 2002. *MPEG-7: the generic multimedia content description standard, part 2,* in IEEE Multimedia, Vol. 9 (3), pp.83-93.

[Martinez *et al.*, 2002]

MARTINEZ J.M., KOENEN R., PEREIRA F. 2002. *MPEG-7: the generic multimedia content description standard, part 1,* in IEEE Multimedia, Vol. 9 (2), pp.78-87.

[Marvie *et al.*, 2004]

MARVIE J.E., BOUATOUCH K. 2004. *A VRML-X3D extension for massive scenery management in virtual worlds,* in Proc. of 9th International Conference on 3D Web Technology, Monterey, California, pp. 145-153.

[McGuinness *et al.*, 2004]

MCGUINNESS D.L., HARMELEN VAN F. 2004. *OWL Web Ontology Language Overview,* W3C Recommandation 10 February 2004, disponible en ligne sur : http://www.w3.org/TR/owl-features/.

[Miller, 2001]

MILLER L. 2001. *RDF Squish query language and Java implementation,* Institute for Learning and Research Technology, University of Bristol, disponible en ligne sur : http://ilrt.org/discovery/2001/02/squish/.

[Min, 2004]

MIN P. 2004. *A 3D Model Search Engine,* Thèse de doctorat, Department of Computer Science, Princeton University.

[Mohan *et al.*, 1999]

MOHAN R., SMITH J.R., CHUNG-SHENG L. 1999. *Adapting multimedia Internet content for universal access,* in IEEE Trans. on Multimedia, Vol.1 (1), pp.104-114.

[Mokhtarian *et al.*, 2000]

MOKHTARIAN F., ABBASI S. 2000. *Automatic view selection in multi-view object recognition,* in Proc. of 15th International Conference on Pattern Recognition (ICPR'00), Barcelona, Spain, Vol. 1, pp. 13-16.

[Mokhtarian *et al.*, 2003]

MOKHTARIAN F., BOBER M. 2003. *Curvature Scale Space representation: Theory, Applications and MPEG-7 standardization (Computational Imaging and Vision, 25)*, Kluwer Academic Publishers.

[Mostéfaoui *et al.*, 2004]

MOSTEFAOUI K., PASQUIER-ROCHA J., BREZILLON, P. 2004. *Context-aware computing: a guide for the pervasive computing community,* In Proc. of IEEE/ACS International Conference on Pervasive Services (IPCS'04), IEEE Computer Society, pp. 39-48.

[Mulloni *et al.*, 2007]

MULLONI A., NADALUTTI D., AND CHITTARO L. 2007. *Interactive walkthrough of large 3D models of buildings on mobile devices,* in Proc. of Twelfth international Conference on 3D Web Technology, Perugia, Italy, pp. 17-25.

[Naef *et al.*, 2003]

NAEF M., LAMBORAY E., STAADT O, GROSS M. 2003. *The blue-c distributed scene graph,* in ACM Workshop on Virtual Environments, Zurich, Switzerland, pp. 125-133.

[Ohbuci *et al.*, 2003]

OHBUCHI R., NAKAZAWA M., TAKEI T. 2003. *Retrieving 3D shapes based on their appearance,* in Proc. of 5th ACM SIGMM International Workshop on Multimedia Information Retrieval, Berkeley, California, pp. 39-45.

[Oliverio *et al.*, 2007]

OLIVERIO J., MASAKOWSKI Y. R., BECK H., APPUSWAMY R. 2007. *ISAS : A human-centric digital media interface to empower real-time decision-making accross distributed systems,* in Proc. of 12th International Conference on 3D Web technology, Perugia, Italy, pp. 81-87.

[Olson *et al.*, 2004]

OLSON M., OGBUJI U. 2004. *Versa,* Fourthought project, disponible en ligne sur : http://uche.ogbuji.net/tech/rdf/versa/etc/versa-1.0.xml.

[Osada *et al.*, 2002]

OSADA R., FUNKHOUSER T., CHAZELLE B., DOBKIN D. 2002. *Shape distributions,* in ACM Trans. on Graphics, Vol. 21 (4), pp. 807-832.

[Ossenbruggen *et al.*, 2001]

OSSENBRUGGEN J. VAN, GEURTS J., CORNELISSEN F., HARDMAN L., RUTLEDGE L. 2001.*Towards second and third generation web-based multimedia,* in Proc. of 10th International Conference on World Wide Web (WWW01), Hong Kong, pp. 479-488.

[Otto, 2005]

OTTO K. 2005. *Semantic Virtual Environments,* in Proc. of 14th international World Wide Web conference, ACM press, Chiba, Japan, pp. 1036-1037.

[Panis *et al.*, 2003]

PANIS G., HUTTER A.; HEUER J.; HELLWAGNER H.; KOSCH H.; TIMMERER C.; DEVILLERS S.; AMIELH M.2003. *Bitstream syntax description: A tool for multimedia resource adaptation within MPEG-21,* EURASIP Signal Processing: Image Communications Journal, Vol. 18 (8), pp. 721-747.

[Pascoe, 1998]

PASCOE J. 1998. *Adding generic contextual capabilities to wearable computers,* in Proc. of 2nd International Symposium on Wearable Computers, pp. 92-99.

[Page *et al.*, 2000]

PAGE M., GENSEL J., CAPPONI C., BRULEY C., GENOUD P., ZIEBELIN D. 2000. *Représentation de connaissances au moyen de classes et d'associations: le système AROM,* Actes des 6èmes Journées Langages et Modèles à Objets, pp. 91-106.

[Piegl, 1991]

PIEGL L. 1991. *On NURBS: a survey,* in IEEE Computer Graphics and Applications, Vol. 11 (1), pp. 55-71.

[Pittarello, 2003]

PITTARELLO F. 2003. *Accessing information through multimodal 3D environments: towards universal access,* in Universal Access in the Information Society, Springer Berlin, Vol. 2 (2), June 2003, pp. 189-204.

[Pittarello *et al.*, 2005]

PITTARELLO F, SELÇUK C.K., AUGUSTO C. 2005. *Context-based management of multimedia documents in 3D navigational environments,* in LNCS 3665, pp. 146-162.

[Pittarello *et al.*, 2006]

PITTARELLO F., DE FAVERI A. 2006. *Semantic description of 3D environments : a proposal based on web standards,* in Proc. of 11th International Conference on 3D Web Technology, Columbia, Maryland, pp. 85-95.

[Polys, 2003]

POLYS N.F. 2003. *Stylesheet Transformations for Interactive Visualization : Towards a Web3D Chemistry Curricula,* in Proc. of 8th International Conference of 3D Web Technology, Saint Malo, France, pp. 85-90.

[Polys *et al.*, 2004]

POLYS N.F., BOWMAN D.A. 2004. *Desktop Information-Rich Virtual Environments: Challenges and Techniques,* in Virtual Reality, Vol. 8 (1), pp. 41-54.

[Power *et al.*, 2004]

POWER R., LEWIS D., O'SULLIVAN D., CONALN O., WADE V. 2004. *A context information service using ontology-based queries,* in Proc. of UbiComp 1th International Workshop on Advanced Context Modelling, Reasoning and Management, Nottingham, England.

[Prud'hommeaux *et al.*, 2007]

Prud'hommeaux E., Seaborne A. 2007. SPARQL Query Language for RDF, W3C Candidate Recommandation, disponible en ligne sur : http://www.w3.org/TR/rdf-sparql-query/

[Raento *et al.*, 2005]

RAENTO M., OULASVIRTA A., PETIT R., TOIVONEN H. 2005. *ContextePhone: a prototyping platform for context-aware mobile applications,* in IEEE Pervasive Computing, Vol.4 (2), pp. 51-59.

[Ramos, 2003]

RAMOS F. 2003. *Modélisation et validation d'un système d'information géographique 3D opérationnel,* Thèse de doctorat, Université de Marne-la-Vallée.

[Reeb, 1946]

REEB G. 1946. *Sur les points singuliers d'une forme Pfaff complètement intégrable ou d'une fonction numérique,* Comptes rendus de l'Académie de Sciences de Paris, Vol. 222, pp. 847-849.

[Reitmayr *et al.*, 2005]

REITMAYR G., SCHMALSTIEG D. 2005. *Flexible parametrization of scene graphs,* in Proc. of IEEE Virtual Reality 2005, pp. 51-58

[Robertson *et al.*, 1991]

ROBERTSON G.G., MACKINLAY J.D., CARD S.K. 1991. *Cone Trees: animated 3D visualization of hierarchical information,* in Proc. of ACM SIGCHI Conference, New Orleans, Louisiana, pp. 189-194.

[Rocchio, 1971]

ROCCHIO J. 1971. *The SMART Retrieval System - Experiments in Automatic Document Processing,* in Relevance Feedback in Information Retrieval: Chapter 14, Prentice-Hall, pp. 313-323.

[Roisin, 1998]

ROISIN C. 1998. *Authoring Structured Multimedia documents,* in Proc. 25th Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'98), Jasna, Slovakia, LNCS 1521, pp. 222 -239.

[Roisin, 1999]

ROISIN C. 1999. *Documents structurés multimedia,* Habilitation de diriger les recherches, spéc. Informatique, Institut National Polytechnique de Grenoble.

[Rui *et al.*, 1997]

RUI Y., HUANG T.S., MEHROTRA S. 1997. *Content-based image retrieval with relevance feedback in MARS,* in Proc. of IEEE International Conference on Image Processing '97, Santa Barbara, CA, pp. 815-818.

[Salton, 1989]

SALTON G. 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer,* Addison-Wesley, Reading, Pennsylvania.

[Schaeffer *et al.*, 2002]

SCHAEFER J. GRANIELA B, HEARD P. AND DE LA CRUZ J. 2002. *Web-Based Repository for STRICOM SNE Objects,* disponible en ligne sur : http://www.sedris.org/download/documentation/web_paper.pdf.

[Schilit *et al.*, 1994]

SCHILIT B., ADAMS N., WANT R. 1994. *Context-aware computing applications,* in Proc. of First International Workshop on Mobile Computing Systems and Applications, pp. 85-90.

[Schuster *et al.*, 2001]

SCHUSTER G., STUCKENSCHMIDT H. 2001. *Building Shared Ontologies for Terminology Integration,* in Proc. of KI-01 Workshop on Ontologies, Vol. 48, disponible en ligne sur : http://ceur-ws.org/Vol-48/KI-STuckenschmidt.pdf.

[Schwinger *et al.*, 2005]

SCHWINGER W., GRÜN CH., PRÖLL B., RETSCHITZEGGER W., SCHAUERHUBER A. 2005. *Context awareness in Mobile Tourisme Guides – A Comprehensive Survey,* Rapport Technique. Johannes Kepler University Linz (2005) disponible en ligne sur : http://www.wit.at/people/schauerhuber/publications/contextAwareMobileTourismGuides_TechRep0507.pdf.

[Seaborne, 2004]

SEABORNE A. 2004. *RDQL – A query language for RDF,* W3C Member submission 9 Jan. 2004, disponible en ligne sur : http://www.w3.org/Submission/RDQL/.

[Sebrechts *et al.*, 1999]

SEBRECHTS M.M., CUGINI J.V., LASKOWSKI S.J., VASILAKIS J., MILLER M.S. 1993. *Visualization of search results: a comparative evaluation of text, 2D, and 3D interfaces,* in Proc. of 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Berkeley, California, pp. 3-10.

[Simard *et al.*, 1998]

SIMARD M., DEGRANDI G., THOMSON K.P.B. 1998. *Adaptation of the wavelet transform for the construction of multiscale texture maps of SAR images,* In Canadian Journal of Remote Sensing, Vol. 24 (3), pp. 264-285.

[Smith *et al.*, 1998]

SMITH J. R., MOHAN R. AND LI C. S. 1998. *Content-based Transcoding of Images in the Internet,* in Proc. of IEEE International Conference on Image Processing, Chicago, USA, pp. 7–11.

[Sommaruga *et al.*, 2007]

SOMMARUGA L., CATENAZZI N. 2007. *Curriculum visualization in 3D,* in Proc. of 12th International Conference on 3D Web Technology, Perugia, Italy, pp. 177-180.

[Sowizral, 2000]

SOWIZRAL H. 2000. *Scene graphs in the new millennium,* in IEEE Computer Graphics and Applications, Jan/Feb. 2000, Vol. 20 (1), pp. 56-57.

[Strauss, 1999]

STRAUSS S. 1999. *System and method for optimizing a scene graph for optimizing rendering,* Platinum technology IP, inc., US Patent n° 5896139.

[Teller, 1992]

TELLER S. J. 1992. *Visibility Computations in Densely Occluded Polyhedral Environments,* Technical Report. UMI Order Number: CSD-92-708, University of California at Berkeley, disponible en ligne sur : http://digitalassets.lib.berkeley.edu/techreports/ucb/text/CSD-92-708.pdf.

[To *et al.*, 1999]

TO D.S.P, LAU R.W.H, GREEN M. 1999. *A method for progressive and selective transmission of multi-resolution models,* in Proc. of ACM Symposium on Virtual Reality Software and Technology, London, UK, pp. 88-95.

[Versprille, 2005]

VERSPRILLE K. 2005. *Dassault Systèmes' Strategic Initiative: 3D XML for Sharing Product Information,* Technology Trends in PLM, Collaborative Product Development Associates, http://www.3ds.com/uploads/tx_user3dsplmxml/3DXML_for_sharing_product_information.pdf.

[Vetro *et al.*, 2002]

VETRO A., DEVILLERS S. 2002. *Delivery Context in MPEG-21,* in W3C Delivery Context Workshop, INRIA Sophia-Antipolis, France, March 2002.

[Vetro *et al.*, 2005]

VETRO A., TIMMERER C. 2005. *Digital item adaptation overview of standardization and research activities,* in IEEE Trans. on Multimedia, Vol. 7 (3), June 2005, pp. 418-426.

[Villard *et al.*, 2000]

VILLARD L., ROISIN C. ET LAYAÏDA N. 2000. *An XML-Based Multimedia Document Processing Model For Content Adaptation,* in Proc. Of Digital Documents and Electronic Publishing (DDEP00), p.104-119.

[Visintini *et al.*, 2007]

VISINTINI D., SPANGHER A., FICO B. 2007. *VRML model of Victoria Square in Gorizia (Italy) from laser scanning and photogrammetric 3D surveys,* in Proc. of Web3D Symposium, Perugia, Italy, pp. 165-168.

[Walczak *et al.*, 2003]

WALCZAK K., CELLARY W. 2003. *X-VRML for advanced virtual reality applications,* in IEEE Computer, Vol.36 (3), pp. 89-92.

[Web3D, 1997]

WEB3D CONSORTIUM. 1997. *Information technology — Computer graphics and image processing — Virtual Reality Model Language (VRML) –Part 1: Functional specification and UTF-8 encoding,* ISO/IEC 14772-1:1997, disponible en ligne sur : http://www.web3d.org/x3d/specifications/vrml/.

[Web3D, 2004]

WEB3D CONSORTIUM. 2004. *Information technology — Computer graphics and image processing — Extensible 3D (X3D) — Part 1: Architecture and base components,* ISO/IEC 19775-1:2004, disponible en ligne sur : http://www.web3d.org/x3d/specifications/

[Web3D, 2005]

WEB3D CONSORTIUM. 2005. *Information technology — Computer graphics and image processing — Extensible 3D (X3D) encodings,* ISO/IEC 19776:2005, disponible en ligne sur : http://www.web3d.org/x3d/specifications/.

[Web3D, 2006]

WEB3D CONSORTIUM. 2006. *Information technology — Computer graphics and image processing — Extensible 3D (X3D) — Part 1: Architecture and base components,* ISO/IEC 19775:2004/Am1:2006, disponible en ligne sur: http://www.web3d.org/x3d/specifications/.

[Weibel *et al.*, 1998]

WEIBEL S., KUNZE J., LAGOZE C., WOLF M. 1998. *Dublin Core Metadata for Resource Discovery,* RFC 2413, disponible en ligne sur : http://www.ietf.org/rfc/rfc2413.txt.

[Weiss *et al.*, 2001]

WEISS I., RAY M. 2001. *Model-based recognition of 3D objects from single images,* in IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 32 (2), pp 116-128.

[Wenyin *et al.*, 2001]

WENYIN L., DUMAIS S., SUN Y., ZHANG H.J., CZERWINSKI M., FIELD B. 2001. *Semi-Automatic Image Annotation,* in Proc. Interact Conference on Human–Computer Interaction, pp. 326-333.

[Williams, 1983]

WILLIAMS L. 1983. *Pyramidal Parammetrics,* in Proc. of 10th Conference on Computer Graphics and Interactive Techniques, pp. 1-11.

[Willis, 2007]

WILLIS S. 2007. *Protein CorreLogo: an X3D representation of co-evolving pairs, tertiary structure, ligand binding pockets and protein-protein interactions in protein families,* in Proc. of 12th International Conference on 3D Web Technology, Perugia, Italy, pp. 71-80.

[Wiza *et al.*, 2005]

WIZA W., CELLARY W., WALCZAK K. 2005. *A Method of Holistic 3D Visualization of Arbitrarily Large Datasets,* in Proc. of 7th IEEE International Symposium on Multimedia, Irvine, CA, USA, December 12 - 14, pp. 151-158.

[Xia *et al.*, 1997]

XIA J.C., EL-SANA J., VARSHNEY A. 1997. *Adaptive real-time level-of-detail based rendering for polygonal models,* in IEEE Trans. on Visualization and Computer Graphics, Vol. 3 n° 2, Apr-Jun 1997, pp. 171-183.

[Yergeau *et al.*, 2004]

YERGEAU F., BRAY T., PAOLI J., SPERBERG-MCQUEEN C.M., MALER E.2004. *Extensible Markup Language (XML) 1.0 (Third Edition),* W3C Recommendation, disponible en ligne sur : http://www.w3.org/TR/REC-xml.

[Yin *et al.*, 1997]

YIN L., BASU A. 1997. *MPEG4 face modeling using fiducial points,* in Proc. of International Conference on Image Processing (ICIP'97), Vol. 1, pp. 109-113.

[Zaharia *et al.*, 2001]

ZAHARIA T, PRETEUX F. 2001A. *Three-dimensional shape-based retrieval within the MPEG-7 framework,* in Proc. of SPIE Conference 4304 on Nonlinear Image Processing and Pattern Analysis XII, San Jose, CA, pp. 133-145.

[Zaharia *et al.*, 2002]

ZAHARIA T., PRETEUX F. 2002. *Shape-based retrieval for 3D mesh models,* in Proc. of IEEE International Conference on Multimedia and Expo, Vol. 1, pp 437-440.

[Zlatanova, 2000]

ZLATANOVA S. 2000. *On 3D topological relationships,* in Proc. of 11th International Workshop on Database and Expert Systems Applications, London, UK, pp. 913-919.

Note : l'ensemble des liens vers les documents en ligne a été vérifié le 26/10/2007.

Résumé

Le nombre d'applications et de dispositifs d'accès adoptant le 3D accroît sans cesse. Cette tendance encourage la création d'outils pour la recherche, la réutilisation et l'adaptation de données 3D. La plupart du temps, ces outils sont développés dans le cadre d'une application spécifique. Ceci est en partie dû au fait que les données 3D ne sont décrites généralement que par leur géométrie et par leur apparence ce qui rend difficile leur utilisation indépendamment d'un domaine d'application spécifique.

Nous proposons une solution qui complète la description par des annotations sémantiques et les organise au sein d'un modèle (3DSEAM) suivant la dimension décrite (géométrie, apparence, topologie, sémantique et profil média). Afin d'accommoder la variabilité du modèle en termes de descriptions sémantiques, nous proposons une extension d'OQL spécifique au modèle 3DSEAM. Une plate-forme (3DAF) fédère les moyens d'instanciation et d'interrogation des descriptions. Au-dessus de 3DAF, nous construisons une plate-forme (3DSDL) pour la réutilisation de données 3D et de leur sémantique. Nous adoptons une approche d'adaptation de données 3D à base de règles interprétées et exécutées par une plate-forme d'adaptation (Adapt3D). La réutilisation et l'adaptation s'appuient sur l'utilisation d'un standard XML de représentation de données 3D : le X3D. Les transformations apportées aux données sont décrites en utilisant XSLT.

Une application de gestion de scènes 3D urbaines est proposée pour valider les divers formalismes introduits par notre approche. L'application s'appuie sur une représentation MPEG-7 de l'entrepôt de descriptions. L'interrogation est assurée par XQuery.

Mots-clés : 3D, adaptation, recherche, réutilisation, sémantique

Abstract

The number of applications and access devices 3D enabled increases steadily. This trend is encouraging the development of tools for the retrieval, the reuse and the adaptation of 3D data. Usually, such tools are developed in the narrow context of a specific application. This is partly because the 3D data are usually defined only by their geometry and by their appearance and are difficult to be considered in contexts independent from a specific application domain.

We consider these issues by proposing a solution that enhances 3D data description by semantic annotations and organizes them into a model (3DSEAM) according to the dimension described: geometry, appearance, topology, semantics and media profile. To accommodate the variability of the model in terms of semantic descriptions, we propose an extension of the OQL using 3DSEAM specific constructs. A platform (3DAF) federates the instantiation and the interrogation of such descriptions. Above 3DAF, we build a platform focusing on the reuse of 3D data and their semantics (3DSDL). In addition, we are adopting an approach of adapting 3D data based on semantic rules interpreted and executed by an adaptation platform (Adapt3D). The reuse and the adaptation processes employ an XML standard for 3D data: X3D. The transformations applied to data are implemented using XSLT.

An application that concerns the management of 3D urban scenes is proposed in order to validate the various formalisms introduced by our approach. The application is based on the instantiation of the description model using MPEG-7. The query process is supported by XQuery.

Keywords : 3D, adaptation, search, reuse, semantics