Du texte à la génération d'environnements virtuels 3D : application à la scénographie théâtrale

Tahiry Andriamarozakaniaina

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse 2 Le Mirail (UT2 Le Mirail)

Cotutelle internationale avec :
Université d'Antananarivo - MADAGASCAR

Présentée et soutenue par :
Tahiry Zaka Filamatra ANDRIAMAROZAKANIAINA
Le mardi 25 septembre 2012

Titre :

Du texte à la génération d'environnements virtuels 3D : Application à la scénographie théâtrale

---

ED ALLPH@ : Arts du spectacle

Unité de recherche :
LLA Creatis

Directeur(s) de Thèse :

Mme Monique MARTINEZ THOMAS
Mr Arthur RANDRIANARIVONY

Rapporteurs :

Mr Claude CADOZ
Mr Remi RONFARD

Autre(s) membre(s) du jury :
Mme Annie LUCIANI
Mme Dominique BRETON
Mme Véronique GAILDRAT

Du texte à la génération d'environnements virtuels 3D : Application à la scénographie théâtrale.

Remerciements

Je voudrais, tout particulièrement, remercier M Claude CADOZ et M Remi Ronfard qui m'ont fait l'honneur d'être les rapporteurs de cette thèse, et qui m'ont permis d'améliorer la qualité de ce rapport par leurs remarques et commentaires précieux.

Je tiens à remercier aussi Mme Monique Thomas Martinez et M. Arthur Randrianarivony, mes deux co-directeurs de thèses, pour leur disponibilité, leurs conseils, et leur soutien. Merci à tous les deux pour leur aide précieuse et pour m'avoir accompagné tout au long de ces années.

Je ne remercierai jamais assez Mme Véronique Gaildrat pour tout ce qu'elle a fait pour moi durant ces années, depuis mon DEA à Fianarantsoa jusqu'à l'aboutissement de cette thèse. Elle m'a fourni des conditions idéales pour mener à bien ce travail. Elle est toujours parvenue à me consacrer du temps, et cela malgré sa charge de travail. Elle a su guider mon travail vers des voies que, sans elle, je n'aurais pas explorées, ce mémoire ne serait pas aussi abouti sans son aide.

J'adresse mes remerciements au directeur de l'IRIT (Institut de Recherche en Informatique de Toulouse), pour avoir accepté de m'accueillir au sein du laboratoire. Je tiens à remercier tous les membres du laboratoire et plus particulièrement les membres de l'équipe VORTEX qui m'ont apporté leur aide lors de mes séjours parmi eux.

Je remercie de même tous les membres du Laboratoire LLA de l'Université de Toulouse le Mirail et en particulier M. Matthieu Pouget qui a toujours montré un réel intérêt à mes travaux et qui m'a donné maintes fois des conseils judicieux.

Merci aux amis..., la liste est loin d'être exhaustive pour leur soutien et leurs encouragements.

Une pensée affectueuse à ma famille qui m'a soutenu tout au long de mes études.

Enfin, je remercie le service de coopération et d'action culturelle (SCAC) au sein de l'ambassade de France à Antananarivo, et l'école doctorale interdisciplinaire de Madagascar (EDIM) pour leur aide financière si précieuse.

Table des matières

Index des figures

Index des tables

Partie I- Introduction

I.1- Introduction

Les environnements virtuels sont de plus en plus utilisés dans des cadres très divers. Les applications qui viennent en premier à l'esprit sont liées à l'imagerie 3D pour la simulation, les effets spéciaux et les jeux vidéo. Pour autant, les environnements virtuels ne sont pas limités à ces domaines et se retrouvent de plus en plus utilisés pour faciliter l'aménagement urbain, d'intérieurs ou même paysager. Les concepteurs souhaitent pouvoir créer des environnements de plus en plus réalistes. De ce fait, la conception et la modélisation d'environnements virtuels sont devenus aujourd'hui un processus complexe. L'utilisation des modeleurs géométriques classiques nécessite une maîtrise des modèles géométriques sous-jacents en plus de la maîtrise totale de l'interface utilisateur de l'outil proprement dit. En effet, avec ce genre d'outils, qualifiés de modeleurs *impératifs*, le concepteur doit avoir défini au préalable toutes les propriétés géométriques des objets constituant l'environnement.

Actuellement, la majorité des outils de modélisation proposent des versions d'apprentissage (version d'évaluation) permettant leur accès à tout un chacun, ce qui implique que tout utilisateur d'un ordinateur est potentiellement un créateur d'images, voire de mondes virtuels.

Ces outils sont parfaitement maîtrisés par des spécialistes formés à leur utilisation, et permettent de générer des scènes de grande qualité. Ils sont très performants, très aboutis, pourtant, il semble que ce sont ces mêmes outils qui constituent le facteur de blocage pour le développement de la modélisation 3D vers le grand public [20]. Même s'ils ont été conçus et développés par des équipes différentes, ces logiciels présentent les mêmes spécificités et ainsi les mêmes inconvénients :

- les modèles géométriques et numériques sous-jacents sont omniprésents, ce qui demande une connaissance approfondie de ces modèles de la part de l'utilisateur,

- les possibilités d'abstraction sont réduites, obligeant l'utilisateur à raisonner en termes de primitives géométriques[1] ou d'entités de très bas niveau[2] au lieu des objets qu'il manipule,
- enfin, les interfaces utilisateurs ont pour point commun de décourager instantanément toute personne n'étant pas très motivée par l'apprentissage du logiciel, étant donnée la difficulté d'appréhender leur complexité.

Typiquement, lors de la modélisation d'une scène, l'utilisateur part d'une image mentale de cette scène et va tenter de générer le modèle tridimensionnel correspondant, en ajoutant et en manipulant des formes de base et des primitives élémentaires. Dans ce cas, l'utilisateur doit avoir en tête, dès le départ, l'image mentale de la scène résultante avant d'avoir commencé l'étape de modélisation, car les modifications s'avèrent difficiles. Les modeleurs actuels ne permettent pas une démarche itérative où l'image mentale se construit au fur et à mesure de la création de la scène. Ils imposent à l'utilisateur de fixer très tôt des positions, des orientations, bref des valeurs numériques alors que le concepteur raisonne plutôt au niveau des objets qui existent dans le monde réel. Pour créer des modeleurs tridimensionnels permettant d'accompagner le raisonnement humain, il est nécessaire d'utiliser d'autres approches, notamment basées sur la modélisation par contrainte, la modélisation déclarative [21], tout en intégrant des lois physiques simples (gravité, gestion des collisions, etc.).

Dans cette introduction, nous allons présenter les principaux points développés dans cette étude. Plus précisément, cette introduction s'articule autour de trois axes :

- le contexte de l'étude,
- les objectifs à atteindre,
- la structure du mémoire.

I.1.1) Contexte de l'étude

Dans cette thèse, nous nous intéressons à l'étude et à la création d'outils de description et de génération automatique d'environnements virtuels 3D, appliquées à la scénographie théâtrale. Cette étude s'inscrit dans le cadre d'un projet pluridisciplinaire, DRAMA.

[1] Formes géométriques unitaires pouvant être des sphères, cubes, cylindres, etc.
[2] Sommets, arêtes, facettes planes, points de contrôle, vecteurs normaux ou tangents, etc.

Ce projet est mené en collaboration entre le laboratoire LLA[3] de l'université du Mirail, et l'équipe VORTEX[4] du laboratoire IRIT[5] de l'université Paul Sabatier.

Son objectif est d'étudier et de développer des outils informatiques à destination des metteurs en scène, que ce soit lors du processus de création du spectacle vivant, ou dans le cadre de la formation des metteurs en scène, ou bien à destination des assistants des metteurs en scène.

Dans ce mémoire, nous présenterons les travaux effectués, comme étant à destination des metteurs en scène, même si en réalité le travail de prise de notes du travail de mise en scène est réalisé la plupart du temps par des assistants (qui sont même dans la majorité des cas des assistantes…).

Ce projet se décline en plusieurs modules qui sont : DRAMAsurtitrage, DRAMAtexte, DRAMAscène et DRAMAmémoire, qui correspondent à des étapes du processus de création théâtrale.

Les précédents travaux menés autour de DRAMA ont permis de spécifier les besoins essentiels des praticiens du théâtre, et ainsi de concevoir les grandes lignes d'un outil qui leur serait destiné. Ces travaux ont été menés en collaboration avec Mathieu Pouget [50], metteur en scène de la troupe Les Anachroniques[6], et dont le doctorat a porté sur la spécification du projet DRAMA.

D'autres travaux sont actuellement menés, toujours autour du projet DRAMA, plus axés sur la codification et la représentation de personnages expressifs, prenant en compte la dimension émotionnelle [51].

I.1.1.1- DRAMAsurtitrage

C'est un module de DRAMA à la fois linguistique et graphique, destiné à faciliter le surtitrage des pièces de théâtre. Le fait que ce soit du spectacle vivant introduit de nombreuses difficultés supplémentaires, tant au niveau du suivi en temps réel du texte tel qu'il est énoncé par les acteurs, qu'au niveau du choix de l'emplacement le plus approprié pour la projection sur l'espace scénique.

---

3 Lettres, Langages et Arts

4 Visual Objects from Reality To Expression

5 Institut de Recherche en Informatique de Toulouse

6 http://www.anachroniques.fr/
http://blogs.univ-tlse2.fr/universcenes/

I.1.1.2- DRAMAtexte

DRAMAtexte, est un module destiné à obtenir et traiter de façon textuelle les caractéristiques d'une mise en scène, voulue par l'auteur ou le metteur en scène, grâce à un outil d'identification automatique et manuelle des différents indicateurs de scénographie. Ces indications ont été introduites par l'auteur dans le texte ou ajoutées par le metteur en scène, mais pourront être identifiées grâce à un outil de balisage intégré dans DRAMAtexte.

DRAMAtexte est également destiné à être un outil d'interrogation du texte théâtral, en rapport avec la problématique de la mise en scène théâtrale, afin de fournir des informations synthétiques. Cette possibilité s'adresse aux personnes désireuses d'analyser le texte sur un plan littéraire, grâce à la possibilité d'effectuer des requêtes (simples et composées). Ceci doit fournir, par exemple, des résultats concernant l'analyse conversationnelle, les récurrences, ou encore la description scénique. La visualisation des résultats doit se faire sous une forme appropriée (listes, graphiques, schémas, visualisation 2D ou 3D (réalité virtuelle)) et peut servir de cadre à l'élaboration de spectacles.

I.1.1.3- DRAMAscène

Ce module est pensé au départ comme un outil de visualisation de scène devant permettre à tous les agents du spectacle et au metteur en scène de travailler ensemble, afin de rendre possible une vision globale du travail scénique. Cet outil devra permettre de répertorier et prendre en compte différents aspects de la création théâtrale (outils spécifiques de notation des mouvements, du dialogue, de la scénographie, des lumières, de la musique, de la bande sonore, etc.).

Dans un premier temps DRAMAscène est conçu comme un modeleur déclaratif [42], devant permettre de visualiser une scénographie virtuelle à partir de propriétés en entrée qui proviennent de résultats obtenus à partir de l'outil DRAMAtexte. Il devra ensuite permettre la prise en compte de particularités liées aux besoins des métiers du spectacle (accessoiriste, décorateur, costumier, etc.). Ces contraintes supplémentaires devront être intégrées afin d'être perceptibles dans la mise en scène virtuelle résultante.

I.1.1.4- DRAMAmémoire

C'est un volet de mémorisation destiné à conserver et à exporter les données scéniques stockées dans DRAMAtexte et DRAMAscène :

- il est destiné à l'apprentissage de la mise en scène, notamment dans le cadre de l'enseignement primaire et secondaire,
- et à la conservation du processus de création théâtrale.

I.1.2) But du projet DRAMA

Dans le cadre de ce projet, notre objectif principal est de développer un outil de conception d'environnements virtuels, convivial et intuitif, destiné à un utilisateur non expérimenté et non spécialiste des modeleurs géométriques. Lors de cette activité de conception, l'utilisateur part d'une image mentale initiale de la scène, qu'il peut affiner grâce au modeleur. Cette image imprécise est amenée à évoluer et à se préciser au fur et à mesure du déroulement des étapes de modélisation, grâce à la façon dont se déroule le processus de création via un modeleur déclaratif. Ainsi, l'utilisateur peut tester plusieurs propositions et lancer plusieurs tentatives, et plusieurs essais sont généralement nécessaires avant d'obtenir un résultat qui le satisfait totalement.

L'approche que nous proposons repose sur le concept de modélisation déclarative [42]. Ce type de modélisation permet à un utilisateur de concevoir et de construire des formes, ou des environnements 3D, en fournissant un ensemble de caractéristiques et de propriétés que le système aura à vérifier et à résoudre. A partir de ces propriétés, le modeleur déclaratif explore l'espace des possibilités et détermine un ensemble de solutions répondant aux exigences de l'utilisateur. Celui-ci n'a alors plus qu'à choisir le ou les résultats qui se rapprochent le plus de l'image mentale de ce qu'il souhaitait obtenir. Cette approche permet ainsi au concepteur de se libérer des contraintes géométriques attachées aux modèles manipulés et de se consacrer pleinement à son activité créative, sans avoir à se préoccuper des difficultés liées à la création de scènes 3D.

L'approche déclarative se base sur un ensemble de connaissances intrinsèques qui facilitent l'énoncé des propriétés de la part de l'utilisateur en lui évitant de décrire la totalité de la scène à générer. La base de connaissance permet au modeleur déclaratif d'obtenir les propriétés usuelles qui n'ont pas été fournies par l'utilisateur, relativement au contexte de la scène en cours de création. Ces connaissances peuvent être enrichies au fur et à mesure de l'utilisation de l'outil, afin d'être prises en compte dans les traitements futurs qui reprennent des éléments similaires, afin d'accélérer les processus, quand il s'agit de créer des environnements qui doivent respecter un ensemble de conditions initiales, identiques ou proches.

La mise en œuvre d'un tel système intègre de nombreux domaines de l'informatique. La modélisation déclarative est ainsi considérée comme un domaine combinant plusieurs disciplines comme le montre la Figure 1.

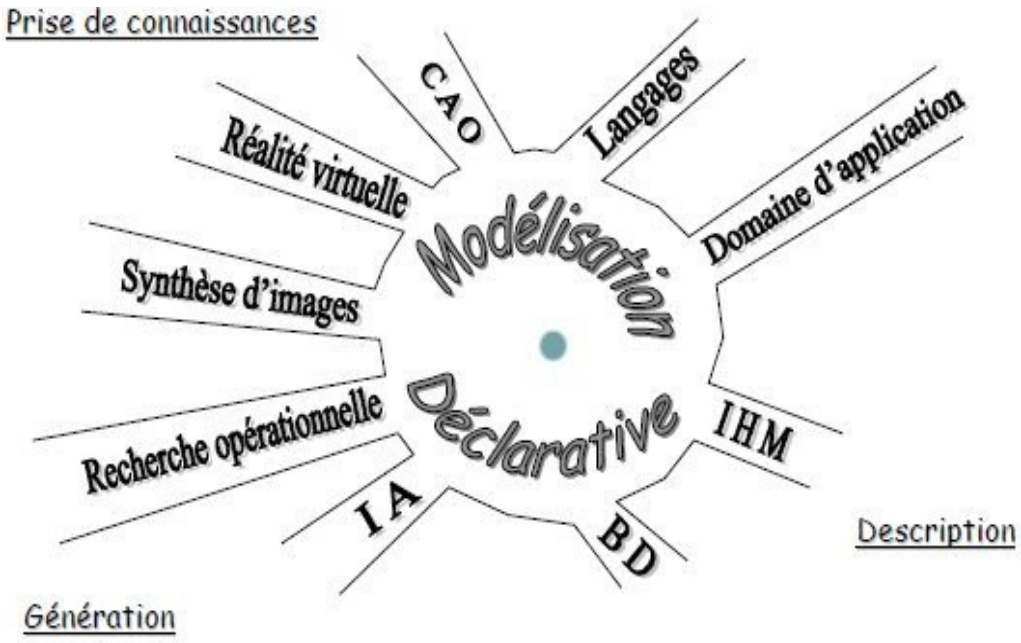


Figure 1: Modélisation déclarative

I.1.3) Démarche

Pour obtenir la visualisation en trois dimensions d'une proposition de mise en scène virtuelle à partir du texte théâtral, quatre étapes sont nécessaires comme l'indique la Figure 2. Ces étapes constituent le noyau des deux premiers outils étudiés dans le cadre de ce projet : DRAMAtexte et DRAMAscène.

La première étape a pour but d'identifier les éléments descriptifs et pour cela d'ajouter des balises dans le texte initial, à l'aide d'un langage à balises basé sur XML[7], et ceci, afin de marquer les éléments constituant la description de l'espace scénique souhaité par le metteur en scène.

Cette première étape est divisée en trois séquences distinctes :

1. le balisage automatique identifiant les éléments structurels (titre, actes, scènes, nom de personnages, didascalies),

[7] *XML : Extensible Markup Language*, ou langage de balisage extensible ; syntaxe reconnaissable à partir des *chevrons* (< >) encadrant les *balises*

2. suivi du balisage manuel réalisé par un utilisateur lecteur, identifiant les éléments descriptifs (éléments de décor, accessoires, éclairages, relations spatiales entre objets ou personnages, etc.),
3. et enfin le balisage manuel effectué par le metteur en scène, afin d'apporter toute modification ou précision qu'il souhaite.

A l'issue de cette première étape les informations, identifiées et notifiées grâce à l'ajout de balises dans le texte de l'auteur, sont définies.

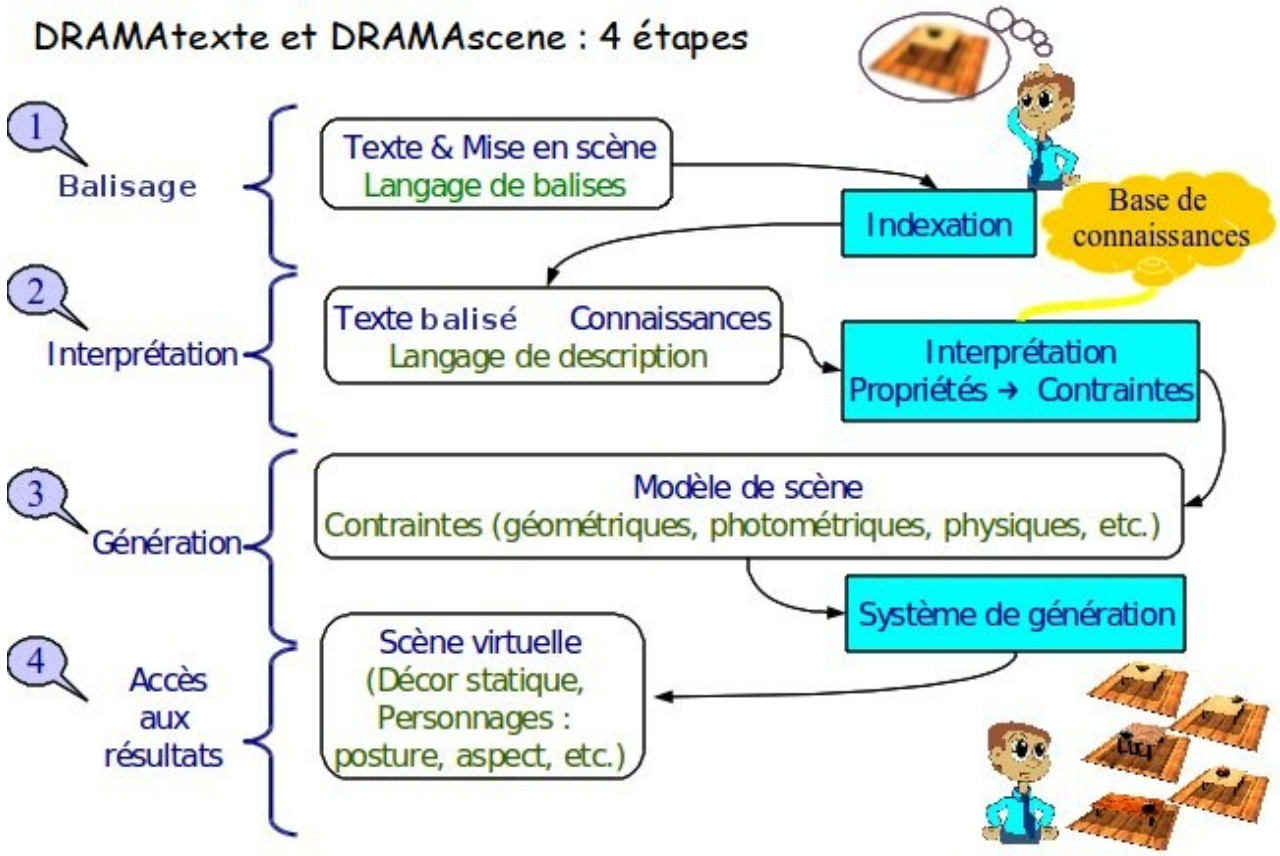


Figure 2: Les 4 étapes de DRAMA (basées sur les étapes d'un modeleur déclaratif)

L'étape suivante va permettre d'interpréter ces propriétés en contraintes numériques, en s'appuyant sur une base de connaissances, complétée au fur et à mesure de l'utilisation de l'outil.

La troisième étape est la résolution des contraintes par un système de génération, afin de calculer une solution et afficher le résultat sous forme d'un environnement virtuel 3D.

La base de connaissance permettra de modéliser et de mémoriser un ensemble de connaissances, comme les caractéristiques géométriques ou fonctionnelles d'objets, la

représentation des personnages, ou toute autre donnée sur les éléments à introduire dans la scène.

La dernière étape consiste en l'instanciation de mises en scènes virtuelles dans l'espace de solutions, afin de présenter à l'utilisateur une, plusieurs ou toutes les possibilités générées à partir des propriétés issues du texte balisé.

À la suite de cette présentation, l'utilisateur a la possibilité de revenir au texte pour modifier manuellement le balisage en ajoutant, modifiant, ou retirant des éléments, afin d'obtenir un nouveau résultat. Enfin, si le résultat lui convient, il peut valider le balisage qui sera conservé pour une utilisation ultérieure. Ce processus itératif permet à l'utilisateur d'affiner le résultat, étape par étape, tout en construisant l'image mentale d'une mise en scène possible.

I.1.4) Structure du mémoire

Le présent document se divise en deux chapitres. Le premier est un chapitre de tour d'horizon présentant des travaux et des projets connexes à notre étude, où on retrouve la problématique et chacun des termes clés du sujet qui nous préoccupe.

Il détaille le concept de modélisation déclarative tel qu'il a été appliqué, ainsi que les principaux systèmes de génération par contraintes.

Le deuxième chapitre présente notre contribution au projet DRAMA qui s'est principalement consacré à l'étude et au développement de solutions pour DRAMAtexte et DRAMAscène. Plusieurs méthodes et algorithmes étudiés, conçus et développés seront présentés en détail, ainsi que des résultats que nous avons obtenus.

Nous terminerons ce mémoire par une brève conclusion et les perspectives qui peuvent se rapporter à nos travaux.

Partie II- État de l’art

II.1- Introduction

Dans ce chapitre, nous présentons les principaux domaines de recherche en rapport avec notre étude. En effet, notre travail intègre les principes introduits par la modélisation déclarative, dans le domaine de la réalité virtuelle et notamment de la génération automatique d'une scène 3D.

Ce chapitre passe en revue et évalue les évolutions importantes accomplies dans ce domaine dans la dernière décennie. Pour cela, dans la première partie, nous allons détailler la modélisation déclarative, et particulièrement les différentes phases du processus de génération et présenter quelques modeleurs, parmi les plus significatifs.

Ensuite, dans la deuxième partie, nous allons présenter des travaux et projets présentant des similitudes avec DRAMA, et qui intègrent la possibilité de générer des scènes 3D à partir de descriptions.

Enfin, dans la troisième partie, nous allons décrire les moteurs physiques qui vont gérer les aspects physiques de chaque objet à placer dans la scène virtuelle (masse, collision, friction, etc.).

II.2- Modélisation déclarative

La modélisation déclarative propose des moyens de s'affranchir des difficultés d'utilisation de la modélisation géométrique classique (dite impérative). Elle est une voie de recherche relativement récente, initiée par Michel Lucas qui la définit comme suit :

« *L'objectif de la modélisation déclarative de formes est de permettre de générer des formes (ou des ensembles de formes) par la simple donnée d'un ensemble de propriétés ou de caractéristiques. »*

L'ordinateur est chargé de généraliser l'univers des formes potentielles, afin de sélectionner celles correspondant à la définition donnée. Le concepteur n'a plus qu'à choisir, à l'aide d'outils appropriés, la ou les formes qui lui conviennent.

Cette définition, qui correspond aux travaux menés à Nantes sous la direction de Michel Lucas(projet ExploFormes) [42] et à Limoges sous celle de Dimitri Pléménos (projet MultiFormes) [47], a été étendue par Véronique Gaildrat [20] qui la formule ainsi :

> *« Le but de la modélisation déclarative est d'offrir à un concepteur la possibilité de décrire une entité ou une scène à modéliser, non plus en fournissant des données numériques, mais en exprimant l'image mentale qu'il a de la scène grâce à un ensemble de propriétés énoncées dans un langage quasi-naturel. Le concepteur donne les propriétés géométriques, photométriques et physiques de l'environnement en énonçant les caractéristiques des différents éléments de la scène ainsi que les relations entre ces éléments. Ces propriétés permettent, à l'aide d'un ordre (instruction) unique, de réaliser une opération complexe qui aurait impliqué un nombre important d'actions élémentaires avec un modeleur géométrique impératif. »*

Ainsi, la modélisation déclarative a pour but d'ajouter de la sémantique dans un domaine qui était jusqu'à présent uniquement mathématique ou géométrique. Cette approche rejoint une tendance actuelle majeure de l'informatique, qui est de déléguer les tâches fastidieuses à l'ordinateur pour que l'utilisateur n'ait plus qu'à effectuer les tâches intéressantes et créatives.

Pour les utilisateurs qui préfèrent penser et agir à un haut niveau d'abstraction, on peut faire une analogie entre les modeleurs déclaratifs et les langages à objets, et entre les modeleurs impératifs et les langages d'assemblage [40]. En outre, on est en droit d'attendre d'un modeleur déclaratif qu'il détecte et mette en évidence les incohérences potentielles de la description fournie en entrée.

Le groupe de travail GEODE[8] [8] a décrit les différences entre modélisation déclarative et modélisation effectuée à l'aide d'un modeleur impératif. Ainsi, le tableau suivant nous montre les principales différences entre ces deux catégories de modeleurs.

Une conséquence évidente des définitions précédentes est que le développement d'un modeleur déclaratif doit faire appel à des compétences au-delà du domaine de l'infographie et de la synthèse d'images :

[8] équipe du GDR-PRC AMI. Elle est constituée de l'équipe MGII de l'IRIN, du CERMA de l'école d'architecture de Nantes, et de l'équipe synthèse d'images de l'École des Mines de Nantes (www.emn.fr/dept_info/GEODE).

- de nombreuses études ont montré l'intérêt des interactions multimodales pour un modeleur déclaratif et leur apport dans la conception d'une interface conviviale ; ces techniques sont issues du domaine de l'IHM (Interface Homme Machine),
- la possibilité d'expression orale doit s'appuyer sur une reconnaissance et une analyse du langage,
- les différents principes de génération employés reposent sur des techniques développées en intelligence artificielle et en recherche opérationnelle,
- enfin un modeleur, déclaratif ou non, peut être considéré comme un outil de conception assistée par ordinateur.

Les travaux réalisés au sein de l'équipe GEODE [8] ont mis en évidence la présence de trois phases successives au sein d'un modeleur géométrique. Ces trois phases se succèdent au cours d'un processus itératif qui permet, par affinage successif de la description, d'obtenir une solution qui satisfait l'utilisateur.
Il est intéressant de préciser que la solution générée peut-être radicalement différente de l'image mentale initiale de l'utilisateur, tout en correspondant pourtant à la description qu'il avait fournie (en particulier, ceci est vérifié dans [17]).
L'un des avantages que tire la modélisation déclarative de ce processus itératif, est la possibilité pour l'utilisateur (ou pour l'outil de modélisation lui-même) de détecter des incohérences ou des problèmes très tôt dans le processus de conception. Ceci est rendu possible par la capacité d'un modeleur déclaratif à présenter des scènes très tôt (des esquisses) alors que toutes les propriétés ne sont pas encore fixées.

Modeleur impératif	Modeleur déclaratif
Le concepteur a une idée d'objet à modéliser	À partir de l'idée : description d'une scène ou de la forme d'un objet à partir de ses propriétés
Un processus mental doit amener de l'idée à un ensemble d'opérations élémentaires fournies par le modeleur	L'image mentale de l'objet solution n'est pas directement utilisée pour la conception
Construction de l'objet par planification mentale des opérations élémentaires	Le modeleur fournit les solutions à partir de propriétés -> pas nécessaire de vérifier la validité
Réalisation des opérations élémentaires, pas à pas	À partir des solutions obtenues : processus incrémental de modification des spécifications
Tests de validité négatifs => reprendre la décomposition	Le concepteur peut se concentrer sur des tâches de haut niveau d'abstraction
Modification à apporter => reprendre le processus de conception depuis le début	Le modeleur déclaratif fournit un modèle de solutions à partir desquelles il est possible d'obtenir une à une, plusieurs ou l'ensemble des solutions. Le modèle géométrique sous-jacent est totalement caché au concepteur

Tableau 1: Tableau de comparaison des modeleurs impératifs et déclaratifs à partir des travaux du groupe GEODE [8] [20]

II.2.1)Principes :

Un modeleur déclaratif est généralement organisé en phases successives qui bouclent selon un processus de conception en spirale (voir Figure 3). La description permet au concepteur de définir les propriétés de la scène à construire.

L'énoncé des propriétés peut se faire à différents niveaux d'abstraction et avec plusieurs modalités : énoncé en langage naturel qui nécessite un outil de compréhension du langage oral ou écrit, énoncé construit à l'aide de sélections via une interface graphique

appropriée, énoncé sous la forme d'un script composé de propriétés d'un niveau d'abstraction moindre.

Selon le niveau d'abstraction de la description, celle-ci nécessite une interprétation des propriétés en contraintes de bas niveau, utilisées par le système de génération.

La génération est l'étape pendant laquelle le système calcule une, plusieurs ou l'ensemble des solutions respectant la description fournie par le concepteur.

La prise de connaissance permet au concepteur de visionner les solutions obtenues afin de pouvoir choisir celle qui lui convient et modifier la description le cas échéant.

Du fait de cette organisation, un modeleur déclaratif intègre généralement :

- un *interpréteur* pour traduire les propriétés énoncées en contraintes de plus bas niveau,
- un *solveur* de contrainte calculant et produisant le modèle 3D solution,
- un outil de visualisation des solutions 3D pour que le concepteur puisse éventuellement naviguer à l'intérieur des solutions et sélectionner celles qui lui conviennent.

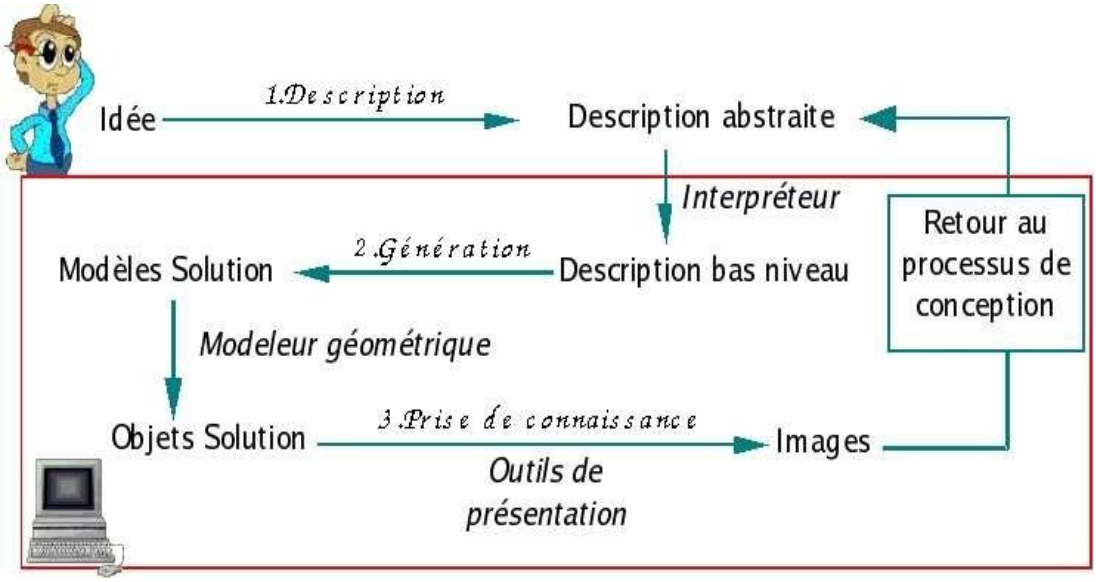


Figure 3: Les phases de la modélisation déclarative [20] d'après GEODE

II.2.1.1- Phase de description :

Durant cette phase, l'utilisateur décrit la scène ou modifie une description antérieure après visualisation. Le principal problème de cette phase consiste à utiliser un système

permettant d'éviter autant que possible toute ambiguïté dans la description. Par exemple, la notion de gauche et de droite n'est pas a priori clairement définie si on ne précise pas que l'on parle de la gauche de l'utilisateur ou de celle de l'objet concerné (orientation déictique ou intrinsèque) [20].

Une phase de description peut proposer divers moyens d'interaction à l'utilisateur :

- interface à base d'objets graphiques tels que les menus déroulants, boîtes de dialogue, zone de texte, etc.,
- structures langagières prédéfinies (structures de trait),
- langage (quasi) naturel.

<table>
<tr><th colspan="5">Propriétés</th></tr>
<tr><td>Strictes</td><td colspan="2">Vérification booléenne stricte de la propriété.</td><td>Floues</td><td>Associées à une fonction d'appartenance et des modificateurs.</td></tr>
<tr><td rowspan="9">Générales ou Spécifiques</td><td rowspan="2">Morphologiques</td><td>Explicites</td><td colspan="2">Devant être mentionnées par le concepteur.</td></tr>
<tr><td>Implicites</td><td colspan="2">Mises à des valeurs par défaut sauf indications contraires.</td></tr>
<tr><td rowspan="4">Structurelles</td><td>Unaires</td><td colspan="2">Relatives à un seul élément.</td></tr>
<tr><td>Binaires</td><td colspan="2">Relation entre deux éléments.</td></tr>
<tr><td rowspan="2">N-aires</td><td colspan="2">Définissent les structures ou organisations complexes.</td></tr>
<tr><td colspan="2">Définissent une structure hiérarchique méréotopologique (relation partie-tout, inclusion topologique).</td></tr>
<tr><td rowspan="3">Relatives à l'apparence</td><td colspan="3">Conditions spatio-temporelles d'éclairage.</td></tr>
<tr><td colspan="3">Textures et propriétés photométriques des matériaux utilisés.</td></tr>
<tr><td colspan="3">Style.</td></tr>
<tr><td rowspan="2">Mécaniques et Physiques</td><td colspan="3">Liaisons dynamiques entre parties.</td></tr>
<tr><td colspan="3">Propriétés des matériaux.</td></tr>
<tr><td rowspan="3">Relations spatiales</td><td colspan="2">Positionnement absolu</td><td colspan="2">A gauche, à droite, au milieu, etc.</td></tr>
<tr><td colspan="2">Positionnement relatif</td><td colspan="2">A gauche de, devant, à côté de, contre, etc.</td></tr>
<tr><td colspan="2">Positionnement contextuel</td><td colspan="2">Intrinsèque grâce à un objet polarisant (tableau) ou à une situation particulière (conversation).</td></tr>
</table>

Tableau 2: Classement des propriétés en modélisation déclarative [20]

Il est aussi possible d'utiliser divers périphériques d'interaction pour cette phase. On peut imaginer une interface multimodale (interface homme-machine qui combine plusieurs moyens de communication entre l'utilisateur et la machine) [35] combinant :

- souris,
- parole,
- geste (par caméra ou gants de données),
- périphériques à retour d'effort,
- clavier.

La principale difficulté de cette étape descriptive ne réside pas dans les moyens offerts à l'utilisateur pour décrire la scène qui peuvent être plus ou moins assistés, mais dans l'interprétation de cette description en contraintes géométriques, prises en charge lors de l'étape suivante, qui correspond à la phase de génération.

Pour pouvoir effectuer cette interprétation, le modeleur déclaratif doit posséder des informations sur le contexte de l'environnement à créer, et doit pouvoir déduire de ces informations les données qui n'ont pas été exprimées par l'utilisateur. Ceci afin de simplifier la tâche de description et générer les contraintes qui correspondront à ce qui est réellement souhaité par l'utilisateur.

Cette étape réellement critique s'appuie sur un ensemble de connaissances liées au contexte de l'environnement à créer, et doit également s'appuyer sur le modèle géométrique sous-jacent traité par la phase de génération.

Les différents modeleurs déclaratifs existant traitent un ensemble de propriétés qui ont fait l'objet d'une proposition de classement (voir Tableau 2) [20].

II.2.1.2- Phase de génération :

Durant cette phase, le modeleur tente de générer les scènes correspondant à la description fournie par l'utilisateur.

La description ayant été traduite en un ensemble de contraintes, c'est cet ensemble de contraintes que va tenter de résoudre le solveur qui est au centre de la phase de génération.

Si les contraintes correspondant à la description sont tellement précises qu'elles correspondent à une seule solution possible (on parle de problème *bien contraint*) alors c'est cette solution qui est calculée et proposée à l'utilisateur.

Étant donné les contextes auxquels s'appliquent les modeleurs déclaratifs, ce cas là n'est quasiment jamais rencontré.

Si les contraintes correspondant à la description sont incohérentes (on dit que le problème est « *sur-contraint* »), il peut n'y avoir aucune solution, auquel cas le modeleur devra être capable de détecter la partie de la description qui pose problème pour proposer à l'utilisateur des méthodes qui évitent le blocage.
On rencontre facilement ce cas, il suffit que l'utilisateur indique que la table est à gauche de la fenêtre ET que la table est à droite de la fenêtre ...
Si les contraintes sont cohérentes mais encore imprécises (on parle de problème « *sous-contraint* »), il peut y avoir une énorme quantité de solutions. C'est le cas le plus souvent rencontré en modélisation d'environnements virtuels où la description reste assez vague, étant donné le grand nombre d'éléments composant une scène.
Dans ce cas, le traitement varie selon les approches :

- génération d'une unique solution ; plusieurs approches sont possibles : fournir la première trouvée par le solveur même si elle est rarement équilibrée, choisir une solution aléatoire dans l'espace des solutions, adopter une méthode heuristique de choix d'une solution « *représentative* »,
- génération de toutes les solutions,
- génération d'un échantillon de solutions « *représentatif* » de l'ensemble des solutions.

Ces variantes ont un coût en terme d'occupation mémoire et de temps de calcul très variable, en fonction de la complexité du problème, et dans une optique visant des temps interactifs on peut envisager que le modeleur choisisse la variante adéquate. Pour effectuer cette phase, différentes techniques ont été envisagées :

- arbres de parcours explicite,
- grammaires génératives,
- approche procédurale spécifique [9],
- transformation de la description en un ensemble de règles et utilisation d'un moteur d'inférence capable de créer de nouveaux faits et de déduire des caractéristiques géométriques [47],
- transformation de la description en contraintes et utilisation d'un algorithme numérique ou d'un solveur de contraintes pour résoudre le système [4] [20],
- application des techniques stochastiques [52],
- application d'un optimiseur numérique.

Comme pour l'étape de description, une façon de concevoir un modeleur déclaratif générique serait qu'il intègre le plus de techniques possibles, néanmoins le choix de la technique adéquate par rapport au problème est loin d'être un problème trivial. Ce qui fait que les modeleurs déclaratifs disposent généralement d'un système de résolution adapté, développé spécifiquement pour le type de problèmes traités par le solveur.

II.2.1.3- Phase de visualisation :

Si aucune scène n'a pu être générée, le modeleur doit communiquer à l'utilisateur ses recommandations afin que la description puisse être modifiée et qu'au moins une solution puisse être trouvée.

Si au contraire de très nombreuses scènes ont été générées, alors plusieurs solutions sont possibles :

- trier et classer l'ensemble des solutions de façon à présenter à l'utilisateur une solution représentative de chaque classe [6], problème très vaste et, a priori, quasiment impossible à appliquer à un modeleur générique,
- comparer les différentes solutions en mettant en valeur les similitudes et les différences,
- choisir une solution au hasard ou sélectionner une solution à partir de critères de qualité de cette solution (critères très dépendants du contexte) et laisser l'utilisateur affiner sa description afin de réduire l'espace de solutions.

La visualisation des résultats va dépendre du modèle de solution fourni par le solveur. Généralement le solveur va fournir les données permettant de visualiser la scène grâce à un modeleur 3D, mais si le solveur génère un graphe de scène de type VRML alors le résultat pourra être affiché via un visionneur VRML. Des travaux ont porté sur une présentation des résultats en calculant un point de vue [13] et un chemin de la caméra, de pour que l'utilisateur puisse voir au mieux la scène générée, en permettant par exemple de voir le plus grand nombre d'objets, grâce à la position fixe de la caméra ou grâce au trajet qu'elle suit dans la scène 3D.

Ici encore, la recherche de la généricité pour le modeleur déclaratif n'est que difficilement compatible avec cette voie trop spécifique et trop dépendante du modèle généré par le solveur.

Pour aider le concepteur à visualiser les scènes générées, il est envisageable de s'appuyer sur les éléments issus de la phase de description pour faire les choix permettant

de choisir une solution dans l'espace des solutions, puis de naviguer à l'intérieur même des solutions.

L'objectif étant alors de choisir les vues qui permettent à l'utilisateur de vérifier que les propriétés ont bien été prises en compte en privilégiant l'affichage des entités les plus contraintes, ou des points de vues considérés comme les plus significatifs dans le contexte du problème.

Pour une scénographie virtuelle, on pourra par exemple privilégier le point de vue du metteur en scène (place idéalement située parmi les sièges des spectateurs).

II.2.2)Base de connaissances

On peut considérer les connaissances comme correspondant à des informations contenues dans l'esprit des individus (informations qui peuvent être ou non nouvelles, uniques, utiles ou précises) liées à des faits, des procédures, des concepts, des interprétations, des idées, des observations ou encore des jugements.

Ainsi, nous pouvons, dans le contexte de notre étude, définir les connaissances comme une collection de données permettant de définir une entité, de façon à faciliter l'ajout de cette entité dans la scène. Le modèle de représentation de ces connaissances a été conçu pour que les connaissances soient modifiables, et qu'on puisse les enrichir au fur et à mesure de l'utilisation de l'outil. Ceci est, d'une certaine manière, analogue à une base de données où l'information est stockée avec un modèle spécifique, et présentée conformément aux requêtes des utilisateurs.

Les bases de connaissances permettent de regrouper d'une façon structurée des données relatives à un sujet, à un domaine, de façon à être ensuite exploitées pour calculer un résultat, déduire une action, etc.

Un type spécifique de base de connaissances est appelée ontologie[9]. D'un point de vue philosophique, l'ontologie est la spécification explicite d'une conceptualisation d'une simple vue et abstraite du monde. Pour les systèmes basés sur les connaissances, ce qui « existe » est exactement ce qui peut être représenté.

Les bases de connaissances sont très utilisées dans une grande variété de domaines, allant de la médecine [65] à la génétique [24], et pour des tâches diverses :

[9] Ontologie désigne, dans un cadre non philosophique, l'ensemble des termes ou concepts primitifs qui peuvent décrire de façon canonique une classe particulière de concepts (dans le cadre de notre étude : formes géométriques et relations spatiales).

- déterminer la non-redondance dans les architectures de systèmes d'information,
- gérer des données dont la quantité ne permet pas une exploitation simple, telles que les informations d'assistance ou les manuels d'utilisation,
- permettre une compréhension commune et la communication entre différentes entités,
- faciliter l'interopérabilité entre des systèmes différents [29] [60].

Comme nous l'avons indiqué précédemment, une partie importante de la phase de description de la modélisation déclarative est la gestion sémantique. En d'autres termes, l'interprétation des propriétés données par l'utilisateur lors de la phase de description va mobiliser des connaissances relatives au contexte courant, de façon à générer un ensemble de contraintes qui vont amener au choix d'une solution réellement représentative de la description. La façon dont une entité ou un concept vont être traités va dépendre du contexte. Il y aura plusieurs façons de définir l'apparence ou les fonctionnalités relatives à une entité ou un concept.

Par exemple, selon la façon dont on veut le représenter ou le prendre en compte dans le contexte de l'application, un humain virtuel peut être représenté sémantiquement comme un tout si on souhaite uniquement le représenter statiquement et émotionnellement neutre. Il peut être décrit comme la conjonction de plusieurs éléments (tête, torse, bras, jambes) si on souhaite lui associer une posture particulière ou l'animer. Ou encore, il peut être défini comme un ensemble d'énoncés, indépendamment de toute représentation géométrique.

De nombreux langages et outils ont été proposés pour représenter et utiliser les connaissances. Parmi les outils on peut citer Protégé [10], développé par l'université de Stanford et qui est un des outils de gestion d'ontologies les plus utilisées.

Différents types de représentations tels que la logique ou des règles [27] sont utilisés en représentation formelle pour des ontologies, des logiques de description ou des schémas logiques, et dans diverses applications comme les systèmes experts ou applications de workflow [2].

Lorsque la connaissance d'un domaine est représentée dans un formalisme déclaratif, l'ensemble des objets qui peuvent être représentés est appelé « univers des dialogues » (*speech universe*). L'ensemble des objets, et les relations qui peuvent être décrites entre objets, se retrouvent dans le vocabulaire utilisé par la représentation basée sur la connaissance [59].

Une ontologie basée sur le monde réel devrait intégrer les relations qui existent entre les entités réelles, classées selon la façon dont elles doivent être atteintes [5]. Les systèmes et services basés sur les connaissances sont coûteux à concevoir, à développer, à tester et à maintenir. Une spécification formelle des ressources partagées et réutilisables est nécessaire. La spécification du vocabulaire partagé a une place importante, parce que les différentes applications qui peuvent être développées autour d'une ontologie peuvent nécessiter des processus déductifs différents, ainsi qu'un langage spécifique. Par conséquent, il y a plusieurs défis à surmonter pour le développement de systèmes de gestion de bases de connaissances, partagés et réutilisables.

II.2.3)Les avantages de la modélisation déclarative

II.2.3.1- Le caractère intuitif

La modélisation déclarative permet d'aborder la conception d'une scène ou d'un environnement virtuel de manière intuitive. Le simple fait que l'utilisateur se consacre à la description de son objectif est un apport considérable par rapport à la modélisation géométrique impérative classique.

Le caractère intuitif, qui est la qualité essentielle d'un modeleur déclaratif, dépend directement des moyens donnés à l'utilisateur pour décrire une scène.

Si le modeleur offre à l'utilisateur la possibilité de décrire la scène grâce à l'énoncé de propriétés exprimées en langage naturel ou quasi-naturel, le vocabulaire autorisé aura une importance capitale dans la facilité d'expression [15][9]. Dans ce cas, le vocabulaire utilisera des notions d'un haut niveau d'abstraction, propres au domaine traité par le modeleur déclaratif, et connu dans la base de connaissances.

II.2.3.2- La vérification automatique des propriétés de la scène

Le modeleur déclaratif reçoit de l'utilisateur la description de la scène qu'il souhaite créer. Cette description met en avant les relations qui existent entre les éléments de la scène. Ces relations sont intégralement prises en charge par le modeleur déclaratif.

L'utilisateur n'a plus à se préoccuper de la vérification de ces propriétés, car c'est le modeleur qui s'en charge [31]. La solution retenue à la fin du processus vérifiera donc l'ensemble des critères de la description dans le cas où toutes les propriétés ont pu être prises en compte.

Dans le cas où des propriétés n'ont pu être prises en compte (propriétés contradictoires, ou ne pouvant être satisfaites) le modeleur pourra être en mesure d'indiquer à l'utilisateur les causes de ces impossibilités, pour qu'il puisse modifier sa description s'il le désire.

II.2.3.3- La manipulation des aspects non géométriques

Un modeleur déclaratif peut manipuler des aspects de la scène autres que les aspects purement géométriques ou graphiques. Le modeleur peut prendre en compte des propriétés liées aux conditions d'éclairage ou de température. Le poids d'un objet destiné à être intégré à une scène peut être également un critère pris en compte pour une relation entre plusieurs objets [15]. Ainsi on peut intégrer des connaissances liées à la prise en compte de critères physiques pour tenir compte de la résistance des matériaux des objets présents dans la scène, des frottements, etc.

II.2.3.4- L'adaptation au processus de conception

La modélisation déclarative est plus adaptée à la gestion intégrale du processus de conception. Un même objet peut être perçu différemment selon les acteurs de sa conception.
En effet, celui qui conçoit l'objet n'a pas nécessairement les mêmes préoccupations que celui qui le fabrique et que celui qui l'utilise. La modélisation déclarative peut prendre en charge l'ensemble du processus de création et arriver à ce que les différentes propriétés souhaitées par les différents intervenants pendant l'étape de conception soient vérifiées.

II.2.4)Les inconvénients de la modélisation déclarative

II.2.4.1- L'interprétation de la description

L'interprétation de la description est un problème important que pose la modélisation déclarative. La manipulation de concepts de haut niveau induit souvent une multiplicité d'interprétations possibles [12]. La notion d'interprétation est souvent subjective, et la plupart des modeleurs déclaratifs n'ont que très partiellement abordé le problème. Pourtant l'efficacité d'un modeleur déclaratif dépend de l'adéquation entre l'interprétation de l'utilisateur et celle de l'outil.

La plupart des modeleurs déclaratifs actuellement existant ne prennent pas en charge cette étape d'interprétation et considèrent en entrée un script descriptif de bas niveau d'abstraction, très proche des contraintes attendues par la phase de génération.
Cette étape d'interprétation est donc totalement occultée, sauf par des travaux qui ont, a contrario, fait porter leurs efforts sur elle, et n'ont pas cherché à créer des scènes d'une complexité approchant celle de scènes réalistes.
C'est le cas du projet WordsEye [9] qui est un système de type « text-to-scene » qui prend en entrée une description en langage naturel et qui s'appuie sur une base de connaissances très riche. L'interprétation consiste à déterminer pour chaque objet et chaque personnage présent dans la scène quelles sont les relations à prendre en compte à partir de la description : les positions relatives des entités et la façon de les utiliser quand il s'agit d'objets.
La base de connaissances permet à l'outil de déduire la position relative des entités et la posture des personnages en fonction de leurs actions.
Par contre, les scènes contiennent très peu d'objets et ces objets sont tous parallèles aux axes de la scène, comme le montre la scène de la Figure 4 ci-après.

Figure 4: Scène correspondant à la description : « Mary uses the crossbow. She rides the horse by the store. The store is under the large willow. The small allosaurus is in front of the horse. The dinosaur faces Mary. The gigantic teacup is in front of the store. The gigantic mushroom is in the teacup. The castle is to the right of the store. » [9]

II.2.4.2- L'obtention d'un grand nombre de solutions

La plupart des modeleurs déclaratifs utilisent l'exploration d'un univers, souvent fini, de formes possibles, pour rechercher celles vérifiant les propriétés données grâce à la description. Cette recherche peut conduire l'utilisateur vers des solutions qu'il n'envisageait pas [25]. Cependant, en fonction de la précision de la description (cas sous-contraint), il peut y avoir un nombre très important de solutions, bien au-delà de ce que l'utilisateur peut gérer de lui-même pour savoir faire la différence, car beaucoup de solutions peuvent paraître similaires.
Néanmoins la présentation de toutes les solutions possibles peut être un choix délibéré pour certains modeleurs, quel qu'en soit le nombre. C'est le choix qui a été fait dans les projets ExploFormes dirigé par Michel Lucas à Nantes [6][42].

II.2.5)Modeleurs déclaratifs

Pour continuer cette présentation de la modélisation déclarative, nous allons décrire quelques systèmes de génération de scènes virtuelles, représentant un panorama des modeleurs déclaratifs dans ce domaine. Nous préciserons leurs spécificités ainsi que leurs éventuelles limitations.

II.2.5.1- EAAS

EAAS, ou Environnement d'Aide à l'Aménagement Spatial, est un ensemble d'outils pour l'aide à l'aménagement spatial conçu et réalisé par Philippe Charman [7]. N'entrant pas totalement dans le cadre de la modélisation déclarative (orienté CAO[10] et solveur de contraintes) il peut y être rattaché, principalement à cause de la phase de génération, et par le niveau d'abstraction utilisé. L'élément principal d'EAAS est son solveur de contraintes, basé sur une extension des CSP[11], les Spatial-CSP. Ce modèle a été développé spécialement par l'auteur pour tenir compte des caractéristiques géométriques des problèmes d'aménagement spatial. Les méthodes développées dans EAAS, notamment une technique de filtrage (l'arc-cohérence semi-géométrique) ainsi que l'implantation d'un large choix d'heuristiques et de techniques de *backtrack* ou retour en arrière, garantissent une résolution efficace dans de nombreux cas. EAAS a été développé pour résoudre des problèmes d'aménagement spatial dans le plan (voir

[10] Conception assistée par ordinateur
[11] Problème de satisfaction de contrainte (Constraint Satisfaction Problem)

Figure 5), ce qui constitue une première limitation de cet outil par rapport à notre domaine où les scènes générées le sont plutôt dans l'espace. De plus les orientations gérées sont réduites à 0, 90, 180 et 270 degrés, ce qui limite le monde à des objets isothétiques[12], plus exactement à des objets dont la boîte englobante est un parallélépipède isothétique.
En effet, EAAS ne gère pas la géométrie des objets mais uniquement le placement de boîtes englobantes.

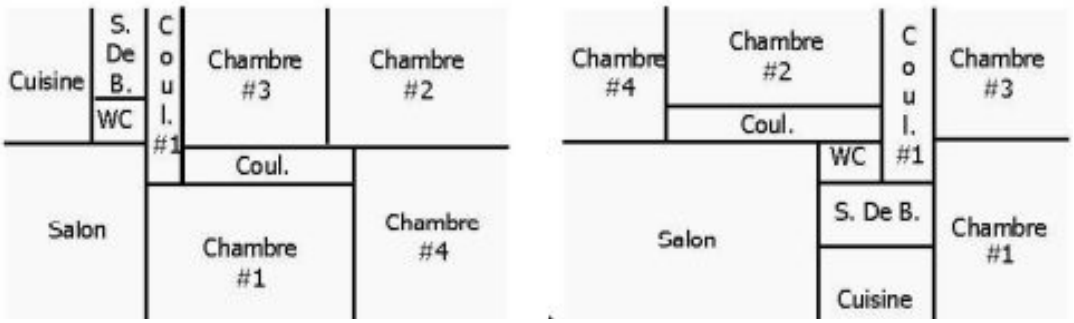


Figure 5: Résultats obtenus par EAAS [7]

II.2.5.2- DEM²ONS

DEM²ONS (DEclarative Multimodal MOdeliNg System) est un projet développé au sein de l'IRIT, sous la direction de Véronique Gaildrat [20] [21].
Ce projet a comme objectif l'étude de la modélisation déclarative. Il a débuté par la proposition d'un prototype de modeleur multimodal permettant à l'utilisateur d'énoncer oralement des requêtes telles que : « mets ça là » tout en désignant l'objet correspondant à l'énoncé « ça » et l'emplacement correspondant à l'énoncé « là ».
Ce prototype a été une première étape montrant l'importance d'un système de génération automatique, permettant de prendre en compte un ensemble de propriétés exprimées oralement ou non, avec un outil multimodal ou non.
Suite à cela les travaux ont essentiellement concerné l'étude de solveurs de contraintes dédiés à la modélisation déclarative.

II.2.5.3- ORANOS

ORANOS est un solveur de contraintes, destiné à la modélisation déclarative, et utilisé dans la phase de génération du modeleur. Conçu et réalisé par Ghassan Kwaiter [37], ce

[12] Parallèle aux axes de la scène

solveur s'appuie sur un modèle étendu de CSP : les DHNCSP (Dynamic Hierarchical Numeric Constraint Satisfaction Problems).

Basé sur les CSP numériques et l'analyse par intervalles, ce modèle introduit une hiérarchisation des contraintes afin de pouvoir automatiquement en relâcher certaines dans les cas sur-contraints (cas où le solveur ne trouvera aucune solution), et présente des caractéristiques très intéressantes pour un solveur dans le cadre de la modélisation déclarative :

- il utilise une gestion dynamique et incrémentale du système de contraintes afin de permettre son intégration dans une application interactive,
- il est basé sur une hiérarchie de boîtes englobantes afin de réduire la complexité des scènes (les objets placés dans des conteneurs différents s'ignorent au sens des contraintes).

Le principal inconvénient d'ORANOS est, comme pour EAAS, la restriction à un placement de boîtes englobantes isothétiques (voir Figure 6).

Figure 6: Résultat obtenu avec ORANOS dans DEM²ONS

II.2.5.3.1) MANHATTAN et Admunsen

Les solveurs développés par Olivier Le Roux pour sa thèse dans le cadre du projet DEM²ONS (DEclarative Multimodal MOdeliNg System) ont levé pour certains le problème du placement isothétique.

Celui de la restriction aux boîtes englobantes n'est par contre pas encore totalement résolu pour des problèmes d'efficacité.

En effet, une boîte englobante se limite à un parallélogramme et donc 6 facettes alors qu'un objet peut nécessiter des centaines voire des milliers de facettes pour être représenté avec un réalisme suffisant. Il faut donc faire un compromis entre la prise en compte de la géométrie réelle de l'objet et le temps de calcul qu'implique le traitement de cette géométrie, essentiellement lors des tests de non-collision.
Manhattan est une extension de EAAS de P. Charman, et également une amélioration de ORANOS, afin de générer des scènes plus réalistes et avec plus d'efficacité (voir Figure 7). Il place les boîtes englobantes selon 24 orientations isothétiques et permet d'orienter les objets librement à l'intérieur de leurs boîtes englobantes respectives [39].

Figure 7: Exemple de scène générée à l'aide du solveur MANHATTAN comprenant 28 objets

Admunsen est un solveur basé sur les CSP mais permettant une orientation quelconque des boîtes englobantes des objets présents dans la scène (voir Figure 8). Il tire parti du fait que les scènes sont généralement décrites de façon fortement sous-contraintes et qu'il existe de nombreuses solutions dans l'espace des solutions. Il utilise un principe de *pari stochastique*, évitant une recherche systématique lors de l'instanciation d'une solution à partir des contraintes initiales.
Ce solveur permet ainsi de gagner un facteur 10 en temps de calcul pour la plupart des scènes. Et quand le problème a peu de solutions, il revient à des techniques habituellement utilisées avec les CSP [41].

Ce solveur utilise des balises de placement (zones de l'espace associées à l'objet) à base de quadriques qui permettent un placement naturel des objets dans la scène en prenant en compte des descriptions imprécises (voir Figure 9).

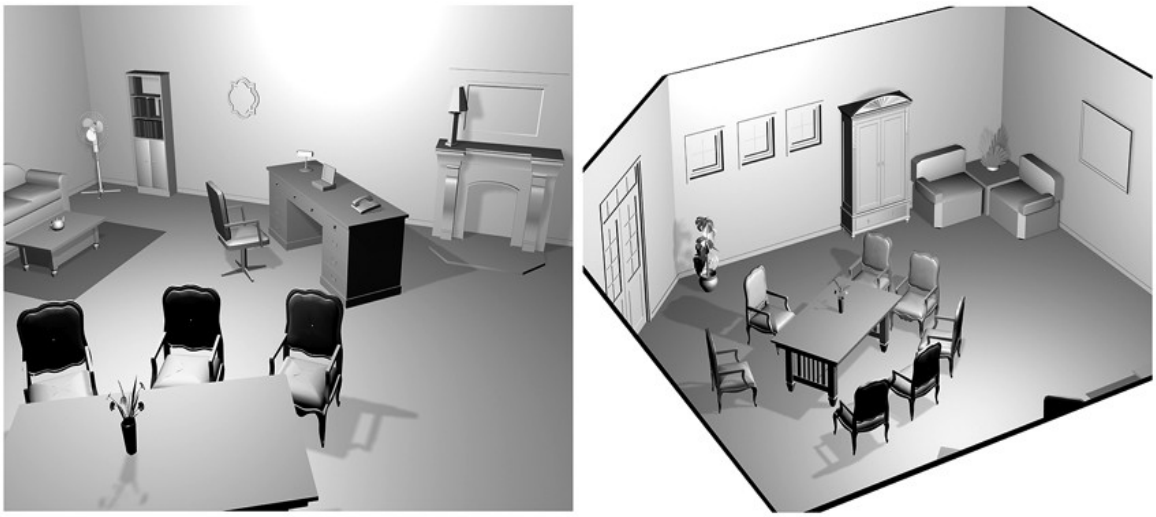


Figure 8: Exemple de scènes générées avec Admunsen illustrant le placement non isothétique.

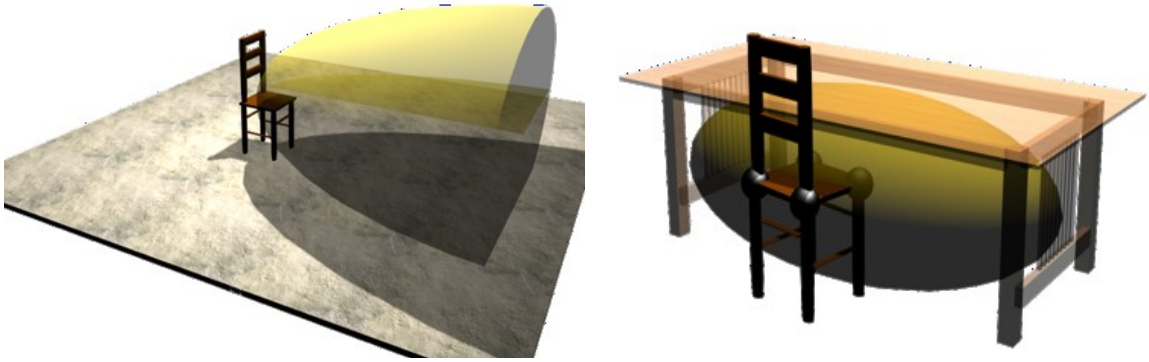


Figure 9: Balises de placement utilisées par le solveur Admunsen. L'image de gauche illustre la possibilité de placer un objet « devant la chaise » et l'image de droite illustre la possibilité de placer « la chaise contre la table ».

II.2.5.3.2) DEMONS_GA et DEMONS_LE

Les derniers solveurs étudiés dans le cadre du projet DEMONS sont basés, non plus sur des techniques complètes qui vont pouvoir explorer la totalité de l'espace de solutions pour trouver un optimum global, mais sur les métaheuristiques.

DEMONS_GA se base sur les algorithmes génétiques [43] (voir Figure 10) et DEMONS_LE se base sur le recuit simulé [63] (voir Figure 11).

Ainsi, ils peuvent traiter un grand nombre d'objets, mais ne peuvent assurer qu'ils trouveront une solution correspondant à un optimum global satisfaisant toutes les contraintes, même si cette solution existe.

Par contre, ces solveurs, basés sur des métaheuristiques, sont intéressants dans le cadre d'un modeleur déclaratif où les descriptions sont le plus souvent sous-contraintes, car ils peuvent être stoppés après un temps de recherche donné et fournir dans tous les cas une solution, même incomplète ou non optimale.

Figure 10: Scène générée avec DEMONS_GA comprenant 28 objets avec un placement non-isothétique et un temps de calcul de l'ordre de la minute.

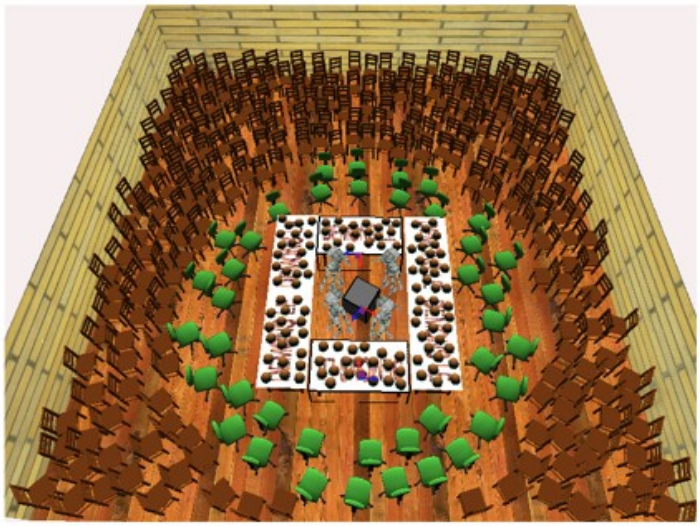


Figure 11: Scène générée avec DEMONS_LE comprenant plus d'une centaine d'objets, avec un placement non-isothétique et un temps de calcul de l'ordre de la minute.

Ces solveurs ont leur utilité dans le cadre de scènes où de nombreux objets non critiques sont à placer, mais sans que leur position exacte ait une grande importance.
Ainsi, ces solveurs peuvent être utilisés pour générer un espace vert où le nombre et la position exacte des arbres ne sont pas critiques, ou pour remplir une bibliothèque avec des livres, ou encore pour placer des personnes et simuler un effet de foule.

II.2.5.4- CAPS

CAPS [64], (Constraint-based Automatic Placement System) ou Système de placement automatique basé sur les contraintes, a été développé en 2001 par K. XU au sein du département Computer Science de l'Université de Toronto. CAPS s'attache à construire une seule scène de façon itérative en laissant l'utilisateur décider du placement des objets. Il agit ici plus comme une aide au placement interactif que comme un modeleur pleinement déclaratif. CAPS utilise de nombreuses méthodes pour faciliter la création de scènes complexes :

- implantation d'une "pseudo-physique", modèle physique simplifié (non prise en compte des accélérations et des vitesses) qui maintient une scène cohérente par rapport à la gravité,
- utilisation des sommes et différences géométriques de Minkowski pour accélérer le placement,
- utilisation de contraintes en deux dimensions, application de la modélisation déclarative,
- développement d'une base de données sémantique qui définit les placements autorisés (verre sur table mais pas l'inverse, assiettes sur le pourtour de la table mais empilées à la cuisine, etc.).

Les résultats que présentent K. XU [64] notamment ceux concernant les temps de génération de scènes relativement complexes (plusieurs centaines d'objets) représentent une référence à laquelle les modeleurs futurs devront se mesurer (voir Figure 12).
Néanmoins, ses travaux ne sont pas directement comparables aux travaux antérieurs à cause de l'absence d'un moteur de maintien de contraintes. Par exemple, si des objets ont été placés sur une table, le fait de déplacer cette table plus tard durant le processus de modélisation ne les conservera pas sur la table.
Dans un tel cas, un modeleur déclaratif doit pouvoir garder les contraintes préétablies satisfaites même en cas de déplacement d'un des objets liés par la contrainte.

Figure 12: Scène générée par CAPS comprenant 500 objets, créée avec à une assistance à l'interaction, en 25 minutes, placements et rectifications compris [64]

II.2.5.5- Système de reconstruction des scènes de crime

Nicholas Davies propose dans [11] une technique pour la génération d'environnements virtuels en 3D destinés à la reconstruction des scènes de crime. Cette application permet aux enquêteurs d'évaluer et d'éliminer plus rapidement les différentes hypothèses.

Beaucoup de travaux actuels se concentrent sur la production de scènes photo-réalistes, et sont considérés pour ce type d'applications trop complexes et trop coûteux. Ainsi, ce système offre une alternative moins coûteuse et plus rapide qui permet d'obtenir des scènes précises, construites et remplies avec des personnages animés présents sur le lieu du crime, avec des résultats très encourageants.

Bien que les outils de modélisation 3D produisent des résultats pouvant être hyper réalistes, il a été constaté que le coût des compétences techniques requises pour créer de telles scènes était si important qu'il faudrait disposer d'une équipe de spécialistes, ce qui est rarement possible. De plus, ces modeleurs sont difficiles à manipuler et consomment beaucoup de ressources en terme de traitement. L'objectif principal était donc de développer des méthodes plus rapides pour la création de scènes, afin de pouvoir les produire avec un minimum de connaissances et un matériel standard.

Cela afin de permettre aux témoins et aux victimes d'actes criminels d'obtenir rapidement des scènes à partir d'un outil simple à utiliser.

Concernant la méthodologie utilisée et la mise en œuvre, ce système permet la génération interactive de scènes de crime (voir Figure 13) grâce à un environnement graphique. La scène a été divisée en plusieurs plans contenant chacun une grille de cellules qui peuvent être remplies avec des objets provenant d'une bibliothèque d'éléments prédéfinis. Il y a un plan pour les murs, le plancher, le plafond, les objets et les personnages. Chaque élément peut être mis à l'échelle et orienté dans une direction quelconque. Quand une scène est créée, les premiers plans délimitant la scène sont mis à l'échelle. Enfin, les objets et les avatars sont placés dans le plan correspondant. Une fois que la scène a été créée dans l'environnement graphique, elle peut être modifiée et visualisée autant de fois que nécessaire jusqu'à ce que l'utilisateur soit satisfait des résultats.

Les principaux avantages de ce système sont que des scènes peuvent être créées et modifiées rapidement et efficacement, sans nécessiter un savoir-faire et un matériel coûteux.

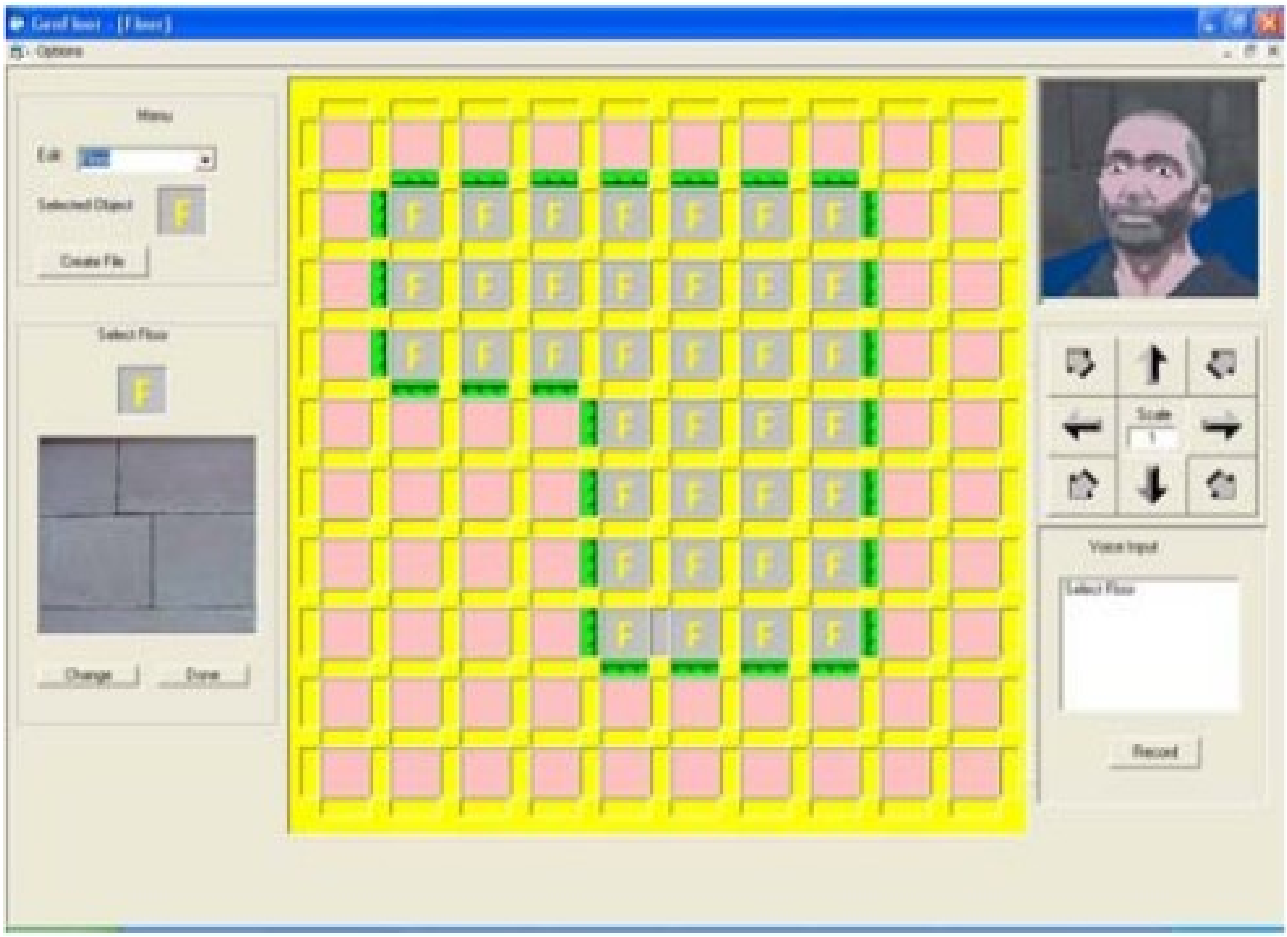


Figure 13: Interface principale du générateur des scènes de crime [11]

La scène pourrait facilement être développée à partir de croquis, ou d'une description fournie par les témoins. Des personnages animés peuvent être incorporés dans la scène, ce qui ajoute une dimension importante à des scènes statiques. Enfin, les scènes sont visualisées en temps réel.

Toutefois, ce système a quand même certaines limites (voir Figure 14). Les avatars sont forcément au format du logiciel 3D-Studio Max, et en créer un nouveau nécessite une bonne connaissance de cet outil ainsi qu'un certain sens artistique. Les animations sont limitées à un ensemble de déplacements déjà définis (rejoue une capture de mouvements). Aucune méthode de détection de collision n'a été utilisée. Le niveau de détails maximum dans la scène de crime est faible.

Les objets peuvent être placés dans une grille de 500 par 500 pixels, et le générateur ne permet de placer qu'un seul élément dans une zone de la grille. Cela limite la capacité de l'outil à placer au même endroit plusieurs petits objets, ce qui peut fausser la scène de crime. En outre, le créateur de la scène est incapable de générer des terrains ou des surfaces inclinés.

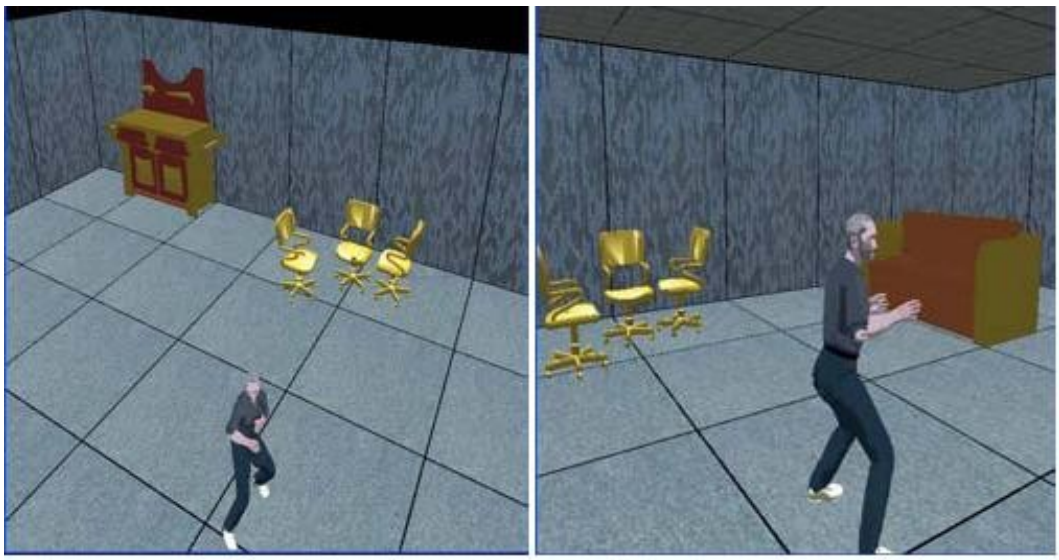


Figure 14: Exemple de scène de crime virtuelle [11]

II.2.5.6- MARINA

MARINA (Modeleur Architectural Interactif et Naturel) [34], est un outil qui permet d'acquérir de façon interactive une maquette numérique 3D à partir d'un dessin ou d'une image et d'un énoncé de contraintes. Le modèle peut être reconstruit à partir d'une ou plusieurs images.

Le processus incrémental peut se décomposer en trois étapes :

- le dessin et la saisie des contraintes,
- la reconstruction du modèle 3D,
- l'affichage et la manipulation de la scène.

Ces étapes peuvent être répétées afin d'affiner progressivement le modèle en ajoutant des éléments géométriques ou des contraintes.

Ce modeleur a été créé dans le cadre d'un projet associant historiens et architectes, et a comme objectif principal de reconstruire des bâtiments anciens à partir de documents graphiques d'archives (photographie, peinture, gravure, dessin, etc.).

Il permet d'obtenir des modèles en trois dimensions à partir d'un seul ou de plusieurs documents graphiques quelconques. L'approche utilisée permet d'améliorer les calculs de reconstruction en termes de vitesse, de précision et de simplification des cas à traiter (voir Figure 15).

L'interface utilisateur est simple, l'utilisateur dessine directement sur les images pour préciser ce qui doit être reconstruit et énonce des contraintes géométriques que le modèle doit respecter. Ces informations sont traduites dans l'algèbre de Grassmann-Cayley avant d'être traitées formellement. La rapidité et la simplicité de la reconstruction avec MARINA permettent d'utiliser ce système, non seulement comme modeleur, mais aussi comme outil de conception.

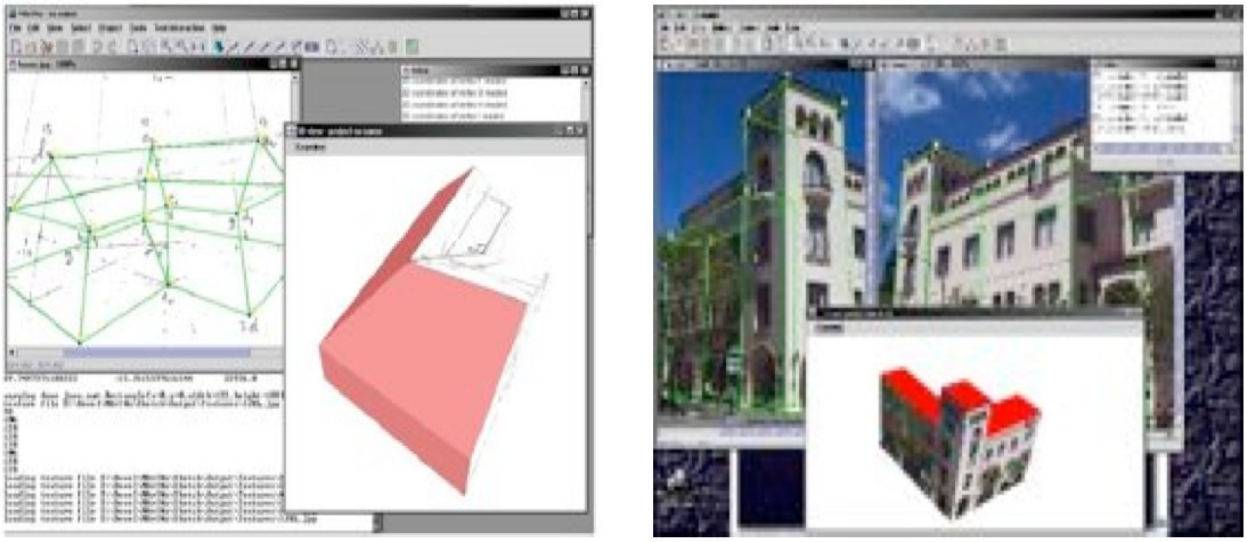


Figure 15: Conception d'un bâtiment avec MARINA [34]

II.2.5.7- GINA

Le projet Gina, ou Géométrie Interactive Naturelle, propose une nouvelle génération de modeleurs 3D [55][56] qui présente les caractéristiques suivantes :

Dans la phase de conception, le papier est utilisé pour faire des croquis, des esquisses, des organigrammes, etc.

Le dessin ne permettant pas de capturer pas toutes les caractéristiques de l'objet, il est nécessaire d'énoncer les propriétés qui complètent sa description.

Dans GINA, l'opérateur utilise donc simultanément :

- Le dessin : croquis à main levée, en perspective, en plan, en coupe,
- Le langage naturel pour exprimer une propriété (angle, coplanaire, orthogonal, parallèle, etc.), une action (déplacement, perspective, etc.) ou encore une dimension sur un croquis purement qualitatif.

Le système doit être capable d'interpréter des déclarations de trois niveaux :

- définitions de formes géométriques simples (plan, parallélépipède, cylindre, etc.),
- contraintes géométriques élémentaires permettant de préciser indirectement l'objet (la sphère n'est plus définie par son centre et son rayon, mais "touche telle sphère, tel plan", "ce plan est orthogonal à cette droite et passe par ce point"),
- contraintes de haut niveau sémantique traduisant le cahier des charges imposé à l'objet (propriétés d'usage, etc.).

Gina est aussi fondé sur la géométrie projective et utilise l'algèbre de Grassmann-Cayley [18]. GINA permet d'intégrer deux aspects essentiels au processus de conception : l'esquisse 2D (vues en projection d'un bâtiment 3D) grâce au dessin à main levée et à la résolution des contraintes géométriques.

II.2.5.8- CarSim

CarSim est aussi un outil de visualisation et d'animation de scènes en trois dimensions à partir de descriptions textuelles [16]. Il s'applique aux accidents de la route et est utilisé grâce aux constats fournis par les compagnies d'assurance. Les constats sont supposés être à l'amiable et les deux conducteurs se sont mis d'accord sur les descriptions de l'accident.

Les descriptions sont présentées sous la forme de courts textes écrits par un des conducteurs après l'accident. Elles correspondent à des incidents relativement simples bien qu'elles soient difficiles et parfois impossibles à interpréter.

L'objectif de CarSim est de reconstituer ces accidents dans un environnement virtuel sans chercher un réalisme du rendu de la scène. L'architecture du système comporte deux parties.

La première partie de CarSim est un module linguistique qui extrait l'information à partir du constat et remplit les éléments d'information de la fiche. La deuxième partie est un modeleur qui prend comme entrée la fiche remplie, crée les entités visuelles et les anime.

Figure 16: Reconstitution de l'accident dans CarSim [16]

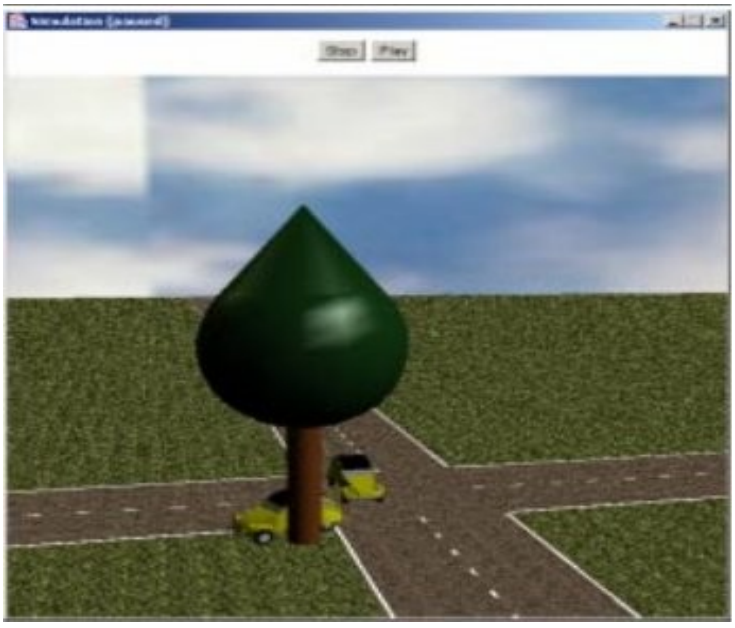


Figure 17: Suite de la reconstitution de l'accident dans CarSim [16]

Exemple de constats rédigés par un conducteur (voir Figures 16 et 17) :
Description *: « Véhicule B venant de ma gauche, je me trouve dans le carrefour, à faible vitesse environ 40 km/h, quand le véhicule B, percute mon véhicule, et me refuse la priorité à droite. Le premier choc atteint mon aile arrière gauche. Sous le choc, et à cause de la chaussée glissante, mon véhicule dérape, et percute la protection métallique d'un arbre, d'où un second choc frontal ».*

II.2.5.9- MultiFormes

MultiFormes est un projet basé sur la modélisation déclarative par décomposition hiérarchique. Il a été développé sous la direction de Dimitri Pléménos [47], et est à l'origine de travaux au laboratoire MSI[13] de Limoges, tant au niveau de l'expression des contraintes [48] que celui de l'interfaçage avec des modeleurs géométriques [53]. Une plateforme de conception, appelée MultiCAD, a été étudiée avec la collaboration de chercheurs d'Athènes [62]. Cette plateforme permet au concepteur de travailler soit sur la représentation graphique du modèle interne (arbre hiérarchique), soit sur la liste des règles de type Prolog qui lui correspond, puis de visualiser les diverses solutions volumiques engendrées par MultiFormes.
La modélisation déclarative par décomposition hiérarchique, développée dans MultiFormes [47] permet de travailler à plusieurs niveaux de détails, suivant en cela une démarche de projet architectural qui demande des représentations graphiques allant du 1/500ème au 1/50ème, voire au 1/10ème.
Une scène pourra être décomposée en plusieurs sous-scènes qui sont contrôlées par des propriétés locales (par exemple, taille ou forme) et/ou des propriétés inter-scènes (placement des sous-scènes les unes par rapport aux autres). En architecture, les scènes sont composées d'espaces habitables, qui ont des propriétés spécifiques. Les murs, les toits ou les cloisons font partie du vocabulaire de construction.
La décomposition hiérarchique d'une scène est représentée par un arbre dont les espaces sont les feuilles terminales (voir Figure 18). Lors de la décomposition hiérarchique, des nœuds intermédiaires peuvent être créés, comme ici *Coin-jour* et *Coin-nuit*.

[13] Méthodes et Structures Informatique

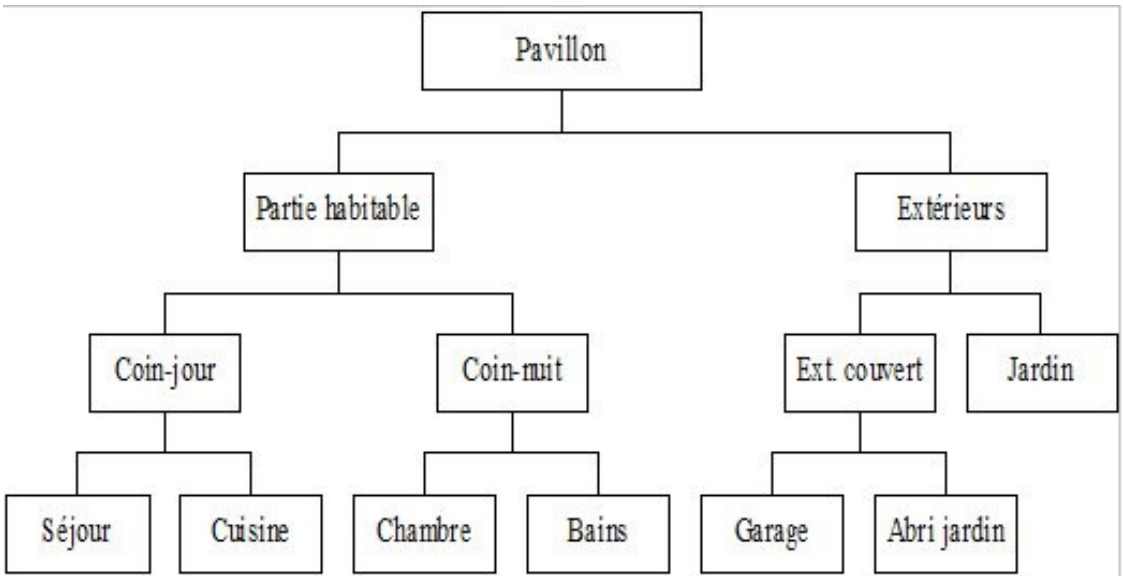


Figure 18: Extrait d'arbre hiérarchique MultiFormes [47]

Le modèle interne issu de la description est une collection de règles de type Prolog (voir Figure 19).

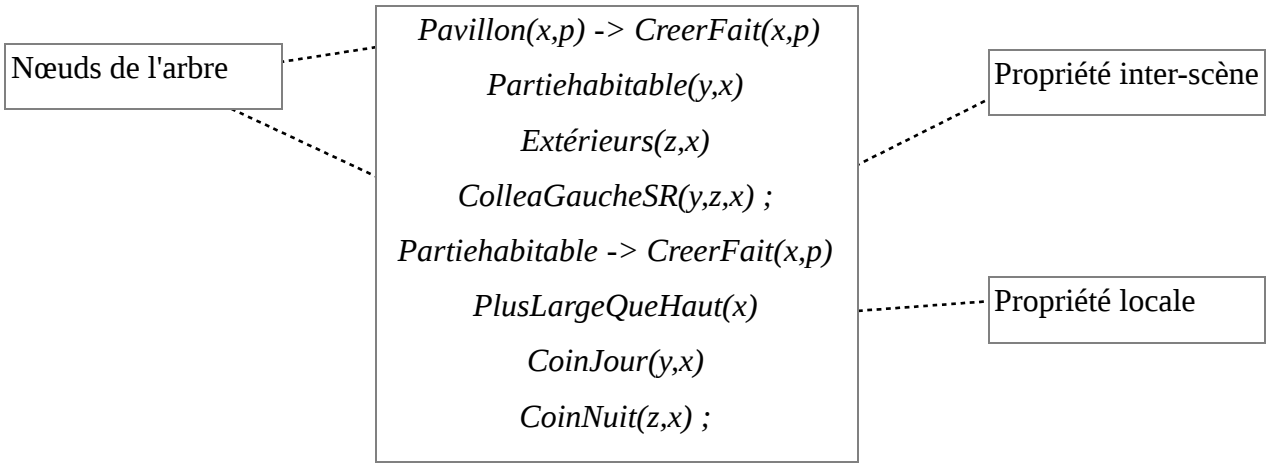


Figure 19: Extrait de modèle interne MultiFormes [47]

La description d'une scène se fait par raffinements successifs et accepte une description incomplète de ses composants. C'est même là son grand intérêt : pour une sous-partie de la scène qui ne serait pas entièrement définie, plusieurs solutions seront proposées, qui permettent alors au concepteur de faire des choix (voir Figure 20).

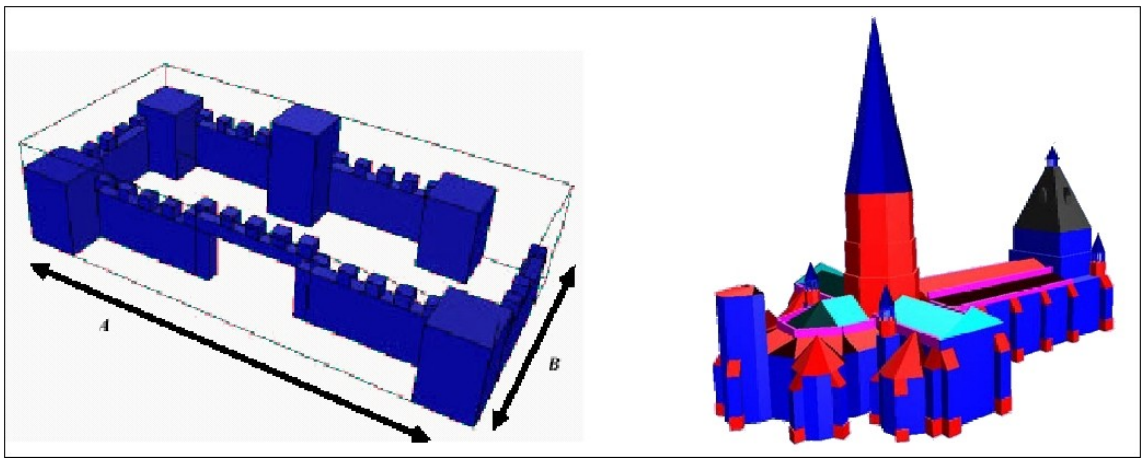


Figure 20: Exemples de solutions proposées par Multiformes [48]

L'outil de description architectural proposé comporte trois volets :

- la description des espaces,
- la description des propriétés,
- la description du projet lui-même.

II.2.5.9.1) Description des espaces

Les espaces habitables sont modélisés sous forme de nœuds terminaux, auxquels sont attachées certaines caractéristiques. Les architectes ont l'habitude de manipuler des espaces auxquels ils associent des dimensions, une orientation, une fonctionnalité, des équipements, etc. L'ensemble des informations regroupant ces caractéristiques correspond à la connaissance implicite que met en œuvre le concepteur à la phase de l'esquisse. Il n'est question ici que de modéliser une petite partie du savoir ou de l'expérience d'un architecte.

Dans un premier temps un dictionnaire est créé, répertoriant les espaces habituellement rencontrés. Pour chaque espace des seuils minimaux sont fixés par l'architecte, et sont modifiables pour chacun des projets. Ces seuils correspondent à :

- une surface minimum,
- une largeur minimum de linéaire en façade,
- une largeur et une longueur d'espace minimum, pour garantir la fonctionnalité de l'espace,
- une hauteur minimum sous plafond.

La description des espaces répertoriés est ensuite complétée par une série de propriétés spécifiques.

II.2.5.9.2) Description des propriétés

Les propriétés utilisées par MultiFormes portent sur :

- La forme du bâtiment : PlusHautQueLarge() ou HautArrondi().
- Une propriété de type Collé....() comme CollléAGauche() contraint un espace à ouvrir sur une façade particulière. ColléEnFaçadeOuest(), impose une ouverture sur la façade Ouest.
- La communication des espaces entre-eux :

  Les espaces habitables communiquent entre-eux par l'intermédiaire d'ouvertures sur des frontières communes (murs ou cloisons), ou de façon moins directe, par un couloir ou une cage d'escalier. Chaque cas correspond à une propriété de collage.
- Le positionnement par niveau ou altitude :

  Certains espaces comportant des accès (entrée, garage) ont des altitudes fixées par la configuration du terrain.
- L'équipement des espaces :

  L'architecte connaît quelquefois de façon très précise l'équipement d'un espace : c'est le cas pour la salle de bains ou la cuisine. Si les dimensions relatives aux équipements sont connues, la vérification grossière de la validité des surfaces des espaces pourra être effectuée avant la génération, de façon à déceler les situations de blocage.
- Les formes :

  Les formes d'un espace peuvent être décidées a priori, et l'ensemble des propriétés actuelles portant sur la forme sera complété par des informations sur la toiture afin de correspondre aux besoins de la profession.
- L'organisation des espaces entre-eux :

  Des contraintes de symétries, d'articulation autour d'axes, de cheminement complètent au mieux les outils de description de contraintes mis à la disposition des architectes utilisant MultiFormes.

II.2.5.9.3) Bilan

Cet outil est limité par le champ étroit du domaine concerné : la modélisation d'un seul type d'habitat individuel ne correspond qu'à une faible part des activités d'un architecte, mais a l'avantage d'être fortement contraint et codifié. L'ambition des auteurs a été de pouvoir, à terme, assister le concepteur dans l'élaboration de projets de grande envergure.

La visualisation des volumes sous MultiFormes peut se faire en présentant le meilleur point de vue à l'utilisateur [13].

Enfin, ce projet intègre la possibilité, développée dans le cadre de MultiCAD, de proposer des solutions types pour tout ou partie de la description [12].

II.2.5.10- Modeleurs déclaratifs de terrains

La modélisation de milieux extérieurs est aussi un cas très fréquent en modélisation déclarative.

Un cas particulier a été spécialement étudié par R. Bidarra et présenté dans [54]. Il concerne la génération de terrains de simulation de combat, dans le domaine militaire. En effet, les jeux de simulation sont de plus en plus utilisés dans les formations militaires, mais pour être pleinement utilisables et paramétrables par les instructeurs, il faut pouvoir offrir un moyen simple et efficace de créer de nouveaux terrains d'entraînement, accessible à des personnes non spécialistes des outils de modélisation. R. Bidarra a ainsi développé un modeleur déclaratif de terrains, qui permet aux concepteurs de créer une description à l'aide d'un éditeur de croquis adapté à cette tâche. Ainsi, à partir d'une interface simple et conviviale, les instructeurs peuvent entrer directement les différentes propriétés et paramètres qui caractérisent le terrain, et ils obtiennent, après sélection parmi une grande variété de choix possibles, les éléments et l'emplacement des dispositifs de défense qui vont composer le terrain d'entraînement ou l'environnement de combat.

II.2.5.10.1) Méthode de génération

Les outils généralement utilisés en modélisation de terrains, que ce soit dans le cadre des *serious games* ou pour d'autres applications, utilisent principalement deux approches :

- Une approche entièrement manuelle : l'interface propose plusieurs modèles prédéfinis et intégrés dans le jeu ; l'utilisateur n'a qu'à choisir les éléments qu'il souhaite ajouter dans la scène, et les placer manuellement. Ceci requiert des compétences très élargies sur l'utilisation de l'outil de modélisation de la part de l'utilisateur. De plus, ces outils offrent rarement la possibilité de créer ses propres objets mais imposent de toujours travailler à partir d'une base d'objets existante.
- Une approche automatique : basée sur la modélisation procédurale et ne nécessitant pas de connaissances en modélisation 3D. La capacité à générer automatiquement une grande quantité de contenu est son principal avantage. Mais l'utilisation de ces techniques procédurales reste encore trop difficile à prendre en

main, notamment au niveau de la complexité des interfaces graphiques. Les L-systèmes ont notamment été utilisés par Parish et Muller [46] dans le projet City Engine, pour générer des villes à partir de données géographiques réelles (cartes hydrographiques, routières, d'élévation, de population, de végétation), et de schémas urbains prédéfinis (voir Figure 21).

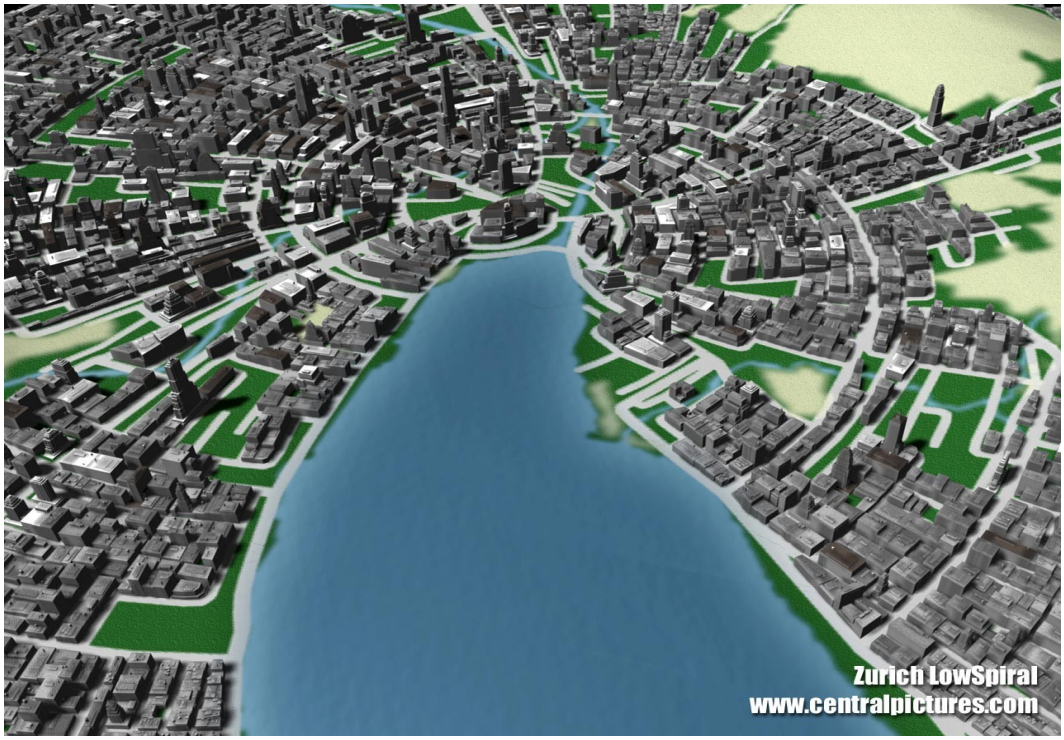


Figure 21: Ville générée grâce à City Engine sut le terrain de Zurich mais avec un schéma radio-centrique [46]

II.2.5.10.2) Principe

Pour remédier aux problèmes liés aux principes mêmes de ces méthodes, R. Bidarra a proposé une nouvelle approche pour la génération de terrain [54]. Le but principal de cette approche n'est pas d'accélérer l'étape de calcul du modèle de terrain, mais de fournir un moyen simple et accessible à toute personne désireuse de créer des modèles de terrain répondant à un ensemble de critères (voir Figure 22). Pour cela, un outil de modélisation répondant à cette demande a été conçu et développé. C'est seulement à partir de l'interface simple d'utilisation proposée par cet outil que les utilisateurs vont intervenir pour exprimer leurs souhaits et obtenir en échange le résultat de la génération.

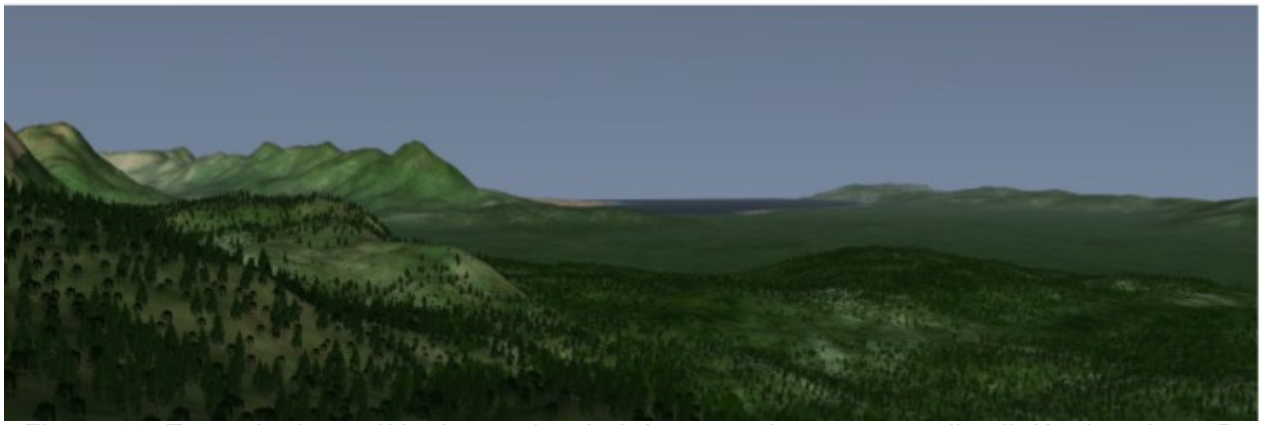


Figure 22: Exemple de modèle de terrain généré automatiquement par l'outil développé par R. Bidarra [54]

Pour arriver à un tel résultat, le processus de modélisation repose sur les étapes suivantes :

- l'utilisateur esquisse grossièrement le terrain en indiquant les zones de forêts, les cours d'eau, les zones urbaines,
- l'outil passe ensuite à la phase de génération proprement dite du terrain répondant aux critères généraux de l'utilisateur,
- l'utilisateur peut ensuite éditer le résultat et modifier à sa convenance le croquis initial ou le résultat généré avec une plus grande précision.

Cette modélisation est donc itérative, et l'utilisateur a la possibilité de modifier le croquis ou le résultat pour ajouter ou enlever des détails, tant qu'il n'est pas satisfait du résultat.

Au final, il peut exporter automatiquement le résultat vers un modèle 3D de terrain pour des usages ultérieurs.

Concernant la structure du terrain, il est décomposé en cinq couches permettant d'obtenir cinq niveaux différents de croquis (voir Figure 23) :

- le sol,
- les zones humides (cours d'eau, lacs, mer),
- les zones de végétation (forêt, cultures),
- réseau routier,
- zones urbaines.

En effet, l'utilisation de ces différentes couches permet d'améliorer l'adaptabilité du terrain, car les modifications effectuées sur une couche sont répercutées dans les autres couches.

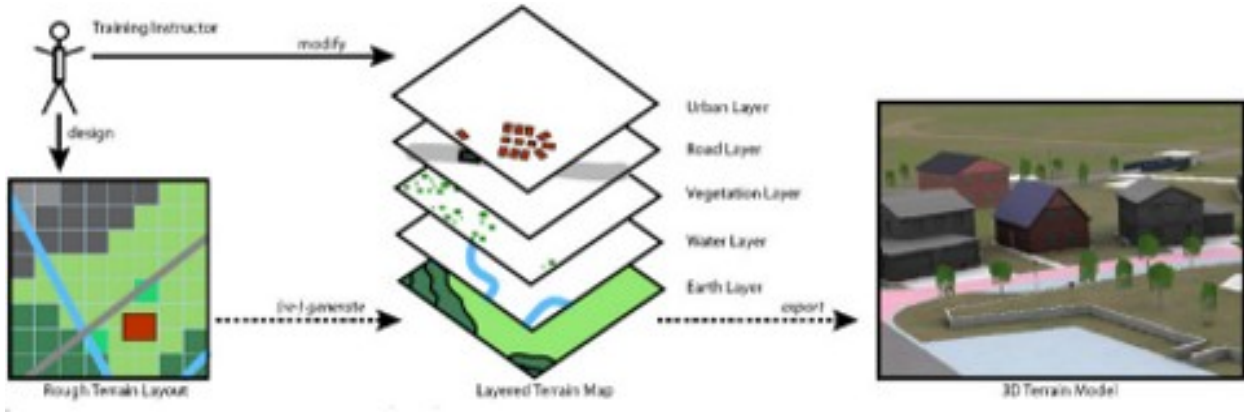


Figure 23: Processus de modélisation déclarative du terrain [54]

II.2.5.11- Bilan

Les modeleurs déclaratifs existants présentent une grande diversité, qui est en grande partie due à la difficulté à concevoir un modeleur à la fois génétique et efficace. Malgré cela, à cause de la relative jeunesse du domaine, de nombreuses voies de recherche n'ont pas encore suffisamment été explorées, notamment, l'étape initiale qui permet à l'utilisateur d'énoncer la description qui sera ensuite prise en charge par un système de génération.

Néanmoins, les techniques de résolution de problèmes utilisées par les modeleurs déclaratifs sont également utilisés par d'autres systèmes, dont l'objectif est de générer des scènes à partir de descriptions entièrement textuelles.

Nous allons voir dans la section suivante certains de ces systèmes, afin de voir quels en sont les principes les plus intéressants.

II.3- Système de conversion de texte en scène

L'imagination est essentielle pour les êtres humains. Lors d'une conversation, nous sommes en mesure d'imaginer des lieux et des événements à partir de la description donnée par notre interlocuteur.

Il en est de même lorsque nous lisons des livres qui ne contiennent pas d'illustrations et qui décrivent un environnement ou le lieu d'un événement. Certaines personnes peuvent avoir un imaginaire plus fertile que les autres, mais chaque individu a besoin de laisser aller son imagination dans sa vie quotidienne.

Les ordinateurs par contre n'ont pas d'imagination. Ils ne peuvent effectuer que les tâches qu'on leur a assignées, et sont totalement incapables de créer ou d'inventer quelque

chose qu'on ne leur a pas fourni auparavant. Donc, sauf si on leur dit précisément comment traduire un événement ou générer un lieu, ils ne sauront pas le faire.

Les systèmes de conversion de texte en scène, que nous notons « TTS », visent à donner l'illusion de l'imagination aux ordinateurs. Lorsqu'on entre une description, un TTS va convertir cette description en une scène statique ou animée.

Les TTS ont de nombreuses applications :

- Les utilisateurs qui ne sont pas familiarisés avec l'utilisation d'outils de modélisation et de création d'animations peuvent, à l'aide de ces outils, convertir des scènes imaginaires en scènes 3D animées.
- Les récits qui n'existent que sous forme textuelle pourraient être visionnés comme un film d'animation.
- De plus, un TTS peut permettre à plusieurs personnes de visionner une situation qui n'existe que sous forme écrite. Ceci pourrait être très utile, par exemple, dans la situation d'accident de voiture [16] ou pour visionner des consignes de sécurité en cas d'incendie dans un bâtiment.

La conception et la mise en œuvre de tels systèmes est très complexe. Les descriptions sont pleines de non-dits subtils que les gens prennent pour acquis car faisant partie de leur expérience commune, mais qui deviennent difficiles à prendre en compte quand ils doivent être interprétés par un ordinateur.

Cette partie permettra de discuter la façon dont trois TTS gèrent certains de ces défis. Les principaux domaines étudiés seront les langages d'entrée de ces systèmes, et la gestion des relations spatiales.

Ce ne sont pas les seuls défis auxquels sont confrontés ces systèmes, mais ce sont ceux qui semblent les plus intéressants dans le cadre de notre projet.

Après l'analyse de la façon dont chaque système traite ces problèmes communs, il y aura une conclusion. Cette conclusion va inclure quelques observations finales sur chaque système. Elle proposera également quelques évolutions possibles pour les TTS.

Les TTS actuels s'appuient sur plusieurs projets antérieurs. De ce fait, de brèves descriptions de certains systèmes, parmi les plus récents, sont incluses.

II.3.1.1- Contexte

Les systèmes de conversion de texte en scène (TTS) sont confrontés à une multitude de questions. Cependant, il reste un problème qu'on ne peut jamais résoudre de façon

satisfaisante car chacun a sa propre version imaginaire de la scène finale. Ainsi, aucun TTS ne va être capable de générer des scènes qui vont correspondre exactement à l'imagination de chaque lecteur. En fait, il peut même ne correspondre à l'imagination d'aucun des lecteurs. Toutefois, s'il vérifie toutes les contraintes définies dans la description donnée, alors on peut estimer que le TTS a rempli la tâche pour laquelle il a été créé.

Dans la plupart des cas, ces descriptions doivent être formatées suivant des règles prédéfinies, selon les contraintes et l'outil de visualisation en 3D.

II.3.1.2- Langage d'entrée

Le choix du langage d'entrée est une étape très importante et nécessaire lors de l'élaboration d'un système de conversion de type texte-en-scène (TTS). Le langage d'entrée représente la façon dont les descriptions seront présentées au système.

Dans certains des travaux présentés précédemment, le langage d'entrée est dans une forme fixe (comme avec la terminologie de Vandeloise [61]). On pourra ainsi exprimer les propriétés sous la forme :

« *Verbe* **Objet** *Relation* Point-de-référence »

Par exemple : « *Mettre* **la chaise** *à côté de* la table ».

Certains systèmes n'utilisent pas de règle ou de langage d'entrée prédéfini, mais proposent un style narratif proche du langage naturel. Ceci permet à un utilisateur de décrire une scène entière de manière plus aisée, plutôt que de tenter de transcrire un texte, comme celui d'une pièce de théâtre, selon un schéma prédéfini tel que la terminologie de Vandeloise.

Mais même quand les systèmes ne font aucune restriction explicite sur le langage d'entrée, il semble y avoir au moins une restriction implicite. Chacune des descriptions se fait selon des formes similaires. Les phrases sont la plupart du temps simples, et ont toujours une organisation respectant la forme suivante :

sujet (verbe de position| verbe d'existence) (objet | phrase prépositionnelle).

II.3.1.3- Système WordsEye

Alors que les modeleurs déclaratifs sont issus de la recherche universitaire, WordsEye [9] a été développé dans l'un des laboratoires de recherche de AT&T[14]. Ceci se traduit directement par la taille des équipes impliquées.

[14] American Telephone & Telegraph

Alors que les modeleurs universitaires sont le résultat du travail d'un étudiant pendant deux (projet CAPS) ou trois ans (projet Manhattan), WordsEye est le fruit du travail d'une dizaine d'ingénieurs pendant plus d'un an (qui a donné lieu à l'article présenté à SIGGRAPH 2001 [9]) ce qui représente un investissement plus que conséquent.

WordsEye est basé sur une description textuelle, associée à des informations sémantiques, pour générer une scène à partir de la description textuelle (il s'agit d'un système text-to-scene). Le texte fourni par l'utilisateur est ensuite interprété de façon à produire une représentation sémantique. Celle-ci est ensuite traduite en un ensemble de descripteurs, relativement à un ensemble de règles, permettant le calcul de la position d'objets 3D dans une scène en respectant au mieux la description fournie. L'exemple, présenté dans Figure 24, correspond à la description :

« *le canapé est placé contre le mur en bois. La fenêtre est placée au mur. La fenêtre est à côté du canapé. La porte est à 2 mètres à droite de la fenêtre. L'homme se trouve à côté du canapé. Un mur avec des animaux est à droite du mur en bois. Le mur avec des animaux est en face du mur en bois. Le mur avec des animaux est tourné vers la gauche. La table basse à peau de zèbre est à deux mètres en face du canapé. La lampe est sur la table. Le sol est brillant.* »

Figure 24: Exemple de résultat obtenu avec WordsEye [9]

Cette technique, qui donne des résultats remarquables, repose sur deux parties essentielles :

- une analyse linguistique du langage naturel, permettant l'interprétation des descriptions fournies par l'utilisateur (on note l'utilisation de WordNet[15] comme système de référence lexical [19]),
- une base de connaissances extrêmement riche (et de conception très coûteuse) contenant des objets 3D associés à des caractéristiques qui permettent de les positionner dans leur contexte, de façon à respecter leurs fonctionnalités (couleur, attitudes, relations spatiales, etc) et leur usage (posture de l'utilisateur de l'objet).

Il n'y a pas d'animation dans WordsEye, certains mouvements peuvent être représentés par l'utilisation de poses (de façon statique).

La première étape consiste à marquer (baliser) et analyser le texte fourni par l'utilisateur. L'arborescence obtenue à l'issue de cette analyse est convertie en une structure de dépendance, qui répertorie les mots dans la phrase et montre les dépendances.

Ensuite, la structure de dépendance est convertie en une représentation sémantique. Pour un mot donné, comme « camion », WordNet est consulté afin de détecter tous les hyperonymes[16] et hyponymes[17].

WordsEye a associé leurs modèles 3D avec la base de données WordNet, alors quand le mot « camion » est retourné, un modèle 3D l'accompagne. Les verbes sont traités de façon similaire. En tout, les auteurs affirment qu'ils ont des entrées pour 1300 objets et 2300 verbes substantifs, ce qui permet de gérer au mieux un vocabulaire particulièrement riche.

II.3.1.4- Système de Kevin Glass

Dans [25], Kevin Glass et Shaun Bangay ont présenté leur nouveau système de génération de scène 3D à partir d'une description textuelle. L'objectif principal de ce système est d'analyser un récit de fiction et de produire une représentation graphique 3D multimodale. Cela comprend non seulement les animations issues du texte, mais aussi l'ajout du son, y compris les voix des différents personnages dans la scène.

La première étape est la génération des animations 3D à partir d'un texte de fiction annoté et la traduction automatique des annotations en contraintes. Les contraintes sont ensuite résolues à l'aide d'une technique basée sur l'analyse d'intervalles.

---

[15] Base de données lexicale, développée par des linguistes du laboratoire des sciences cognitives de l'université de Princeton.

[16] Un hyperonyme est une catégorie générale regroupant des sous-catégories.

[17] Un hyponyme a le sens inverse de l'hyperonyme et correspond à un mot dont le sens est plus spécifique que celui d'un autre.

La description initiale de la scène est utilisée pour instancier les contraintes, afin de fournir un environnement virtuel 3D animé. Le principe de conversion de texte utilisé dans ce système comprend les étapes suivantes :

- annotation des descriptions de la scène, permettant de repérer les termes utiles et éventuellement d'enrichir le texte initial avec d'autres informations jugées nécessaires,
- interprétation de la description de la scène.

L'annotation du texte initial permet d'obtenir une représentation intermédiaire de la scène finale. Cette représentation est enrichie avec les catégories de balises suivantes :

- avatar : pour identifier les personnages dans la scène,
- objet : pour repérer un objet ou accessoire,
- caractéristiques : pour définir l'environnement,
- relation : pour spécifier les relations spatiales,
- transition : pour marquer le début ou la fin d'une séquence d'animation.

La Figure 25 montre un exemple de scène obtenue à partir de la description suivante : « He stole <transition type='inside' subject='he'>in</transition>. His<avatar>uncle</avatar> still snored. He tiptoed by him <relation type='near' subject='he' object='table'>to</relation> the<object>table</object> <relation type='behind' object='chair'>behind</relation> his uncle's <object>chair</object> »

Figure 25: Résultat obtenu à partir du système développé par Kevin Glass [25]

II.3.1.5- Conclusion

Le système développé par Kevin Glass présente une approche plus ambitieuse que celle mise en œuvre par WordsEye pour le traitement des textes descriptifs. Ces deux systèmes prennent en entrée des textes qui n'ont pas forcément été écrits pour un objectif de création de scènes 3D.

Mais en pratique, on observe que les textes en entrée de WordsEye ont été écrits spécifiquement pour WordsEye, et doivent respecter un format imposé.

Les systèmes TTS autorisent l'utilisateur à modifier les descriptions fournies en entrée afin de corriger la scène résultante. L'utilisateur peut modifier ou reformuler la description dans un processus en spirale qu'on retrouve dans tous les modeleurs déclaratifs, mais il n'est pas certain d'obtenir un résultat correspondant à une simple évolution du résultat précédent. En effet, rien n'assure que deux solutions, calculées par un solveur de contraintes à partir de deux descriptions peu différentes, seront forcément proches dans l'espace de solutions.

Le système développé par Kevin Glass a un avantage significatif, car il est capable de traiter sans restriction des textes sans qu'ils soient ciblés sur un domaine spécifique. L'objectif du système de Kevin Glass est en effet de traiter toute forme de texte de fiction, ce qui peut englober un très large domaine d'application.

Par contre, le système de Kevin Glass traite des annotations uniquement géométriques et n'introduit aucune information sur l'état émotionnel des avatars, alors que WordsEye permet de prendre en compte certaines expressions même si elles sont rendues grâce à un affichage sommaire.

II.4- Moteur Physique

II.4.1)Réalité virtuelle immersive

La réalité virtuelle immersive (RV) suppose que l'utilisateur puisse évoluer et agir dans une simulation du monde (réel ou imaginaire) au sein d'un environnement virtuel numérique. Elle est fonctionnellement décomposable en deux composantes.

La première composante, que nous nommerons *moteur physique*, est assimilable à un système logiciel. Le moteur physique a pour fonction de régir en temps réel l'évolution des entités de l'environnement virtuel ainsi que leurs interactions, selon les lois de la physique.

Deux types d'interactions peuvent être identifiés : les interactions liées aux contraintes cinématiques, assimilables à des liaisons mécaniques (pivot, frottement, etc), et celles inhérentes aux collisions entre objets.
La seconde composante d'une plateforme de réalité virtuelle consiste en l'interfaçage entre l'environnement virtuel et le ou les utilisateurs participant à la simulation. Il existe plusieurs types d'interfaces, chacune destinée à la stimulation d'un des sens des utilisateurs : on peut citer les interfaces visuelles, auditives et haptiques.
Dans nos travaux, nous nous sommes intéressés essentiellement aux moteurs physiques, car les problèmes liés aux interfaces hommes-machines devraient intervenir dans une phase ultérieure de développement du projet DRAMA.

II.4.2)Définition

Un moteur physique est un ensemble de logiciels indépendants appliqués à la résolution de problèmes de la mécanique classique. Les résolutions les plus souvent rencontrées concernent les collisions, la chute des corps, les forces, la cinétique, etc. Les moteurs physiques sont principalement utilisés dans des simulations scientifiques pour la visualisation des phénomènes physiques. Dans ce cas, un moteur physique va mettre en œuvre diverses fonctions mathématiques afin de modéliser au mieux les phénomènes physiques du monde réel. On lui attribue également le rôle de gestionnaire de collisions c'est-à-dire que c'est également le moteur physique qui mettra en œuvre des calculs géométriques afin de déterminer si deux corps sont en collision ou non.

II.4.3)Rôle

Le rôle principal d'un moteur physique est donc de fournir au minimum des fonctions de détection des collisions. Il fournira également des outils permettant de calculer la réponse aux éventuelles collisions. Cette réponse se fera généralement sur la base des lois de Newton de la mécanique classique. Les moteurs physiques les plus évolués sont des moteurs permettant un nombre illimité d'interactions entre les objets qui peuvent se déplacer dans le décor. Ils prennent en compte différentes formes de volumes englobants pour les collisions (boîtes, sphères, *SLABs*, etc., voir Figure 26). Et selon les caractéristiques de masse et de taille des objets, les moteurs physiques répondent d'une manière réaliste aux chocs, à l'inertie de rotation et d'entraînement, ainsi qu'aux

accélérations de Coriolis[18], ce qui permet d'obtenir d'une bonne simulation d'un environnement réel.

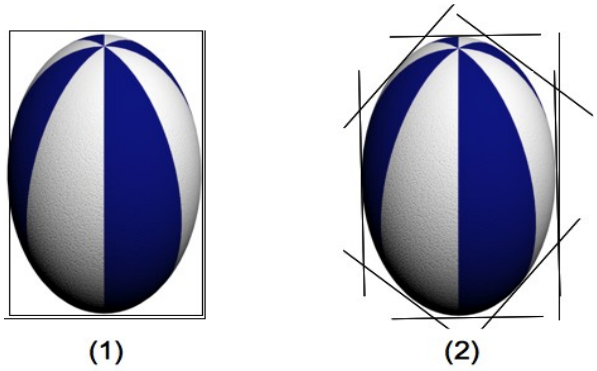


Figure 26: Exemple de volumes englobants de type : (1) boîte englobante, (2) SLABs [28]

Dans une simulation, le moteur physique a ainsi pour rôle de régir l'évolution d'un monde virtuel composé d'objets ayant des propriétés dynamiques et géométriques. Cette tâche est décomposable en deux modules : la gestion du mouvement des objets au sein du monde virtuel et la gestion des collisions entre ces derniers.

II.4.3.1- Gestion des collisions

La détection des collisions peut être effectuée selon plusieurs techniques en fonction du type de géométrie utilisé et des performances souhaitées. Dans le cadre des méthodes dites *discrètes*, qui constituent la majorité des techniques utilisées, ce calcul de collision consiste à détecter le chevauchement entre deux objets, ou bien une grande proximité entre ces derniers (quand la distance devient inférieure à un seuil). Les méthodes dites *continues* détectent l'instant exact du contact entre les objets.

Quelle que soit la méthode choisie, cette première étape s'avère la plus lourde des tâches à effectuer par le gestionnaire des collisions. En effet, la quantité de calculs à effectuer dans le cadre de cette détection est directement liée à la façon dont les objets, ou leurs volumes englobants, sont modélisés dans le monde virtuel, particulièrement en termes de complexité géométrique (compromis à faire entre réalisme et temps de calcul).

Après cette première étape de détection des collisions, il faut déterminer les efforts résultant des collisions entre les paires d'objets concernés, afin de passer la main au gestionnaire de mouvements.

[18] Accélération de Coriolis : (définition) terme d'accélération qui intervient lorsque l'on étudie le mouvement d'un corps se déplaçant dans un référentiel en rotation par rapport au référentiel galiléen.

II.4.3.2- Gestion des mouvements

Le but de ce module est de contrôler les mouvements des objets contenus dans le monde virtuel selon des lois comportementales généralement calquées sur la physique du monde réel.

Dans un premier temps, les efforts appliqués sur l'objet virtuel sont calculés. Ces efforts peuvent être de plusieurs origines : la gravité, les contraintes appliquées aux liaisons (articulations, glissements, etc.), les collisions entre objets, les manipulations interactives effectuées par l'utilisateur.

L'effort appliqué à un objet va être la conséquence de l'ensemble des efforts calculés auparavant, quelle qu'en soit la cause.

Lors de la deuxième étape, les accélérations linéaires et angulaires de l'objet vont être déterminées par application du principe fondamental de la dynamique.

La troisième étape consiste à modifier les positions et orientations de l'objet considéré, au sein de la scène, à chaque pas d'évolution. Comme l'animation est effectuée en temps discret, on obtient un échantillonnage du mouvement des avatars (le mouvement d'un corps est assimilable à une succession de positions). Les calculs des positions à base d'intégration peuvent se faire selon diverses approches mathématiques implicites ou explicites (Euler, Tustin, Runge-Kutta, etc).

Cette approche permet de simplifier la simulation de la dynamique dans un environnement numérique.

Obtenir un tel résultat découle dans un premier de la résolution des contraintes par un système de résolution approprié mais toujours complexe, puis d'un calcul et d'une application des efforts et enfin d'un calcul des positions suivant le tempo de l'animation.

Cette problématique, engendre à elle seule de nombreux travaux au sein de la communauté scientifique, et ne sont pas tous détaillés ici.

II.4.3.3- Problématiques liées à la contrainte temps-réel

Comme indiqué dans la thèse de L. Tching [58], une simulation interactive nécessite une fréquence de fonctionnement élevée et donc des calculs très rapides au niveau du moteur physique, et cela tout particulièrement lorsque les interactions sont effectuées via un système haptique. Or, les moteurs physiques actuels, que ce soient des produits commerciaux ou des prototypes de recherche, ne permettent pas d'obtenir des

performances suffisantes, quand la scène contient un grand nombre d'objets dont la géométrie est complexe.

Cette relative lenteur des moteurs physiques est principalement due à la gestion des collisions. C'est encore pire quand le système manipule des objets déformables, qui nécessitent encore plus de calculs. Une solution consiste à simplifier la scène en traitant des volumes englobants, ou une simplification du maillage des objets.

Mais ceci peut se révéler inutilisable dans le cas d'applications nécessitant une très grande précision comme la simulation d'actes chirurgicaux ou la manipulation d'outils à distance.

II.4.4)Conclusion

Au travers de cette partie, nous avons présenté le cadre d'application et les principes fondamentaux de fonctionnement d'un moteur physique. Nous avons également introduit un exemple utilisant un moteur physique pour une application artistique et non uniquement basée sur de l'animation géométrique comme c'est le cas pour les simulations (dont les jeux vidéo) les plus classiques.

On peut s'apercevoir qu'un moteur physique est un élément crucial pour créer un monde virtuel intéressant. Sans lui, il serait difficile de simuler simplement des phénomènes de collision et certains mouvements complexes.

Par contre, la mise en œuvre d'un moteur physique est extrêmement complexe et nécessite un effort de développement important.

Étant donné que nos travaux ont nécessité l'introduction d'un moteur physique, nous avons préféré nous baser sur un système existant et libre, appelé « Bullet physic library »[19], et ne pas chercher à développer nous-même un moteur physique spécifique, comme nous le verrons dans la partie contribution de ce mémoire.

[19] http://bulletphysics.org

Partie III- Contribution

III.1- Introduction

Dans cette partie, nous abordons notre contribution au projet pluridisciplinaire DRAMA. Avant cela, il faut rappeler que ce projet, mettant en collaboration le laboratoire LLA-CREATIS de l'université Toulouse le Mirail (pour la partie SHS) et l'équipe VORTEX du laboratoire IRIT de l'université Paul Sabatier (pour la partie STIC), a pour objectif principal d'étudier la faisabilité de l'apport de techniques informatiques pour la mise en scène de pièces de théâtre.

Les études menées autour de ce projet s'inscrivent dans le domaine de la modélisation déclarative pour la génération de scène en trois dimensions à partir des descriptions ou des propriétés extraites du texte initial de l'auteur.

Dans un premier temps, la présentation de nos travaux s'articule autour des connaissances spécifiques au domaine théâtral. Ces connaissances doivent être repérées dans le texte initial de l'auteur, afin de mettre en évidence les données pertinentes et nécessaires à la mise en scène, et au-delà, les informations utiles aux personnes travaillant sur des corpus de textes de théâtre. Les textes ainsi annotés doivent constituer ce corpus.

Le stockage de ces informations nécessite l'étude d'un langage de description, permettant la représentation informatique des données théâtrales. Ce langage, basé sur le modèle du langage XML[20], doit être adapté aux besoins de ce contexte particulier.

Ce langage sert dans un premier temps à l'identification des éléments structurants, mais sert également à mettre en évidence les indications précises fournies par un auteur ou le

[20] *XML : Extensible Markup Language*, ou langage de balisage extensible ; syntaxe reconnaissable à partir des *chevrons* (< >) encadrant les *balises*

metteur en scène pour un texte donné, par le biais d'annotations. La prise de notes étant une tâche habituellement effectuée grâce à un *carnet de mise en scène*.

L'annotation permet l'identification d'informations pertinentes et se fait grâce à des balises insérées dans le texte. Ces annotations permettent ainsi au metteur en scène, ainsi qu'aux autres utilisateurs, non seulement d'identifier, mais également d'obtenir des informations sur un texte, un ensemble de textes ou même sur tous les textes présents dans un corpus, grâce à la formulation de requêtes.

A la suite de cette première phase d'étude, nos travaux ont également concerné la modélisation des contraintes, permettant de passer de l'étape de description (identification des propriétés descriptives dans le texte de l'auteur ou ajout d'éléments descriptifs par le metteur en scène) à la phase de génération automatique d'une scène 3D correspondante.

Une base de connaissances permet par la suite de générer des mises en scènes virtuelles, en établissant la correspondance entre un contexte particulier, un texte, un auteur et les préférences du metteur en scène.

De ce fait, l'étude de l'interprétation des descriptifs scéniques en contraintes (par exemple relations spatiales) est nécessaire, pour que la traduction de la description corresponde bien à la scène 3D souhaitée par le metteur en scène. Ses souhaits ayant été exprimés par le biais d'annotations ajoutées dans le texte et prises en compte par l'outil de génération.

Pour cela, Il est nécessaire de disposer d'un système de génération à la fois robuste et efficace, pour que les scènes générées puissent être à la fois suffisamment riches (comportant de nombreux éléments) et obtenues dans des temps acceptables pour l'utilisateur.

La Figure 27 illustre notre contribution apportée au projet DRAMA.

Notre contribution concerne deux étapes :

- DRAMAtexte :l'étape initiale permettant à un utilisateur d'annoter un texte, par le biais de balises, afin d'ajouter les informations qu'il jugera nécessaires, pour identifier notamment les éléments descriptifs,
- DRAMAscène :l'étape de génération qui exploite les informations extraites des annotations afin de générer automatiquement une scène 3D correspondant à une proposition d'aménagement virtuel de toutes les entités devant être présentes sur une scène virtuelle ou réelle.

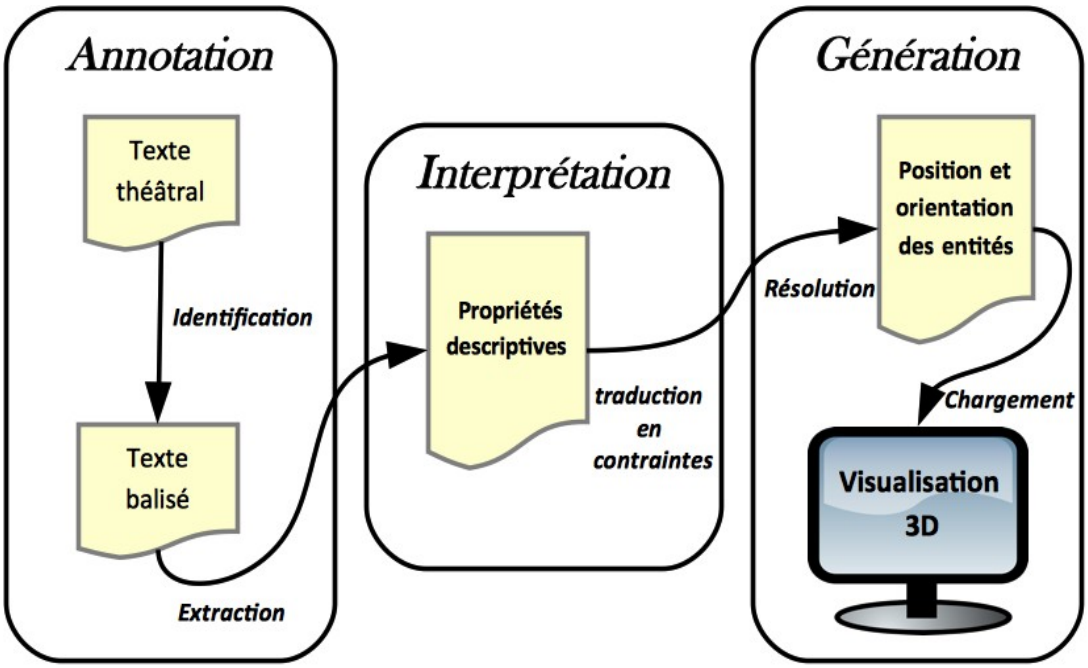


Figure 27: Contribution au projet DRAMA : Etapes de Balisage et Génération

III.2- DRAMAtexte

DRAMAtexte concrétise, sous la forme d'un outil informatique, les études menées dans le cadre du projet DRAMA sur la spectalecture [26][44] et la caractérisation des éléments de notation des différents indicateurs présents dans un texte théâtral.

Ces travaux ont été présentés lors d'un colloque sur la notation informatique du personnage théâtral [22] et ont également donné lieu à une publication dans un colloque international [1][51].

III.2.1) Problématique

Les metteurs en scène utilisent des *carnets de mise en scène*, la plupart du temps en version papier, qui leur permettent de prendre des notes lors de toutes les étapes du processus de création, depuis les premières lectures jusqu'aux répétitions, et même lors des premières représentations. Ces carnets sont très souvent enrichis de croquis, voire de photographies. Mais les données numériques comme les vidéos ou les prises de sons ne peuvent évidemment pas venir enrichir directement ce carnet du metteur en scène.

Les metteurs en scène utilisent éventuellement des carnets de notes sous des formats numériques grâce à l'utilisation de traitements de texte, mais non adaptés aux besoins spécifiques du processus de création.
Par ailleurs, il existe également des outils qui permettent d'assister le scénographe, l'éclairagiste, ou encore le costumier, mais ces outils ne sont pas spécifiques au théâtre et là encore ne sont pas adaptés aux besoins spécifiques du metteur en scène. Aucune plateforme n'est actuellement capable de rassembler les différents outils et de structurer la création théâtrale en fonction des différentes composantes du spectacle vivant. C'est pourquoi, nous travaillons à la mise en place d'un nouvel outil qui s'appuie sur un nouveau système de balisage spécifique au travail théâtral, permettant d'identifier facilement les éléments importants et impliqués dans la création scénique.
Or, ce travail de mise en évidence d'éléments dans un texte de théâtre peut également être effectué par un lecteur (chercheur en études théâtrales ou en littérature), ou bien un metteur en scène, mais en dehors de toute production.
C'est pourquoi l'outil développé dans le cadre de DRAMA doit être ouvert et adaptable aux besoins des différents utilisateurs, sans perdre de vue sa fonction première d'assistant à la mise en scène.

III.2.2) Objectif

L'objectif du module DRAMAtexte est donc de fournir à l'utilisateur, lecteur ou metteur en scène, la possibilité de mettre en évidence les éléments caractéristiques, permettant non seulement de travailler sur des textes, mais aussi d'obtenir la génération automatique d'une mise en scène virtuelle à partir des indications sur les positions relatives d'éléments de décors, d'accessoires ou même de personnages.
Les éléments qui sont mis en avant grâce à DRAMAtexte peuvent être issus du texte initial et donc des indications fournies par l'auteur grâce aux didascalies, ou issus d'indications ajoutées par le metteur en scène pour apporter sa propre vision et ses propres choix de mise en scène.
Ce module est donc destiné à être un outil de balisage du texte théâtral, permettant un travail sur les différents indicateurs de scénographie introduits par l'auteur ou le metteur en scène. DRAMAtexte doit également permettre de visualiser les éléments identifiés, sous une forme adaptée, que ce soit sous la forme de listes, de graphiques, de schémas, d'une visualisation en 2D ou en 3D (via une scénographie virtuelle), afin de servir de cadre

au processus d'élaboration de spectacles, ceci grâce à des requêtes simples ou croisées (recherche simultanée selon plusieurs critères) permettant d'interroger le texte.

Nous présentons, dans cette section, les travaux développés pour DRAMAtexte concernant l'annotation et l'enrichissement de textes de théâtre, via l'introduction de balises. Notre but initial est de concevoir et de mettre en œuvre un outil destiné à la mise en scène d'œuvres dramatiques, et donc de scénariser l'écrit via l'extraction, l'identification et l'ajout de toutes les informations nécessaires à l'élaboration d'une mise en scène théâtrale.

Dans le paragraphe suivant, nous allons commencer par faire un tour d'horizon sur les travaux antérieurs concernant les techniques de balisage (marquage) de texte afin de poser les bases de la conception de notre outil, avant de présenter notre approche.

III.2.3) Travaux antérieurs

Les premières formes de balisage ont été introduites manuellement dans les textes par les typographes, grâce à l'ajout de symboles servant à structurer visuellement le texte. La première forme de balisage est constituée par les symboles de ponctuation et les espaces utilisés pour délimiter les chapitres, les paragraphes et les phrases.

Appliqué au texte sur support électronique, le balisage permet d'ajouter beaucoup plus d'informations, en permettant l'ajout d'informations qui seront traitées par l'outil qui visualisera le texte, que ce soit un navigateur ou un traitement de texte. Le balisage permet notamment de formater et de structurer un texte.

Un système balisage de texte, très répandu et utilisé pour la création de sites web, correspond au langage à balises HTML[21]. Un ensemble de balises est utilisé pour préciser les informations de formatage des pages affichées dans les navigateurs.

L'exemple donné dans la Figure 28 montre un texte balisé avec des balises HTML et la façon dont il s'affiche dans le navigateur.

[21] HTML : HyperText Markup Language

Texte balisé :

Essai de page web avec une partie en gras, une partie en italique. Une partie centr&eacute ;e.

Une liste &agrave ; puces :

premier

second

Résultat :

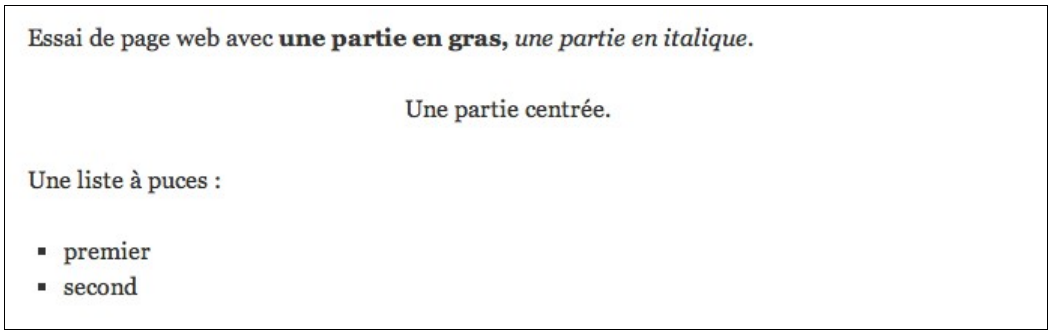


Figure 28: Exemple formatage de texte grâce à des balises HTML

On peut voir dans ce petit exemple que ce langage permet de préciser, grâce à des couples de balises, le format d'une portion de texte. Généralement un couple de balises encadre le texte à formater, comme <*strong*> ...</*strong*> qui encadre un texte devant apparaître en gras. Un couple de balises est donc le plus souvent sous la forme : balise ouvrante <...>, balise fermante </...>. Certaines balises contiennent des attributs permettant de préciser une valeur, comme par exemple la balise <*p*> qui ouvre un nouveau paragraphe et dont l'attribut *style* permet de préciser que le texte de ce paragraphe est centré <*p style="text-align: center;"*>.

On voit également que les caractères accentués, pour être indépendants de l'encodage utilisé par le navigateur, peuvent être transcrits dans le jeu de caractères fourni par le langage HTML. Dans l'exemple, le 'é' est encodé par *é* et le 'à' par *à* (*acute* étant destiné à l'encodage des accents aigus et *grave* pour les accents graves).

Le langage à balises HTML est donc un langage spécifique, destiné à la mise en forme de textes ou d'images à afficher dans les pages web.

La représentation des informations contenues dans le texte de théâtre nécessite l'étude d'un langage de description adapté, permettant la notation informatique des données spécifiquement théâtrales. Nous avons donc besoin de définir notre propre langage à balises, permettant d'enrichir un texte avec les informations pertinentes pour l'application

qui nous concerne. Ce langage doit permettre en effet de baliser le texte pour en extraire ensuite facilement les données identifiées. A partir d'une version imprimée de la pièce de théâtre, ou au mieux, à partir d'une version numérique et d'un traitement de texte, cette tâche ne peut être effectuée que manuellement et laborieusement, si l'on ne dispose pas d'un outil adapté.

Cependant, le marquage et l'extraction d'informations peuvent être rendus difficiles en raison de la nature même des textes. Un texte peut contenir plusieurs passages rédigés dans différentes langues, des renvois, des notes, des citations, des descriptions scéniques sous forme de didascalies, voire des tableaux. Selon le genre littéraire, la structure du texte est organisée en chapitres, sections, scènes, actes, versets, voire un format particulier et non réellement structuré. Tous ces éléments, souvent implicites, peuvent être nécessaires pour certains types d'utilisateurs. Le format d'encodage du texte doit donc tenir compte, selon les besoins particuliers de l'utilisateur, de la structure logique du document.

Pourtant, les traitements actuellement effectués sur des textes de théâtre, même au format numérique, sont la plupart du temps limités à la lecture ou à la simple recherche d'occurrences de mots ou de chaînes de caractères.

Aussi, pour accéder à un texte de façon réellement dynamique et pour fournir des informations nettement plus significatives, le balisage va se révéler une solution particulièrement intéressante.

À partir de ce constat général, et non propre au domaine théâtral, différents systèmes de balisage ont été conçus afin de répondre à des besoins particuliers. On peut citer le système *COCOA [32][33]* qui date du début des années soixante, qui a été développé à Édimbourg dans le cadre de l'étude de textes écossais et qui est utilisé pour identifier la structure de textes anciens. Il est utilisé par la plupart des outils d'analyse de texte, et notamment dans OCP (Oxford Concordance Program) [30]. Le *Thesaurus Linguae Graecae* a aussi créé *Beta code* qui est un autre système de balisage [45].

On trouve également le système TEX (*Tau Epsilon Xi*) [36], qui est libre d'utilisation et qui permet l'écriture de formules mathématiques, ce qui s'avère très intéressant pour les publications scientifiques. D'autres systèmes de balisages propriétaires (*procedural markup*) sont très répandus, étant donné qu'ils sont intégrés dans des logiciels tels que *MSWord* ou WordPerfect, mais ceux-là ne peuvent être utilisés librement.

Pour autant, dès qu'il a pu être mis en place, le balisage est déjà apparu comme étant essentiel et nécessaire pour obtenir une analyse des textes quasi exhaustive et surtout bien structurée. L'utilisation de ces formats, développés indépendamment les uns des autres, n'a pas favorisé l'uniformisation ou la définition de standard, ce qui implique qu'aucun de ces systèmes de balisages ne puisse être facilement transposé au contexte de notre application.

III.2.4) Balises et balisage

Pour le module DRAMAtexte, nous avons étudié et mis en place notre propre système de balisage conçu spécialement pour l'identification des entités présentes dans un texte théâtral. Ce système de balisage a été mis en œuvre dans DRAMAtexte, afin de pouvoir proposer une plateforme de test permettant d'étudier et d'évaluer les apports du système, notamment en ce qui concerne les procédures d'interrogation du texte.

Ce système utilise un langage propre, basé sur le langage à balises XML, mais adapté aux besoins de ce contexte particulier.

Le balisage consiste à ajouter dans le texte initial des balises qui correspondent à des mots clés, associés ou non à des attributs instanciés. L'ajout de balises dans un texte permet de mettre en évidence la structure de ce texte et d'identifier les informations pertinentes qu'il contient, grâce à une application permet l'ajout de balises, qui extrait et qui exploite ensuite ces informations. Une certaine description de la mise en scène peut donc être obtenue au travers de l'ensemble des informations qui ont été balisées dans le texte.

Cette description peut ensuite être présentée au metteur en scène ou à d'autres utilisateurs que ça intéresse, afin qu'ils puissent en prendre connaissance et la rectifier le cas échéant. La phase de balisage dans DRAMAtexte se fait en deux étapes :

- le balisage automatique,
- le balisage manuel.

La première étape de balisage est automatique, afin de décharger au maximum l'utilisateur de tâches qui peuvent être effectuées par l'outil et sans aucune intervention. Lors de cette étape, ce sont les éléments structurels qui sont identifiés, et qui ne nécessitent pas d'expertise particulière.

III.2.4.1- En amont, formatage du texte

Afin de pouvoir automatiser une partie du processus de balisage, nous avons défini un *formatage type* du texte. Ce formatage permet de récupérer directement la structuration du texte à partir d'un ensemble de contraintes assez limitées, mais surtout il permet d'automatiser un ensemble de traitements.

Le **balisage automatique** consiste donc à repérer et marquer automatiquement certaines parties du texte, facilement détectables par l'outil lui-même.

La nécessité d'imposer un formatage type vient du fait que pour chaque éditeur, chaque collection, chaque catégorie de publications, et pour les textes théâtraux en particulier, le texte ne suit pas le même style ou les mêmes règles typographiques. Ainsi, il sera parfois impossible de retrouver automatiquement, dans un même texte mais publié par différents éditeurs, la structuration initiale en actes et scènes.

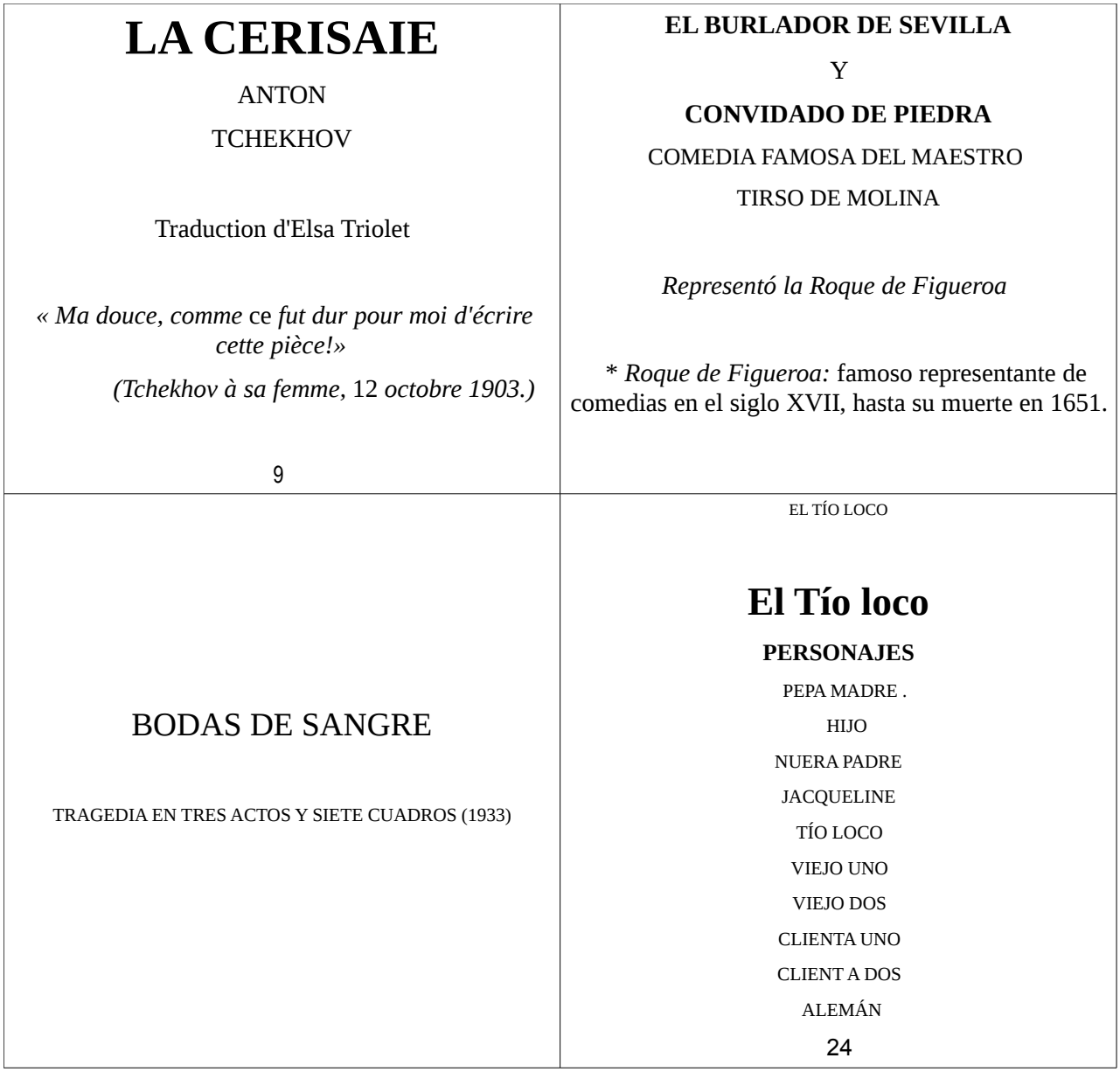
On constate avec les pages de garde présentées dans la Figure 29 la variété et la diversité des mises en forme de ces textes de théâtre[22].

Pour que notre outil puisse reconnaître la structure basique du texte, la version numérique du texte original de l'auteur doit respecter un format imposé, très peu contraignant et simple à mettre en œuvre, pour les textes saisis sous traitement de texte.

Nous avons choisi de nous baser sur le format libre *OpenDocument* (utilisé notamment par la suite bureautique *OpenOffice* qui fournit un traitement de texte sous licence LGPL[23] et donc un logiciel libre). Ce format possède son propre balisage, correspondant à la mise en forme visible via le traitement de texte. Ainsi, il est possible de récupérer ces informations et les traiter pour les adapter à notre propre langage à balises. L'étape de récupération se fera lors de l'importation du document dans notre outil.

[22] Textes cités Figure 29 :
- Tchekhov A. *La Cerisaie*, Hachette, (coll. *Classiques*), Paris, 1991, p.9.
- De Molina T. *El Burlador de Sevilla*, Espasa Calpe, (coll. *Austral*), Madrid, 2000, p.75.
- García Lorca F. *Bodas de Sangre*, Espasa, (coll. *Austral*), Madrid, 1987, p.43
- Cossa R. El Saludador/le salueur, El Tío loco/L'Oncle fou fou, PUM/ Théâtre de la Digue, (coll. Hespérides Théâtre), Toulouse, 2000, p.24.

[23] http://www.openoffice.org/license.html LGPL : GNU Lesser General Public License



LA CERISAIE

ANTON

TCHEKHOV

Traduction d'Elsa Triolet

« Ma douce, comme ce *fut dur pour moi d'écrire cette pièce!»*

(Tchekhov à sa femme, 12 *octobre 1903.)*

9

EL BURLADOR DE SEVILLA

Y

CONVIDADO DE PIEDRA

COMEDIA FAMOSA DEL MAESTRO

TIRSO DE MOLINA

Representó la Roque de Figueroa

* *Roque de Figueroa:* famoso representante de comedias en el siglo XVII, hasta su muerte en 1651.

BODAS DE SANGRE

TRAGEDIA EN TRES ACTOS Y SIETE CUADROS (1933)

EL TÍO LOCO

El Tío loco

PERSONAJES

PEPA MADRE .

HIJO

NUERA PADRE

JACQUELINE

TÍO LOCO

VIEJO UNO

VIEJO DOS

CLIENTA UNO

CLIENT A DOS

ALEMÁN

24

Figure 29: Copie de pages de garde de textes de théâtre [50]

La version numérique du texte que l'on souhaite importer doit donc être au format O*penDocument* (.odt) et respecter un ensemble de règles. Ainsi, notre outil pourra ensuite importer ce texte à partir de l'identification de cette mise en forme prédéfinie.

Ainsi, on peut ajouter les balises ***automatiques*** aux éléments du texte que le logiciel est capable de reconnaître automatiquement : le titre du spectacle, le nom des personnages, les didascalies, les dialogues et les différentes parties.

Le balisage automatique est obtenu à partir de la taille des caractères :

- la taille de caractères pour le titre de la pièce est supérieure à celle du titre des actes,
- la taille de caractères pour le titre des actes est supérieure à celle du titre de la scène,
- etc.

Et le balisage automatique est également obtenu relativement à un ensemble de règles typographiques simples :

- nom des personnages commençant par une majuscule et suivis d'un ':' ou d'un '-',
- titre toujours en gras,
- taille de caractères respectant la structuration des titres des parties,
- didascalies en italique,
- etc.

Le format imposé correspond à celui présenté dans le Tableau 3 ci-après :

TITRE (taille 31)
SOUS TITRE (taille 22)
Dramatis personnae (taille 18)
Partie (taille 15)
Sous partie (taille 13)
Personnage : (nom personnage suivi de « : » taille 11)
Dialogue (taille 11)
Didascalies (Italiques taille 11)

Tableau 3: Format pour les textes importés

Après cette première étape, le texte est chargé dans l'outil de façon totalement transparente pour l'utilisateur, et tout ce qui a pu être balisé automatiquement l'a été.

L'étape suivante est le **balisage manuel**, mais avant d'expliquer la mise en œuvre de ces étapes nous allons présenter l'ensemble de balises qui ont été créées pour permettre à l'utilisateur d'annoter le texte.

III.2.4.2- Balises

L'ensemble des balises proposées à l'utilisateur est organisé en différentes catégories. Nous avons créé[24] les catégories de balises suivantes :

- personnage : dans laquelle nous avons regroupé les balises ayant un rapport avec les personnages présents dans le texte, telles que le nom d'un personnage, un groupe de personnages, la morphologie, les dialogues, etc.,
- structure : regroupant les balises qui spécifient la structure du texte, comme le début du texte, le découpage, la fin, etc.,
- scénographie : identifiant tous les éléments qui composent le décor, tels que les accessoires, les lumières, les espaces, etc.

Après la première phase, qui correspond à la mise en forme du texte et à son importation dans l'outil (basée sur le format OpenDocument et les règles de mise en forme), la structure globale du texte est partiellement balisée et déjà interrogeable par l'utilisateur. L'importation du texte initial déclenche directement l'opération de balisage automatique. Les balises utilisées dans DRAMAtexte ont la structure suivante :

<TYPE propriété="valeur"> entité </TYPE>

Une balise est spécifiée par un nom et elle peut avoir plusieurs attributs, conformément à la définition des balises XML. L'exemple ci-dessus montre la structure d'une balise nommée TYPE qui est une caractéristique associée à *entité,* et TYPE a comme attribut *propriété* dont la valeur est donnée par "valeur".

Les balises insérées dans le texte sont caractérisées grâce à la première lettre de la balise :

- qui est de la forme <A…> pour les *balises automatiques*,
- et de la forme <M...> pour les balises insérées manuellement par un lecteur (que nous appelons *balises manuelles*), mais sans modification du texte initial,
- et de la forme <D...> pour des balises correspondant à une modification du texte initial de l'auteur, uniquement effectuée par le metteur en scène (directeur), ceci pour permettre de différencier les indications présentes dans le texte original (de l'auteur) et les ajouts éventuels (du metteur en scène).

[24] En collaboration avec Matthieu Pouget [50]

Entité	**Signification de la balise**	**Nom de la balise A=Automatique**
Pièce	PLAY=PL	<APL>
Titre pièce	TITLE=TI	<ATI>
Sous Titre Pièce	SUBTITLE=STI	<ASTI>
Dramatis Personae	PERSONAE=PE	<APE>
Titre Dramatis	TITLE=TI	<ATI>
Personnage	CHARACTER=CH	<ACH>
Groupe de personnages	GROUP=GR	<AGR>
Découpage	PART=PA	<APA>
Dialogue	DIALOG=DIA	<ADIA>
Didascalies	DIDASCALIA=DID	<ADID>

Tableau 4: Balises automatiques

Chaque balise ouvrante est toujours associée à une balise fermante. Par exemple <ACH> signifie le début de la définition d'un personnage et </ACH> en marque la fin.

Afin de permettre l'identification de chaque entité dans la scène, nous avons mis en place un ensemble de balises spécifiques à DRAMAtexte, correspondant aux éléments que nous souhaitons permettre de mettre en évidence. Nous avons associé à ces balises des attributs décrivant leurs propriétés ainsi que des valeurs permettant de qualifier les attributs.

Illustrons les résultats dans DRAMAtexte sur un exemple de *didascalie*, correspondant à l'extrait d'un texte théâtral[25], décrivant les entités présentes sur la scène :

> *Dans le salon bourgeois de Mrs Peacock. La vieille dame, Mrs Peacock, est assise dans son fauteuil, un gros carnet et un stylo plume en mains. Elle griffonne, réfléchit. Auprès d'elle, Rose, la bonne, dépoussière vigoureusement les nombreux bibelots alignés sur les étagères, nettoie la pendule à coucou sur laquelle elle peut éventuellement s'attarder. Mrs Peacock s'interrompt dans son écriture et observe la bonne d'un œil sévère.*

Texte 1: Extrait de « Sous les masques »

[25] « Sous les masques » Auteurs : Pascale Hedelin, Christophe Merlin. Éditeur : Milan. Collection : Aujourd'hui théâtre

La phase de balisage automatique va déterminer que ce paragraphe dans son entier est une didascalie, car il est écrit en italique (convention d'écriture usuelle dans les textes théâtraux, que nous reprenons dans le formatage que nous utilisons).
Le passage balisé va donc être encadré par les balises <ADID> et </ADID> :

> *<ADID> La vieille dame, <ACH>Mrs Peacock</ACH>, est assise dans son fauteuil, un gros carnet et un stylo plume en mains. Elle griffonne, réfléchit. Auprès d'elle, <ACH>Rose</ACH>, la bonne, dépoussière vigoureusement les nombreux bibelots alignés sur les étagères, nettoie la pendule à coucou sur laquelle elle peut éventuellement s'attarder. <ACH>Mrs Peacock</ACH> s'interrompt dans son écriture et observe la bonne d'un œil sévère. </ADID>*

Texte 2: Extrait de « Sous les masques » après l'étape de balisage automatique

Nous avons défini un ensemble de balises que nous avons jugées utiles, non seulement pour l'étude d'un texte théâtral, mais également pour la phase de génération de scène.
Cet ensemble de balises n'est pas limitatif et l'utilisateur a la possibilité d'ajouter des balises en précisant leur nom, les attributs et les valeurs qui sont associés à cette nouvelle balise.
Les balises automatiques sont listées dans le Tableau 4. Le texte après avoir été importé dans l'outil et après avoir été balisé automatiquement est visible dans la Figure 30. Le texte balisé dans cet exemple est celui de la pièce « *Sang de lune »* de José Sanchis Sinisterra [26].
Une fois le balisage automatique effectué, les didascalies et les dialogues ont été identifiés, ainsi que les personnages et les parties.
Les balises manuelles sont présentées dans le Tableau 5 qui récapitule les entités, les significations et les noms de balises avec le préfixe 'M' pour manuel et le préfixe 'D' pour Director (modifications et annotations effectuées par le metteur en scène).
Le détail de la signification des balises et leur interprétation en termes de mise en scène peut être consulté dans le mémoire de doctorat de Mathieu Pouget [50] qui est metteur en scène et qui a participé à la spécification de l'outil DRAMAtexte dans le cadre de son doctorat.

[26] La pièce Sangre lunar de José Sanchis Sinisterra a été jouée pour la première fois au Théâtre de la Digue par la compagnie Les Anachroniques en avril 2003 et la première publication du texte de Sinisterra en mars 2003 : Sanchis Sinisterra J. Conspiración Carmín/Conspiration Vermeille;Sangre Lunar / Sang de lune, PUM/ Théâtre de la Digue, (coll. Hespérides Théâtre), Toulouse, 2003.

Entité	Libellé	Balises Manuelle	Balises Directeur	Attributs
Pièce	PL	<MPL>	<DPL>	
Titre pièce	TI	<MTI>	<DTI>	
Sous-titre	STI	<MSTI>	<DSTI>	
Dramatis personae	PE	<MPE>	<DPE>	
Titre Dramatis	TIT	<MTIT>	<DTIT>	
Personnage	CH	<MCH>	<DCH>	
Découpage	PA	<MPA>	<DPA>	
Dialogue	DIA	<MDIA>	<DDIA>	
Didascalies	DID	<MDID>	<DDID>	
Accessoires	PR	<MPR>	<DPR>	Type, Character, Color
Actes	AC	<MAC>	<DAC>	Type, Character, Time
Actes de langage	LAC	<MLAC>	<DLAC>	Type, Character, Time
Morphologie	MOR	<MMOR>	<DMOR>	Type, Costume, Coiffure, Makeup, Character, Color
Entrées/Sorties	SPO	<MSPO>	<DSPO>	Type, Character, Time
Déplacements	MO	<MMO>	<DMO>	Character, Object, Set, Time
Positions	POS	<MPOS>	<DPOS>	Type, Localisation, Source, Cible
Orientation	OR	<MOR>	<DOR>	Localisation, Source, Cible
Regards	LO	<MLO>	<DLO>	Type
Ouvertures	ST	<MST>	<DST>	Type, Time
Fermetures	END	<MEND>	<DEND>	Type, Time
Emotions	EM	<MEM>	<DEM>	Type, Degree, Character, Time
Voix	V	<MV>	<DV>	Type, Degree
Eléments Décors	S	<MS>	<DS>	Type
Espaces	SP	<MSP>	<DSP>	Type, Size
Lumières	L	<ML>	<DL>	Degree, Color
Silences	SIL	<MSIL>	<DSIL>	Localisation, Type, Time
Son	SOU	<MSOU>	<DSOU>	Type, Character, Object, Set, Time
Temps	T	<MT>	<DT>	Size
Rôle de parole	F	<MF>	<DF>	Type, Character
Type	E	<ME>	<DE>	Type, Character

Tableau 5: Extrait de la liste des balises utilisées dans DRAMAtexte

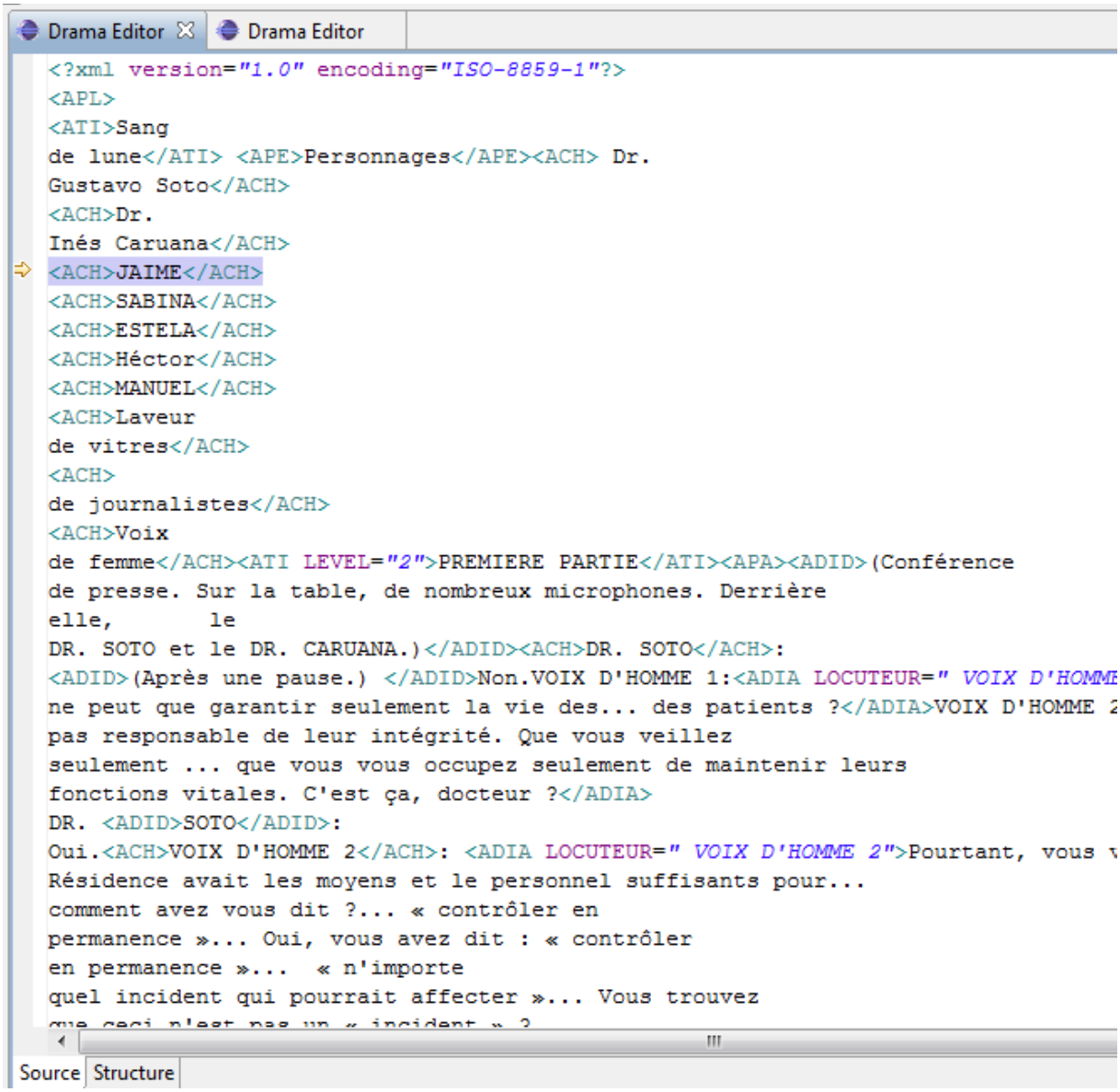


Figure 30: Résultat de l'étape de balisage automatique du texte "Sang de lune"

Notons aussi que différentes balises peuvent être imbriquées, et nous pouvons donc trouver des balises contenues dans une partie de texte déjà encadrée par une autre balise. Dans l'exemple montré dans le Figure 32, nous trouvons deux balises <ACH> qui marquent deux personnages se trouvant à l'intérieur d'une didascalie et d'un dialogue. Ces relations d'emboîtement entre balises, mais également les attributs, doivent suivre un ensemble de règles prédéfinies afin de préserver la logique entre les entités et leurs propriétés. Par exemple, une didascalie peut contenir des personnages ou des accessoires mais l'inverse n'est pas vrai.

Pour cela, nous avons prédéfini un fichier de type DTD (*Document Type Definition*) qui permet de définir, grâce à une grammaire, les différentes règles régissant l'utilisation de chaque balise et de chacun des attributs présents dans la définition d'une balise. En effet, ce type de fichier permet de déclarer les éléments (ou balises) autorisés à apparaître dans le document, ainsi que leurs imbrications possibles. La forme générale d'une déclaration de balise est la suivante :

<!ELEMENT nom_balise>

Mais ces déclarations correspondent à une syntaxe précise que nous avons adaptée au contexte particulier de DRAMA.

Par exemple nous avons défini la balise MCH (balise manuelle Personnage) de la façon suivante dans notre fichier DTD :

- si une balise peut contenir du texte brut, on l'associera avec l'élément #PCDATA,
- la balise qui définit un personnage <MCH> n'admet aucune autre balise entre les éléments <MCH> et </MCH>, mais par contre, il est possible d'y utiliser des valeurs textuelles.

Donc ce type de balise est déclaré sous la forme :

<!ELEMENT MCH (#PCDATA)>

Pour permettre l'imbrication (insertion d'une balise à l'intérieur d'une balise), il faut au préalable recenser toutes les entités qui seront autorisées à être insérées à l'intérieur de cette balise.

Par exemple, la syntaxe de la déclaration de la balise <MAC> (balise manuelle *Acte*) dans notre fichier DTD indiquera que les balises ADIAG (balise automatique Dialogue), ACH (balise automatique Personnage) et MSP (balise manuelle Espace) sont autorisées et cette syntaxe se présente comme suit :

<!ELEMENT MAC (ADIAG ?,ACH+,MSP ?)>

Enfin, si on souhaite limiter les valeurs que peuvent prendre les attributs d'une balise, on définit une règle grâce au fichier DTD. Dans ce cas, l'attribut ne peut prendre qu'un nombre fini de valeurs définies à partir d'une liste exhaustive qui est donnée dans la règle spécifiant une balise. Les valeurs possibles des attributs sont séparées par des '|'.

Illustrons ces différentes règles sur un extrait du contenu de notre fichier DTD :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <!ELEMENT MTI (MCH*)>
    <!ELEMENT ADID (MCH,MSP,MENV,MPOS,MSET,MMO,MSIL,ML,MMOR,MLO,
    MEM,MS)>
    <!ELEMENT ADIA (MCH,MSP,MENV,MPOS,MSET,MMO,MSIL,ML,MMOR,MLO,
    MEM,MS)>
    <!ATTLIST ADIA locuteur #REQUIRED>
    <!ELEMENT MPR (#PCDATA)>
    <!ATTLIST MPR Character #REQUIRED Type #IMPLIED Color #IMPLIED>
    <!ELEMENT MSP (#PCDATA)>
    <!ATTLIST MSP Type #REQUIRED Size #IMPLIED>
    <!ELEMENT MMO (#PCDATA)>
    <!ATTLIST MMO Character #IMPLIED  Object #IMPLIED  Set #IMPLIED
                Time #IMPLIED>
```

Texte 3: Configuration de DTD

Commentaires :

1. Pour chacun des éléments, on définit sa composition par <!ELEMENT nom_element (structure)>,
2. Pour chacun des attributs, on définit sa composition par <!ATTLIST nom_element nom_attribut (structure)>,
   L'attribut « LOCUTEUR » de la balise ADIA (dialogue) est, ici, défini comme étant un choix obligatoire et donc imposé,
3. MCH* signifie que dans une balise MTI, c'est-à-dire la balise identifiant le titre, on peut trouver de 0 à plusieurs éléments de type MCH (personnage),
4. Pour les attributs de la balise MSP, la valeur REQUIRED signifie que l'attribut « type » doit toujours être défini, c'est-à-dire non vide,
5. Par contre la valeur de l'attribut « size » qui définit la dimension de l'espace n'est pas obligatoire,
6. #PCDATA signifie que l'élément en question peut contenir à la fois des textes et des balises.

Notons que ce fichier est créé avant la première utilisation de DRAMAtexte et ne nécessite pas de modification de la part de l'utilisateur qui peut utiliser l'outil avec un fichier de configuration, tel qu'il existe dans sa version initiale.
Par contre, il pourra être mis à jour automatiquement si l'utilisateur ajoute un nouveau type de balise ou lors de la modification de la liste des balises.
Enfin, ce fichier nous permet donc vérifier la conformité des informations associées à notre texte théâtral grâce aux balises, car il permet de valider la syntaxe du texte balisé et donc le contenu des balises.

III.2.5) Mise en œuvre

III.2.5.1- Importation

Une fois que le texte suit les règles typographiques que nous avons définies en collaboration avec Matthieu Pouget [50], nous pouvons effectuer l'importation du texte dans DRAMA. Cette opération se déroule en deux étapes :

- importation et conversion du format initial au format HTML,
- à partir du document HTML obtenu, identification de toutes les entités détectables pendant l'étape de balisage automatique.

III.2.5.1.1) Conversion en format HTML

Par défaut le texte de l'auteur, pour pouvoir être importé et traité dans DRAMAtexte, doit être donné au format .odt[27], mais il est aussi possible d'importer des documents dans d'autres formats, tels que .doc, .rtf, .html, sous réserve de pouvoir ouvrir ces documents avec OpenOffice (ou LibreOffice), et à condition que le texte soit formaté convenablement, suivant les règles que nous avons préétablies.

Les conversions sont effectuées grâce à la procédure suivante :

- OpenOffice (ou LibreOffice) est lancé en mode serveur qui « écoute » sur le port 8100. Pour cela, il faut inclure d'une façon totalement transparente pour l'utilisateur les arguments de la commande « soffice », ce que nous montre la Figure 31.
- Une fois que la connexion est établie avec le serveur, la conversion proprement dite est effectuée grâce à une librairie java : *jodconverter*, qui va convertir le texte au format HTML.

[27] OpenDocument text, format par défaut de OpenOffice et LibreOffice

- Une fois le fichier HTML obtenu, les caractères spéciaux contenus dans le fichier résultant doivent être d'abord encodés en caractères lisibles pour l'utilisateur. Pour cela, la liste des caractères spéciaux et leur encodage HTML est stockée dans un autre fichier de configuration (cf. Tableau 6).

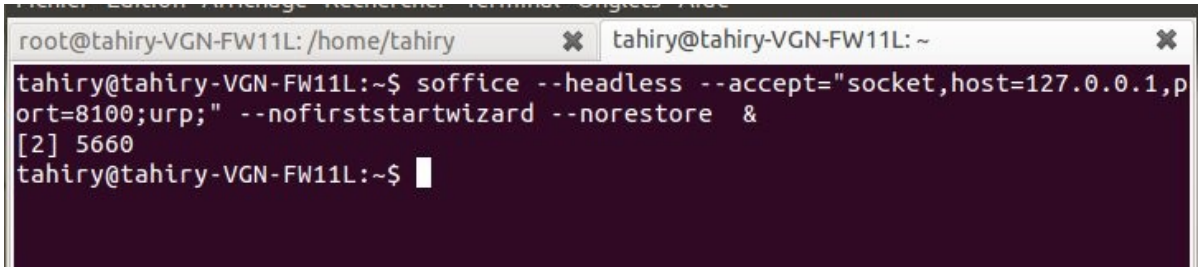


Figure 31: Démarrage de LibreOffice serveur

Caractères spéciaux	Encodage HTML
Espace	;
ê	&ecirc ;
ô	ô
à	à
'	’
é	é
è	è
etc.	etc.

Tableau 6: Caractères spéciaux

III.2.5.1.2) Mise en place des balises automatiques

Après avoir converti et encodé le fichier HTML temporaire sous la forme découlant de l'application de la procédure d'importation décrite au paragraphe précédent, DRAMAtexte peut commencer la détection des entités que nous avons référencées, à partir des formats ou des règles typographiques définies précédemment.

Ainsi, ces entités seront extraites grâce aux balises HTML ajoutées automatiquement dans le texte lors de la conversion au format HTML. Les balises HTML qui vont être utilisées par DRAMAtexte concernent :

- la taille de la police de caractères, ainsi que la hiérarchisation des titres (le titre de la pièce doit être plus gros que le sous-titre, etc.),
- les marques indiquant un formatage,
- les règles typographiques, particulièrement celles indiquées ci-après :

-   spécifie la taille de caractère et permet de repérer le titre des actes,
- <h1> </h1> et <h2> </h2> spécifient les titres dans le fichier HTML, car ils délimitent le titre et les sous-titres de la pièces,
- marque les textes en italique, permettant d'extraire les didascalies,
-   balise des textes mis en gras, utilisée aussi pour les titres.

III.2.5.1.3) Résultats

Le balisage automatique de DRAMAtexte donne aux utilisateurs la possibilité de détecter certaines informations jugées importantes. Ces informations devraient être identifiées et marquées manuellement dans le texte si on ne disposait pas d'un outil sachant effectuer cette tâche à partir des informations dont il dispose au chargement du fichier.

Cette étape nécessite beaucoup de temps aux lecteurs et aux metteurs en scène. DRAMAtexte permet d'éviter cette perte de temps importante, grâce au principe de balisage automatique que notre outil propose. Le tableau suivant montre le temps nécessaire pour effectuer la tâche de balisage automatique de différents textes :

Texte	Nombre de pages à baliser	Temps de balisage (millisec.)
Sous le masque	8	1241
Ces Hommes	45	4046
Los que rien los ultimos	25	1390
Sang de lune	53	6711

Tableau 7: Evaluation des résultats de balisage automatique

III.2.5.2- Balisage manuel

Suite à cette première opération, l'utilisateur peut effectuer le balisage manuel, qui consiste à identifier manuellement les informations implicites présentes dans le texte, mais non identifiées lors de l'étape de balisage automatique. Ces informations concernent en particulier les relations spatiales (localisation relative des entités) et temporelles, les descriptions de personnages, les sons, l'éclairage, etc.

Pour simplifier la tâche des utilisateurs de DRAMA, nous avons cherché à simplifier le plus possible cette tâche (interface utilisateur détaillée dans le paragraphe suivant) en cachant le format du texte balisé sous-jacent.

En effet, l'insertion d'une balise dans le texte se fait simplement en deux actions. L'utilisateur sélectionne le texte qu'il souhaite baliser et ensuite il choisit le type de balise grâce à des menus contextuels qui lui proposent un choix de balises et les valeurs des attributs associés.

Toujours dans un but de simplification et de facilitation, chaque attribut est automatiquement associé à une valeur par défaut, de façon à éviter à l'utilisateur de devoir renseigner tous les champs de façon systématique. Toutefois, l'utilisateur peut modifier les choix par défaut chaque fois qu'il le souhaite.

L'utilisateur a également la possibilité de créer ses propres balises. Et enfin, nous avons inclus dans DRAMAtexte un assistant permettant de faciliter la détection et la correction des erreurs syntaxiques éventuelles pouvant advenir lors de la phase de balisage.

Ainsi, DRAMAtexte permet de prendre en compte les informations initiales contenues dans le texte de l'auteur dramatique, mais également les choix introduits par le metteur en scène. Au final, nous nous retrouvons après les différentes étapes de balisage avec trois types de balises.

Par exemple, la notion de personnage (CHARACTER) peut être caractérisée dans le texte grâce aux trois types de balises : *automatiques* commençant par 'A' <ACH>, *manuelles* <MCH>, et du *metteur en scène* <DCH>. Ceci nous permet de distinguer les annotations effectuées par un lecteur non autorisé à modifier le texte de l'auteur, des annotations effectuées par un metteur en scène qui souhaite adapter le texte de l'auteur.

Cette différentiation des rôles des utilisateurs permet d'éviter des modifications non souhaitées du texte initial, les différents rôles pouvant être affectés à une même personne à des moments différents du travail sur un texte.

Le Tableau 5 (page 87) donne la liste des balises créées par défaut dans DRAMAtexte, mais il faut noter aussi que nous avons mis à la disposition des utilisateurs la possibilité de créer ou de personnaliser leurs propres balises avec les attributs de leur choix. Ces nouvelles balises vont être disponibles en plus de la liste des balises initiales et permettront d'enrichir encore plus l'ensemble des informations qu'il est possible d'extraire ou d'interroger à partir du texte.

III.2.5.2.1) Interface utilisateur pour le balisage manuel

Après l'étape de balisage automatique, le texte doit être balisé manuellement pour renseigner les éléments susceptibles d'être intéressants lors de la phase d'interrogation, pour une personne étudiant un corpus de textes ou lors de la mise en scène pour toute personne intervenant dans la mise en scène.

En effet, l'outil est dans l'incapacité d'identifier automatiquement des éléments descriptifs, car aucun processus de traitement automatique du langage naturel n'a été mis en place dans la version actuelle de l'outil.

Ce qui implique que c'est à l'utilisateur de marquer manuellement les informations qu'il juge importantes dans le texte, à l'aide des balises manuelles, commençant par 'M' ou commençant par 'D' si l'utilisateur est le metteur en scène.

L'exemple déjà présenté dans le Texte 1, une fois balisé, donne le résultat présenté dans le Texte 4.

<ADID>Dans le <MSP> salon bourgeois </MSP> de <ACH>Mrs Peacock</ACH>. La <MMOR AGE= " vielle" SEX = " F" CHARACTER="Mrs Peacock">vieille dame </MMOR>, <ACH>Mrs Peacock </ACH>, est <MPOS TYPE=" assis" >assise </MPOS>dans son <MS TYPE= " Fixed" >fauteuil</MS>, un <MPR TYPE= "mobile" CHARACTER="Mrs Peacock">gros carnet</MPR> et un <MPR TYPE= "mobile" CHARACTER="Mrs Peacock">stylo plume</MPR> en mains. Elle <MAC TYPE="simple">griffonne</MAC>, <MAC TYPE="simple">réfléchit</MAC>. Auprès d'elle, <ACH>Rose</ACH>, la <MMOR SEX = " F" TYPE= "bonne" CHARACTER="Rose">bonne</MMOR>, <MAC TYPE="simple">dépoussière vigoureusement</MAC> les <MPR TYPE= "fixed" NUMERAR="lot">nombreux bibelots</MPR> alignés sur les <MS TYPE="fixed">étagères</MS>, <MAC TYPE= "simple"> nettoie</MAC> la <MS TYPE="fixed">pendule à coucou</MS> sur laquelle elle peut éventuellement s'attarder. <ACH>Mrs Peacock </ACH> <MAC TYPE= "simple> s'interrompt dans son écriture</MAC> et <MAC TYPE="simple"><MLO SOURCE="Mrs Peacock" CIBLE= "Rose">observe </MLO></MAC>la bonne d'un œil sévère. </ADID>

Texte 4: Fichier XML résultant de l'étape de balisage manuel

Notons aussi que l'utilisateur final de l'outil n'est jamais obligé d'avoir à manipuler le texte au format XML tel qu'il est montré ci-dessus. D'ailleurs l'utilisateur de l'outil n'a jamais accès au texte sous sa forme interne mais accède aux informations via une interface adaptée.

L'introduction de balises manuelles est réalisée dans le texte initial à l'aide de l'interface utilisateur de l'outil DRAMAtexte, telle qu'elle est présentée Figure 32 et Figure 33. Cette interface masque totalement le format XML du balisage à l'utilisateur, de façon à faire de DRAMAtexte un outil convivial.

Une fois que l'utilisateur juge le texte correctement balisé, il peut en conserver le résultat pour une utilisation ultérieure. Ceci permettra à terme de créer une base de données de textes pré-balisés accessibles aux personnes intéressées.

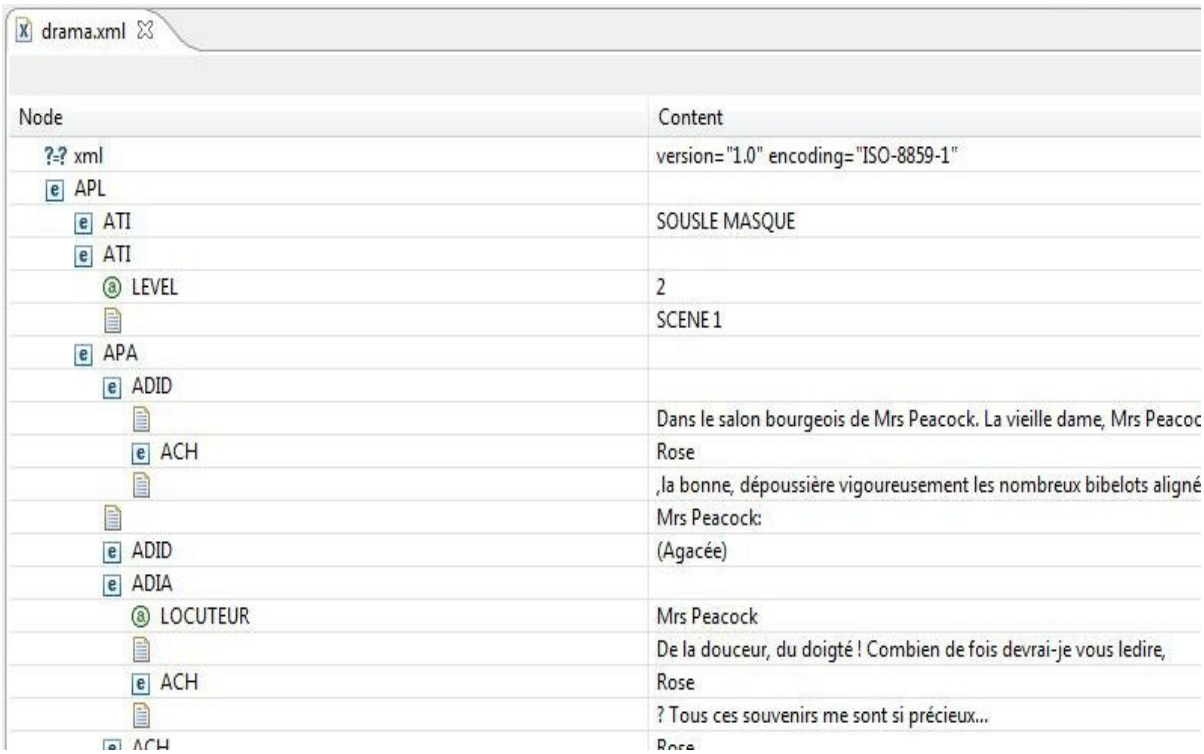


Figure 32: Affichage structuré d'un texte après le balisage automatique

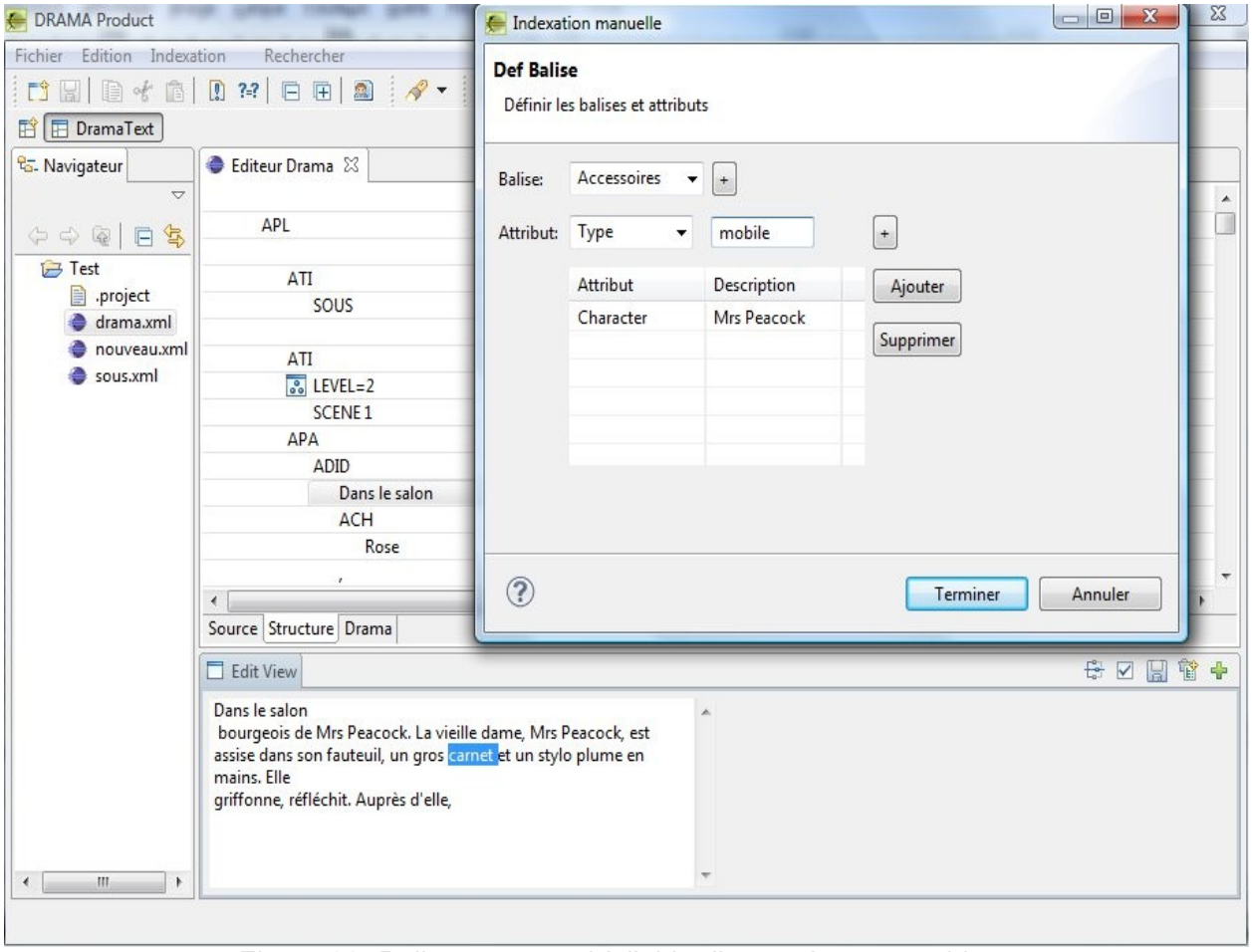


Figure 33: Balisage manuel à l'aide d'un assistant graphique

III.2.5.3- Création de balises

La liste des balises prédéfinies dans DRAMAtexte peut être aussi modifiée dans le cas où l'utilisateur veut créer ses propres balises ou en définir les attributs.

La modification de cette liste dépend du statut de l'utilisateur. En effet, cette manipulation n'est pas accessible à tous les utilisateurs et nécessite d'avoir les permissions nécessaires.

Pour cela, nous avons mis en place un système d'identification qui nous permet d'identifier chaque utilisateur à chaque ouverture de session, en lui demandant son nom d'utilisateur, son mot de passe, et son statut (Figure 34).

Le rôle de l'utilisateur est ainsi défini dès le début de l'utilisation de l'outil, car cela est nécessaire pour définir le type de balises qu'il peut ajouter dans le texte (de type 'M' ou 'D') et également s'il a un statut lui donnant le droit de modifier la liste de balises.

Notons que les utilisateurs ne peuvent ajouter *que* des balises manuelles et des balises du metteur en scène.

En effet, l'éventuel ajout de balises automatiques nécessite des modifications importantes au niveau de la mise en œuvre de l'outil et ne peut se faire dynamiquement.

Figure 34: Fenêtre d'authentification de DRAMAtexte

Après l'ouverture d'une session par un utilisateur, celui-ci peut utiliser la liste de balises prédéfinies ou charger des balises contenues dans des listes supplémentaires créées par lui-même ou d'autres utilisateurs. En effet, après chaque création de balise, l'utilisateur peut sauvegarder la modification qu'il vient d'effectuer dans un fichier afin de la conserver pour des usages ultérieurs.

Au final, nous pouvons ainsi disposer de plusieurs listes de balises qui peuvent être choisies par des utilisateurs voulant encore plus approfondir l'étude d'un texte.

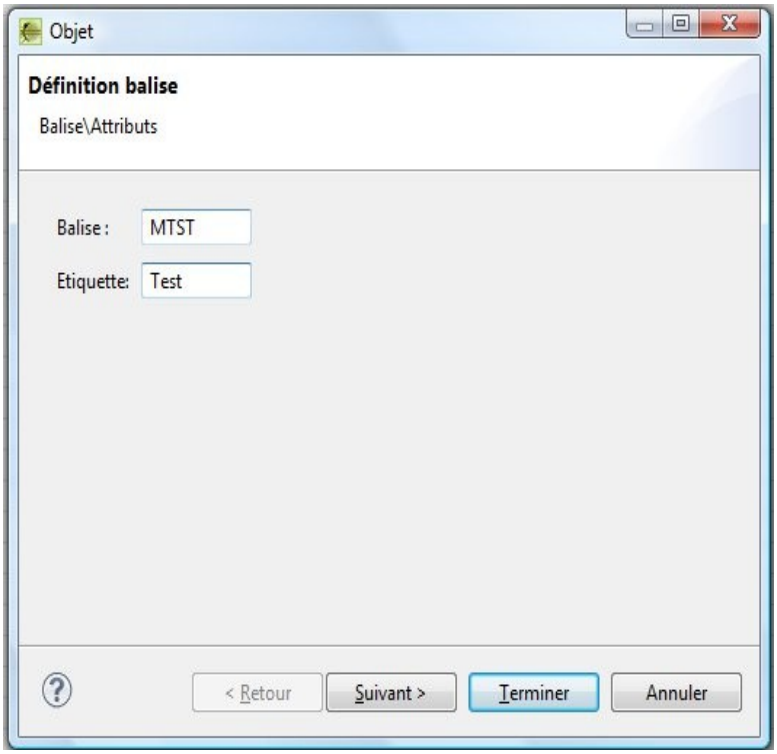


Figure 35: Formulaire d'ajout de balise

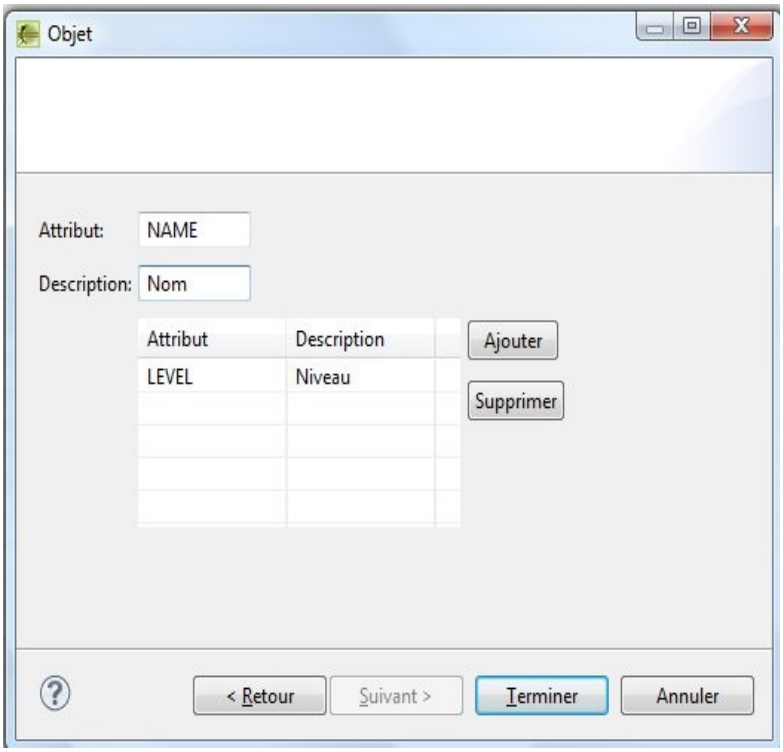


Figure 36: Formulaire de définition des attributs

III.2.5.4- Requêtes

Le texte balisé offre la possibilité de lancer des requêtes sur le texte à l'aide de critères simples ou combinés, afin de pouvoir obtenir la liste des éléments qui auront été balisés automatiquement ou manuellement.

Les différentes requêtes que l'utilisateur peut lancer après la phase de balisage automatique sont les suivantes :

- l'extraction de toutes les didascalies,
- l'extraction de tout le texte (didascalies et textes associés) pour un ou plusieurs personnages,
- l'extraction des didascalies d'une partie du texte (voir l'exemple Figure 37),
- l'extraction des dialogues pour un personnage,
- etc.

L'utilisateur peut ensuite croiser certains éléments entre eux selon les critères sur lesquels ils sont basés.

L'utilisateur peut ainsi lancer des requêtes relatives aux dialogues et obtenir, par exemple :

- les dialogues dans une partie donnée,
- ou bien obtenir tous les dialogues d'un personnage pour la totalité de la pièce, ce qui est intéressant pour fournir les dialogues qui concernent un acteur,
- ou encore obtenir les dialogues correspondant à un acteur et pour une partie donnée.

Mais il pourra également obtenir :

- les informations sur les costumes des personnages dans une scène donnée,
- compter le nombre d'entrées de personnages dans une partie,
- obtenir les accessoires relatifs à un personnage pour la totalité de la pièce,
- ou obtenir les accessoires pour tous les personnages présents dans une scène,
- etc.

Ce processus d'interrogation se base donc sur toutes les informations identifiées par les balises, tels que les éléments scénographiques, les costumes, l'éclairage, etc.

La possibilité de recouper ainsi des informations va permettre au lecteur ou au metteur en scène d'avoir une lecture plus efficace afin d'avoir une vue synthétique du texte.

Mais elle pourra permettre également d'élargir les champs d'étude, en mettant en évidence les aspects encore inexplorés de l'étude du théâtre.

Toutes ces fonctionnalités, basées uniquement sur la structure du texte, offrent des possibilités intéressantes pour toutes les personnes désirant étudier le texte d'un point de vue littéraire.

Le balisage, à condition que les balises correspondantes aient été définies, pourra à terme également permettre :

- de faciliter la lecture partielle ou intégrale de la scène textuelle,
- de créer la partition rythmique du texte,
- d'analyser l'histoire conversationnelle,
- de créer une archive de textes balisés, suivant des points de vues différents : le balisage effectué par un metteur en scène sera probablement différent de celui effectué par un chercheur qui étudie un corpus de textes.

Ainsi, à partir d'un texte balisé, l'utilisateur pourra effectuer des interrogations complexes et dynamiques, afin d'extraire les informations nécessaires à l'étude d'un texte théâtral. Ces deux possibilités complémentaires de balisage et d'interrogation vont être très utiles et sont très attendues par les personnes étudiant le domaine théâtral. Elles faciliteront le travail, qui peut être fastidieux, de recherche exhaustive d'informations, et ce travail pourra à terme se faire dans un corpus de textes, en vue d'en établir une synthèse ou une analyse.

Enfin, dans un but plus pratique, le processus d'interrogation pourra permettre d'obtenir immédiatement des informations sur les éléments de décors ou les accessoires, les éclairages, les costumes ou encore le maquillage, et ceci dans le but de faciliter la tâche de tous les corps de métiers impliqués dans la réalisation d'une mise en scène.

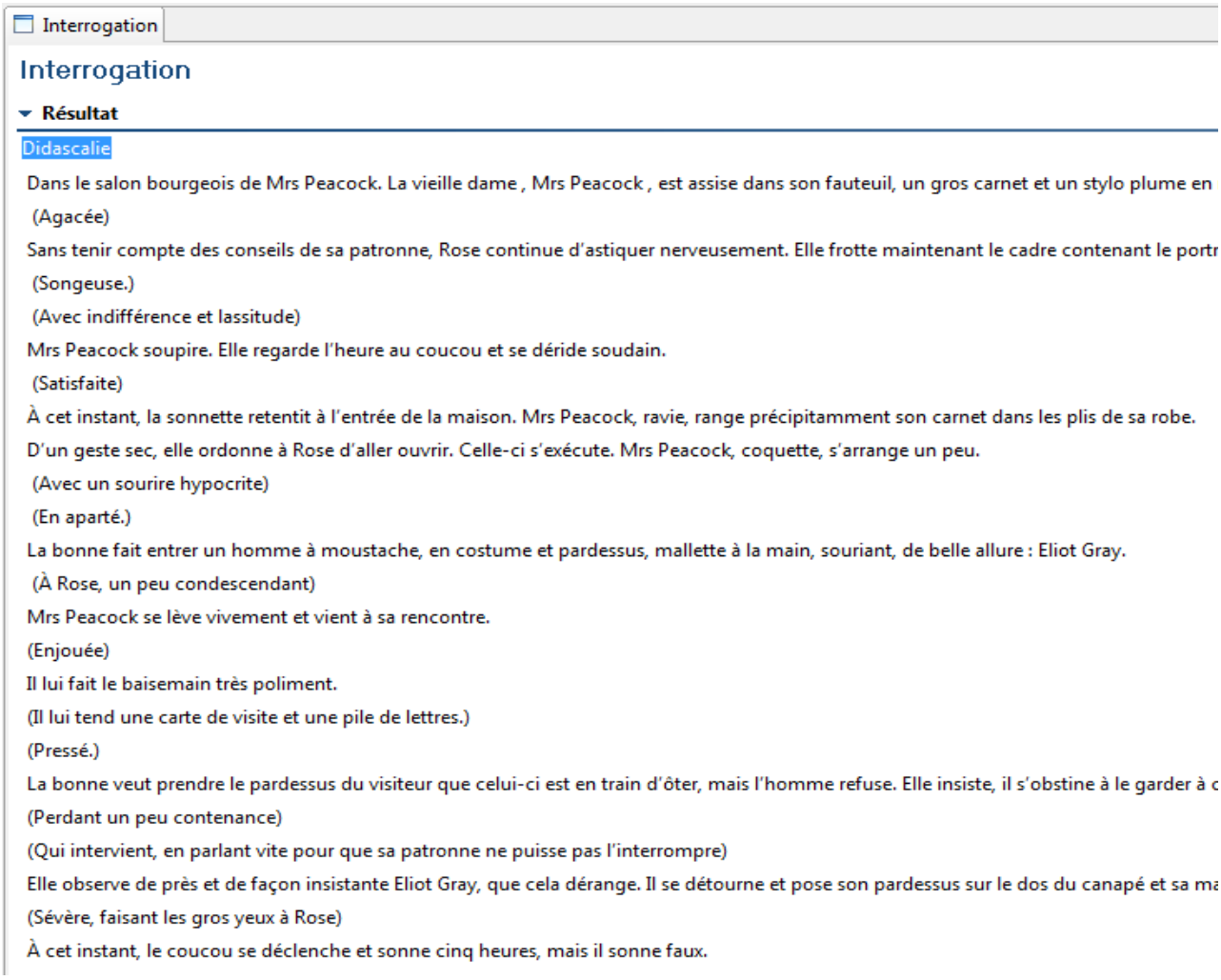


Figure 37: Résultat d'une requête de recherche des didascalies dans un texte après l'étape de balisage automatique

III.2.6) Synthèse

DRAMAtexte est un outil qui permet à un lecteur ou à un metteur en scène d'annoter un texte théâtral grâce à l'ajout de balises qui identifient des informations, des propriétés, etc. Le balisage est également un moyen d'enrichir le texte, en ajoutant des informations ou en identifiant des informations implicites, autorisant par la suite des interrogations pouvant fournir des résultats, dont l'obtention grâce à des moyens classiques se révélerait fastidieuse.

Les opérations de balisage automatiques et manuelles vont pouvoir avoir des applications dans des domaines qui vont au-delà de ceux présentés dans cette thèse. Concernant la recherche fondamentale cet outil va pouvoir faciliter, voire rendre enfin possible à un large public :

- l'analyse synchronique des différents éléments (impossible dans le cas d'une lecture diachronique),
- la visualisation de l'espace et de son occupation par les personnages, grâce à l'apport de l'imagerie 3D telle qu'elle est conçue dans DRAMAscène (visualisation plus difficile dans l'espace imaginaire de la lecture, surtout pour des personnes se formant à la mise en scène),
- la globalisation de la construction du personnage (difficile à réaliser en amont dans la lecture).

Pour la recherche appliquée aux arts de la scène, l'outil va permettre de fournir aux praticiens du théâtre (scénographes, costumiers, acteurs, techniciens, etc.) des listes, des graphiques, et des schémas, en permettant une visualisation de ces différents éléments qui serviront de cadre à l'élaboration de la dramaturgie scénique. Tout ceci va permettre aux généticiens du théâtre de comparer le texte original et le texte enrichi résultant de différentes mises en scènes.

Cet enrichissement du texte se traduit également par l'identification de propriétés (notamment des relations spatiales entre objets ou personnages présents sur la scène), utilisées afin de calculer et proposer une visualisation scénique virtuelle en trois dimensions.

Pour cela, la prochaine étape après la phase de balisage avec DRAMAtexte sera consacrée à l'étude et la mise en place d'un système de génération 3D robuste et rapide

permettant de déterminer, à partir des propriétés et d'une base de connaissances dédiée, un ensemble d'instanciations de mises en scènes possibles, respectant la description et fournissant des solutions satisfaisant les choix matérialisés grâce aux balises ajoutées dans le texte.

Ce système de génération, qui est le fondement de DRAMAscène, est basé sur le principe de la modélisation déclarative. Il a pour but de fournir à l'utilisateur des exemples schématiques de ce que pourrait être la mise en scène, en fonction des contraintes issues du texte, de la volonté du metteur en scène, ou bien encore de la configuration géométrique de la scène.

Cette étape nécessite également, dans sa version finale, l'étude et le développement d'un système d'interprétation dont le rôle est de traduire les propriétés, obtenues grâce au balisage des didascalies et des valeurs des paramètres associés aux balises, en contraintes géométriques ou non géométriques, prises en charge et résolues par le système de génération.

III.3- DRAMAscène

III.3.1) Problématique

Après les phases initiales de lecture du texte accompagnées de prises de notes dans son cahier de scène, un metteur en scène peut passer à l'étape de préparation effective de la mise en scène. Pour cela, il dessine généralement des croquis correspondant aux descriptions qu'il a recueillies dans le texte.

Ainsi, il tente d'imaginer tous les aspects de la mise en scène qu'il souhaite créer, tels que l'emplacement de chaque élément (décor, objets, accessoires), l'environnement de la scène (éclairages, bruitages, etc.), costumes et accessoires de personnages.

Les personnes, en charge des aspects matériels de la mise en place de la scénographie, vérifient la disponibilité de ces éléments dans leur propre collection d'objets. Si ceux-ci sont disponibles mais ne correspondent pas aux contraintes implicites révélées par les croquis du metteur en scène, soit celui-ci modifie ses choix, soit l'acquisition d'un objet ou d'un élément de décor est effectuée.

Il existe différents outils permettant de concevoir une scène numérique en trois dimensions. Mais on peut constater que, d'une part, la manipulation et la maîtrise de ces

logiciels restent encore des difficultés majeures pour des utilisateurs non confirmés, et d'autre part, la prise en charge des contraintes imposées par le texte initial n'est pas garantie.

Ce qui justifie le développement de DRAMAscène, un autre module de DRAMA capable de proposer aux utilisateurs un échantillonnage de scènes, correspondant à la description de la scénographie obtenue à partir du travail de balisage, et respectant de fait les contraintes issues de cette description.

III.3.2) Objectif

Le but principal du module DRAMAscène est donc d'offrir à l'utilisateur, lecteur ou metteur en scène, la possibilité de visualiser une scène obtenue à partir d'éléments de description extraits du le texte de l'auteur ou du metteur en scène. Ainsi, DRAMAscène doit être capable de placer convenablement chaque élément de la scène, et pour cela de calculer la position et l'orientation des objets, selon les propriétés identifiées et extraites grâce à DRAMAtexte.

Dans le cas où la description n'est pas bien contrainte (où les relations spatiales ne sont pas totalement définies), le nombre de solutions pouvant être identifiées dans l'espace de recherche peut être quasiment infini (comme l'illustre la Figure 38).

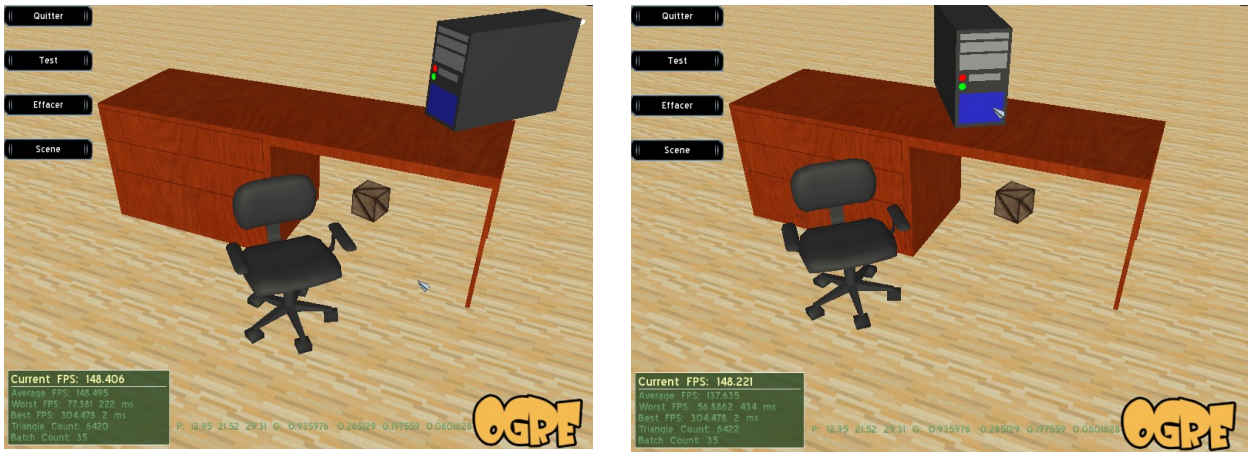


Figure 38: Exemple de deux propositions de scènes, obtenues à partir d'une description sommaire qui correspond à une infinité de configurations possibles en 3D : « le bureau est dans la scène, l'ordinateur est sur le bureau et le fauteuil est devant le bureau. »

Ainsi, avec DRAMAscène, les utilisateurs doivent pouvoir afficher un ensemble de scènes correspondant aux critères initiaux, mais aussi sélectionner la proposition de scène qui leur convient et qu'ils souhaitent conserver.

Par ailleurs, DRAMAscène doit aussi permettre de gérer une banque d'objets 3D utilisés pendant la phase de visualisation. En effet, il devra être capable d'importer des objets à différents formats ou créés via des modeleurs graphiques standards (comme Blender game engine[28], 3DSMax[29], etc.).

Avant d'ajouter un type d'objets dans la banque d'objets, il faudra définir au préalable les propriétés qui définissent ses caractéristiques géométriques et les données qui définissent les contraintes spatiales pouvant exister entre ce nouveau type d'objets et les autres catégories d'objets.

Des travaux concernant la base de connaissances et l'interprétation des propriétés en contraintes sont l'objet d'une étude réalisée dans le cadre d'une autre thèse de doctorat qui est en cours. Cette thèse est consacrée à la conception d'une base de connaissances permettant de générer des acteurs virtuels expressifs à partir des éléments descriptifs présents dans le texte.

En ce qui nous concerne, nous allons voir dans les sections qui suivent les différentes solutions que nous avons étudiées pour la réalisation de DRAMAscène, concernant :

- la génération par contraintes,
- les algorithmes que nous avons adoptés pour la résolution des contraintes,
- l'instanciation et la visualisation de la scène finale.

III.3.3) Génération par contraintes

La génération par contraintes permet de créer automatiquement des scènes, grâce à l'analyse et l'évaluation des propriétés fournies par l'utilisateur. Ces propriétés sont interprétées en un ensemble de contraintes et résolues par un système de génération. Selon la technique de génération employée, les propriétés peuvent être utilisées pour l'obtention exhaustive de toutes les solutions.

Comme les contraintes sont issues des propriétés ou des données contextuelles issues du texte, ceci implique que les propriétés soient traduites au préalable en un ensemble de contraintes, dont la composition dépend du contexte du problème étudié.

[28] Blender: http://www.blender.org/ and Bullet physics library: http://bulletphysics.org/

[29] http://www.autodesk.com/3dsmax

Cette interprétation n'étant pas l'objectif principal des travaux présentés ici, nous avons considéré un sous-ensemble de propriétés auxquelles nous avons associé a priori un ensemble de contraintes géométriques.

Les propriétés sont utilisées pour trois tâches principales : définition de la disposition des objets, partitionnement de l'espace, et construction d'objets complexes par assemblage d'objets simples.

Nous pouvons également rappeler le classement des propriétés en cinq grandes catégories [20] :

- basiques, quand elles sont utilisées pour définir les caractéristiques d'un objet précis,
- floues, qui permettent une description non complète et imprécise,
- générales, concernant principalement les caractéristiques morphologiques, les positions, la numérotation, et l'apparence,
- spécifiques, attribuées à des modèles prédéfinis de forme,
- spatiales, utilisées pour définir la position relative ou absolue des objets dans l'espace scénique.

La génération par contrainte est utilisée pour explorer l'espace des solutions en vue de trouver des solutions correspondant aux souhaits de l'utilisateur. Ces solutions peuvent être trouvées en appliquant les méthodes de recherches *locales* ou des méthodes *complètes,* seules capables d'explorer l'ensemble de l'espace de solutions.

Cependant, l'efficacité des méthodes complètes dépend de l'adéquation de la méthode de résolution, la représentation des contraintes, la complexité de l'espace de recherche, et le domaine d'application.

Pour faire le choix des méthodes, nous considérons quatre critères : l'espace mémoire, la précision de la représentation, l'efficacité et la simplicité de la mise en œuvre.

Les travaux de O. Le Roux [40] ont montré que les méthodes complètes comme les CSPs[30] sont plus adaptées à la résolution de problèmes fortement sous-contraints où beaucoup de solutions existent dans l'espace de recherche, tandis que les méthodes à base de métaheuristiques permettent d'obtenir de meilleures performances pour des problèmes correspondant à un espace de recherche de très grande taille contenant peu de solutions.

[30] Constraint Satisfaction Problem ou problème de satisfaction des contraintes

Par ailleurs, les métaheuristiques ne garantissent pas l'optimalité des solutions qu'elles vont pouvoir trouver. Tandis que les procédures complètes peuvent se révéler incapables de trouver des solutions dont la qualité est proche de celle obtenue par les principales métaheuristiques, en particulier pour des problèmes simulant le monde réel qui atteignent souvent des niveaux particulièrement élevés de complexité.

D'autre part, un type d'objets doit être caractérisé afin d'être associé à un ensemble de contraintes. Et la représentation d'un type d'objets doit inclure sa position, son orientation et sa taille, ainsi que la définition des relations spatiales entre ce type d'objets et tout autre type d'objets pouvant être présent dans la scène.

Pendant la phase de génération, trouver l'emplacement des différents objets dans la scène est une étape très importante de la création d'une scène modèle. Les propriétés fournies par l'utilisateur, ou plus précisément obtenues à partir de DRAMAtexte sont interprétées en contraintes, qui doivent être vérifiées et résolues en assurant la configuration correcte de toutes les solutions proposées par le système de génération.

Pour réduire l'espace de recherche pendant la phase de génération, nous utilisons un algorithme capable de réduire l'espace de recherche susceptible de contenir des solutions. Et avant de passer à l'étape ultime de visualisation, nous appliquons des méthodes basées sur les métaheuristiques, après la réduction de l'espace de recherche, afin de résoudre le système de contraintes obtenu après l'interprétation des propriétés en contraintes.

Si une contrainte ne peut pas être satisfaite alors qu'elle correspond à une entité devant figurer dans la scène, la description initiale doit être modifiée par l'utilisateur et la résolution relancée. Ainsi, ces étapes se répètent selon un processus, classique en modélisation déclarative, de résolution en spirale jusqu'à ce qu'une solution satisfaisante soit trouvée ou jusqu'à ce qu'une condition d'arrêt soit atteinte.

III.3.3.1- Interprétation des propriétés en contraintes

L'interprétation des propriétés en contraintes consiste à transformer les propriétés obtenues à partir de DRAMAtexte en contraintes spatiales, pour permettre au système de génération de placer chaque objet dans la scène finale.

Dans [40], O. Le Roux a défini la propriété comme étant une entité générique descriptive qui se traduit en un ensemble de contraintes lorsqu'on l'associe avec des valeurs.

Ces contraintes dépendent souvent des caractéristiques intrinsèques de l'objet (propres à l'objet) et aussi à l'utilisation de l'objet. Lors de la phase de génération, nous devons tenir compte non seulement de l'orientation intrinsèque de l'objet, mais également du contexte de son utilisation.

Pour DRAMAscène, la taille, la position ainsi que l'orientation de chaque objet seront les données nécessaires pour calculer leur placement dans la scène. Nous avons conçu un modèle de base de connaissance qui va préciser les caractéristiques morphologiques et géométriques de chaque objet. Le tableau 8 montre l'exemple d'une propriété d'un objet « boîte », telle qu'elle est modélisée et stockée dans un fichier *XML* créé à partir du modèle de base de connaissances.

```
<OBJET NAME="CAISSE" MESH="box.mesh">
        <MATERIAL NAME="bois/bois2" />
        <REPERE>
                <ORIENTATION>
                        <ANGLE>1.0</ANGLE>
                        <A>0.0</A>
                        <B>0.0</B>
                        <C>0.0</C>
                </ORIENTATION>
                <ORIGIN>
                        <X>0.0</X>
                        <Y>0.0</Y>
                        <Z>0.0</Z>
                </ORIGIN>
          </REPERE>
          <VOLUME TYPE="BOITE">
                <M>4.0</M>
                <N>4.0</N>
                <P>4.0</P>
          </VOLUME>
          <TAILLE>
                <R>4.0</R>
                <S>4.0</S>
                <T>4.0</T>
          </TAILLE>
          <FRICTION>0.8</FRICTION>
          <RESTITUTION>0.1</RESTITUTION>
          <MASS>1</MASS>
          <SHADOW_CASTED>TRUE</SHADOW_CASTED>
</OBJET>
```

Tableau 8: Extrait du fichier contenant les éléments définissant les objets

Notons aussi au passage que les propriétés géométriques et physiques de chaque objet sont stockées dans un autre fichier *XML* dans lequel nous trouvons :

- le nom de l'objet et le nom du fichier contenant le maillage, et donc l'ensemble des informations utilisées pour visualiser l'objet, notamment la position de tous les sommets et triangles utilisés par le moteur graphique,
- le nom du fichier définissant les matériaux, contenant tous les paramètres de rendu de l'objet (textures, lumières, etc.),
- un repère orthonormé relatif rattaché par défaut à l'objet, qui nous permet à la fois de positionner et d'orienter un objet,
- le type et la taille du volume (boîte, capsule, cylindre, sphère, volume concave, volume convexe) englobant par défaut de l'objet,
- les dimensions de l'objet (selon les trois axes),
- les valeurs des forces et les paramètres physiques de l'objet tels que la masse, le coefficient de friction[31] et de restitution[32].

Dans l'exemple, montré dans le tableau ci-dessus, nous retrouvons :

- la balise orientation

```
<ORIENTATION>
        <ANGLE>1.0</ANGLE>
        <A>0.0</A>
        <B>0.0</B>
        <C>0.0</C>
</ORIENTATION>
```

  qui indique l'orientation par défaut de l'objet. En effet, les valeurs 1.0 et 0.0 se trouvant à l'intérieur des balises, <ANGLE>, <A>, <B> et <C> désignent les valeurs des angles que fait l'objet avec les trois axes du repère. Ici, ces valeurs signifient qu'aucune orientation par défaut n'est appliquée à l'objet.

- la balise origine

```
<ORIGIN>
        <X>0.0</X>
        <Y>0.0</Y>
        <Z>0.0</Z>
</ORIGIN>
```

  définissant les coordonnées de l'origine du repère par rapport à son centre de gravité.

[31] Rapport entre la force de frottement et la force de maintien entre deux surfaces en contact.
[32] Coefficient physique intervenant dans l'étude d'une collision

- la balise volume

```
<VOLUME TYPE="BOITE">
        <M>4.0</M>
        <N>4.0</N>
        <P>4.0</P>
</VOLUME>
```

caractérisant le volume par défaut englobant l'objet. <M>, <N>, <P> désignent la longueur, la largeur, et la hauteur d'un volume englobant.

- La balise taille

```
<TAILLE>
        <R>4.0</R>
        <S>4.0</S>
        <T>4.0</T>
</TAILLE>
```

utilisée pour gérer la mise en échelle de l'objet. Ici, la valeur 4.0, qu'on retrouve dans la balise <VOLUME> et <TAILLE>, signifie que l'objet et son volume englobant (donné précédemment) ont la même taille.

Pour procéder au placement des objets, nous avons utilisé les notions d'objet *cible*, correspondant à l'objet à placer, et objet *site*, qui est l'objet de référence pour le placement, conformément à la terminologie de Vandeloise [61] utilisé également dans [52]. Le but étant de placer ces objets en satisfaisant au mieux les propriétés identifiées dans le texte initial, nous avons également défini pour chaque type de contraintes autorisées pour ce type d'objets :

- un volume définissant la zone de placement par rapport à l'objet site,
- et un repère rattaché à l'objet site.

Les paramètres de ces volumes et repères sont stockés dans des fichiers XML.

Soit $V_i=\{Xmin_i, Ymin_i, Zmin_i, Xmax_i, Ymax_i, Zmax_i\}$ un ensemble définissant le volume de contrainte utilisé par un objet cible C_i suivant la contrainte ou relation spatiale R_i par rapport à un objet site S_i . Pour placer l'objet cible suivant les contraintes dans la description, il faut limiter la position du repère relatif rattaché à l'objet cible dans le volume de contrainte de l'objet site.

De ce fait, il faut placer d'abord le repère de contrainte par rapport au repère propre (rattaché) à l'objet site. Notons $o_i=(o_x, o_y, o_z)$ ce repère.

A partir de ce repère, nous pouvons tracer le volume de contrainte V_i défini précédemment, et délimité par les valeurs *min* et *max* sur les 3 axes X, Y et Z. Ainsi, pour placer l'objet cible, il faut déterminer la position de son point de repère propre pour qu'il soit à l'intérieur de ce volume. Afin de limiter l'orientation de l'objet cible, nous utilisons aussi des valeurs min et max pour les angles de rotation associés à chaque axe.

Ces contraintes sont ainsi traduites sous forme de système d'inéquations et donnent comme résultat la scène présentée dans la Figure 39, relativement à l'exemple de la Figure 38. Cet exemple illustre les volumes associés aux objets *cible* et *site*, relativement à la contrainte stipulant que « le fauteuil est devant le bureau ».

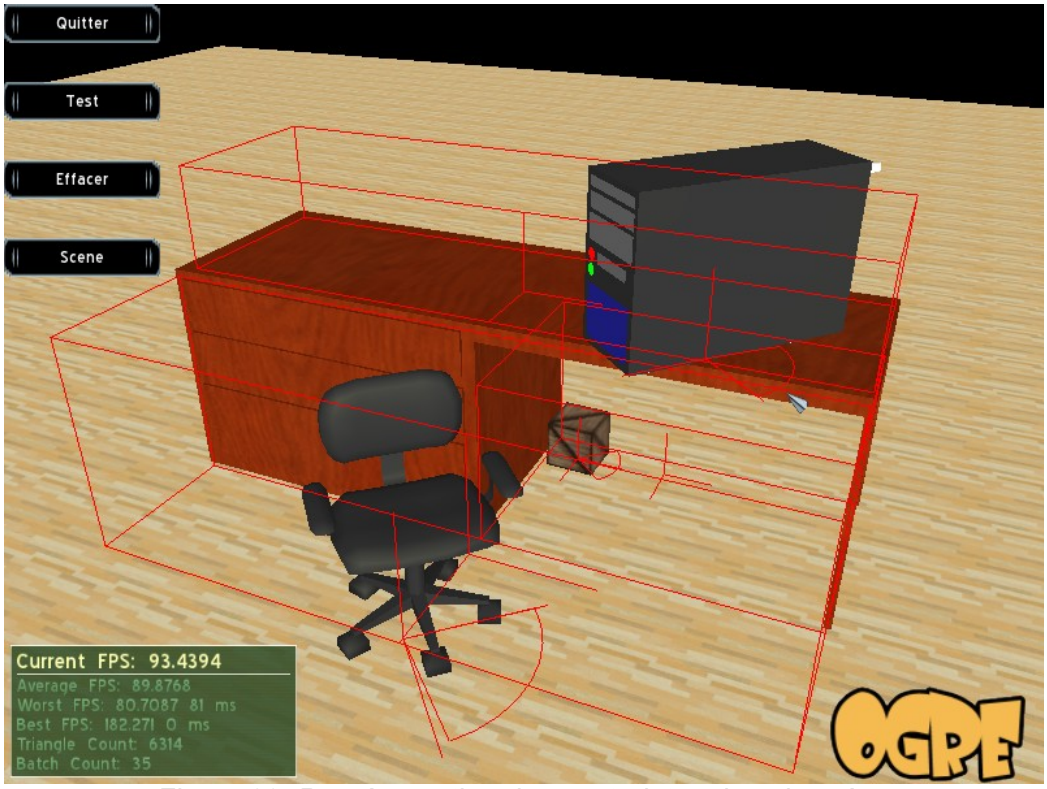


Figure 39: Représentation des contraintes dans la scène

Les valeurs des paramètres de ces contraintes sont récupérées à partir de deux fichiers *xml* qui stockent notamment le repère de contrainte rattaché à l'objet *cible* pour un placement par rapport à un objet *site*, et le volume de contrainte pour l'objet *site*. Le tableau 9 montre un exemple de ces paramètres de contraintes.

Les valeurs se trouvant dans la balise VOLUME définissent la dimension du volume de contrainte, utilisé par l'objet *site* "bureau" qui permet de placer l'objet *cible* "chaise" pour la contrainte de position "devant".

Ce volume de contrainte est défini par rapport au repère de contrainte dont les paramètres sont définis à l'intérieur de la balise <REPERE>.

ChaiseCible.xml	BureauSite.xml
<CIBLE_MANAGER> <CIBLE OBJET="CHAISE"> <REPERE_CONTRAINTE CONTRAINTE="DEVANT" SITE="BUREAU"> <REPERE> <ORIGIN> <X>1.0</X> <Y>2.0</Y> <Z>1.0</Z> </ORIGIN> <ORIENTATION> <W>1.0</W> <M>0.0</M> <N>0.0</N> <P>0.0</P> </ORIENTATION> </REPERE> </REPERE_CONTRAINTE> <CIBLE_MANAGER>	<SITE_MANAGER> <SITE OBJET="BUREAU"> <VOLUME_CONTRAINTE CONTRAINTE="DERRIERE" CIBLE="CHAISE"> <VOLUME> <XMIN>3.0</XMIN> <YMIN>2.0</YMIN> <ZMIN>10.0</ZMIN> <XMAX>9.0</XMAX> <YMAX>7.0</YMAX> <ZMAX>15.0</ZMAX> </VOLUME> </VOLUME_CONTRAINTE> </SITE> </SITE_MANAGER>

Tableau 9: Extrait des deux fichiers contenant les contraintes spatiales entre les objets

Pour satisfaire la contrainte, le repère relatif rattaché à l'objet *cible* doit se trouver dans le volume de contrainte défini pour l'objet *site*. Après avoir interprété les propriétés en valeurs numériques et géométriques, la phase de génération consiste à résoudre les systèmes d'équations et inéquations correspondant à ces contraintes.

III.3.3.2- Définition interactive des volumes et repères de contraintes

Pour définir graphiquement les repères et les volumes de contraintes que nous avons mentionnés précédemment, nous avons créée une interface graphique qui permet à

l'utilisateur de tracer directement les limites de placement et le repère correspondant à un objet, quand est joue le rôle de cible ou le rôle de site, et par rapport à une contrainte donnée.

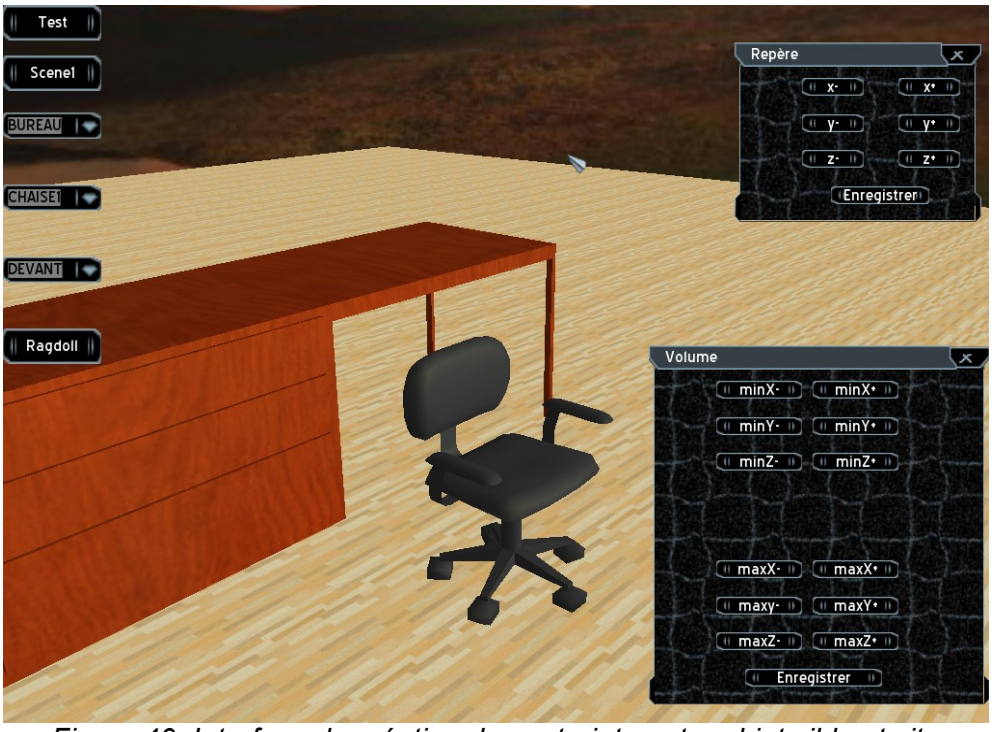


Figure 40: Interface de création de contrainte entre objet cible et site

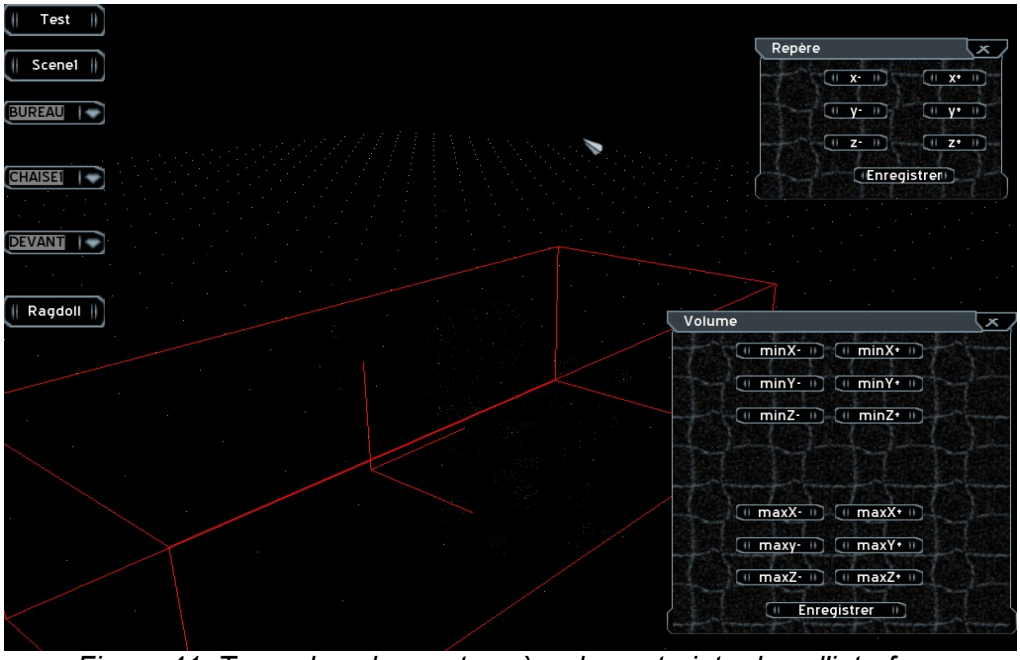


Figure 41: Trace du volume et repère de contrainte dans l'interface d'administration des contraintes

Dans l'exemple montré dans la figure 40, nous définissons les contraintes spatiales (volume et repère) qui correspondent à : « *chaise devant bureau* ». Nous commençons par associer un repère aux objets. Une fois que ce repère est mis en place, nous pouvons tracer les volumes (Figure 41) correspondant à la zone de placement possible d'une « chaise » devant un « bureau ».
Enfin, il ne nous reste plus qu'à sauvegarder ces paramètres de contraintes dans les fichiers XML des objets sites et objets cibles concernés.

III.3.4) Résolution des contraintes

La résolution des contraintes consiste à déterminer dans l'espace de recherche les scènes qui peuvent répondre aux critères décrits dans l'énoncé. Pour cela, il nous faut étudier les méthodes qui permettent de résoudre ce type de problèmes, c'est-à-dire des méthodes résolvant les contraintes que nous venons d'obtenir à l'issue de l'étape d'interprétation. Avant de présenter le résultat de notre étude, nous allons voir quelques définitions et notions de base sur les contraintes.

III.3.4.1- Définitions :

III.3.4.1.1) Contraintes

Une contrainte est une relation entre les inconnues ou variables d'un problème qui limite les combinaisons de valeurs qu'elles peuvent prendre simultanément. Prenons par exemple la contrainte de non-chevauchement de deux rectangles qui est une contrainte typique des applications graphiques basées sur un système de contraintes. Cette contrainte signifie simplement que les positions et dimensions des rectangles respectent les équations suivantes :

$$\begin{cases} x_1 + l_1 \leqslant x_2 \quad , ou \quad x_2 + l_2 \leqslant x_1 \\ y_1 + h_1 \leqslant y_2 \quad , ou \quad y_2 + h_2 \leqslant y_1 \end{cases}$$

avec (x_1, y_1) et (x_2, y_2) qui définissent les variables représentant les coordonnées du point inférieur-gauche et l_1, l_2, h_1 et h_2 qui sont les variables représentant la taille du rectangle (largeur, hauteur).

III.3.4.1.2) CSP

Un CSP[33], ou problème de satisfaction de contrainte est un problème consistant à modéliser un problème sous forme de contraintes posées sur des variables qui prennent des valeurs dans un domaine. Un CSP est donc défini par un triplet (X, D, C) tel que :

- $X=\{X_1, X_2, ..., X_i, ...X_n\}$ est l'ensemble des variables,
- $D=\{D_1, D_2, ..., D_i, ...D_n\}$ est l'ensemble des domaines et D_i représente l'ensemble des valeurs que peut prendre la variable X_i ,
- $C=\{C_1, C_2, ..., C_i, ...C_m\}$ est l'ensemble des contraintes mettant en relation les variables.

Le domaine des variables peut être discret et donc fini, ou continu et donc infini.

Comme exemple de domaines discrets nous pouvons donner :

- {rouge, vert, bleu} ou toute énumération d'éléments,
- {0, 1, 2, 3, 4, 5} des entiers naturels,
- { {a,b}, {a,c}, {b, c}, {a, b, c} } des ensembles,

Comme exemple de domaine continu nous pouvons donner :

- [0.0, 100.0] composé de l'intervalle borné par deux nombres réels.

III.3.4.1.3) Résolution et affectation

La résolution est définie par l'affectation de valeurs aux variables de telle sorte que toutes les contraintes soient satisfaites. Par ailleurs, quatre types d'affectations peuvent être effectués :

- affectation totale si on associe des valeurs à toutes les variables,
- partielle lorsque certaines variables seulement sont affectées,
- consistante, si toutes les contraintes sont vérifiées,
- inconsistante, dans le cas où au moins une contrainte n'est pas satisfaite.

Par suite, une solution à un CSP est une affectation complète des valeurs du domaine à des variables, de telle sorte que toutes les contraintes sont satisfaites, c'est-à-dire une affectation totale et consistante.

III.3.4.1.4) Modélisation d'un problème

Pour modéliser un problème, plusieurs étapes doivent être suivies :

[33] Constraint Satisfaction Problem

- identification des variables du problème, permettant de définir les inconnues,
- définition des domaines initiaux de valeurs de ces variables,
- spécification des contraintes.

Plusieurs modélisations possibles peuvent se présenter pour un problème, mais néanmoins nous pouvons retenir les critères suivants :

- la simplicité et homogénéité de l'expression des contraintes,
- l'efficacité, en tenant compte la taille de l'espace de recherche de solutions.

III.3.4.2- Contraintes au niveau de DRAMA

Une fois que toutes les propriétés ont été traitées et traduites en contraintes, le processus se poursuit avec la résolution de ces contraintes, et donc la vérification des positions de tous les éléments.

Ceci est géré dans un premier temps par un CSP permettant de modéliser les contraintes nécessaires au placement des objets.

L'origine $O=\{0.0,0.0,0.0\}$ du repère orthonormé que nous avons adopté se trouve au centre de la scène. Comme chaque contrainte obtenue à partir de la description met en relation deux objets à chaque situation, les contraintes que nous avons à manipuler sont donc des contraintes binaires.

Concernant le problème traité dans le projet DRAMA, le modèle CSP utilisé est constitué par :

- un ensemble de variable $X=\{X_1,X_2,...,X_i,...,X_n\}$ obtenues à partir des propriétés existant sur les entités à placer sur la scène, chacune représentée par les données géométriques suivantes : $X_i=\{P_i,O_i,S_i\}$ où $P_i=\{x_i,y_i,z_i\}$, $O_i=\{\alpha_i,\beta_i,\gamma_i\}$ $S_i=\{S_{xi},S_{yi},S_{zi}\}$ qui correspondent respectivement à la position, l'orientation et la dimension de cette entité.
- des domaines des variables, ou ensemble des valeurs que peuvent prendre les variables, sont définis comme suit $D(P_i)=[-\infty,+\infty]$, $D(O_i)=[0.0,2\Pi]$, $D(S_i)=[0.0,+\infty]$ avec $D(P_i),D(O_i),D(S_i)$ quelle que soit la variable *Xi* appartenant à l'ensemble *X*,
- un ensemble de contraintes spatiales mettant en relation deux à deux les entités dans la scène.

III.3.4.2.1) Contrainte de non-chevauchement

La contrainte de non-chevauchement correspond à la contrainte de non collision entre les différentes entités. Afin de détecter les collisions entre les entités, nous avons utilisé des sphères comme volumes englobant les objets. Ainsi, la détermination de la collision entre deux objets sera obtenue simplement et rapidement à partir du calcul de la collision entre deux sphères. Ce calcul fournit une approximation, mais est suffisant pour éliminer des domaines des variables des valeurs où la collision est quasi certaine.

Deux sphères entrent en contact si et seulement si la distance entre leurs centres est inférieure à la somme de leurs rayons. Autrement dit, soient $S_i(O_i, R_i)$ et $S_j(O_j, R_j)$ deux sphères ayant comme centre les points $O_i(xo_i, yo_i, zo_i)$ et $O_j(xo_j, yo_j, zo_j)$,

$$d(O_i, O_j) = \|\overrightarrow{O_i O_j}\| = \sqrt{(xo_j - xo_i)^2 + (yo_j - yo_i)^2 + (zo_j - zo_i)^2} \leqslant (R_i + R_j) \quad ,$$

où $d(O_i, O_j)$ représente la distance entre les centres des deux sphères.

III.3.4.2.2) Arithmétique d'intervalles

Avant de procéder à la résolution proprement dite, il s'avère nécessaire de présenter brièvement la notion d'arithmétique d'intervalles (qui est plus largement présentée dans [57]), car les domaines de variables de notre problème sont continus. En effet, l'arithmétique des intervalles est une méthode de calcul appliquée sur des intervalles qui a l'avantage d'être peu coûteuse en temps de calcul et de permettre l'encadrement des erreurs produites par les calculs arithmétiques sur les domaines.

Pour cela, nous allons voir ci-après quelques propriétés basiques importantes de l'arithmétique d'intervalles. Soient +, −, * et / respectivement l'addition, la soustraction, la multiplication et la division, opérations appliquées sur des intervalles.

Alors, si $X = [a, b] \subset \mathbb{R}$ et $Y = [c, d] \subset \mathbb{R}$ on a :

$$X + Y = [a + c, b + d]$$

$$X - Y = [a - d, b - c]$$

$$X * Y = [min(a*c, a*d, b*c, b*d), max(a*c, a*d, b*c, b*d)]$$

$$1/Y = [1/d, 1/c] \quad \text{si} \quad 0 \notin Y$$

$$X/Y = X * (1/Y) \quad \text{si} \quad 0 \notin Y$$

Ces différents calculs nous seront utiles dans chaque étape et procédure de résolution de contrainte étant donné que sans l'utilisation de l'arithmétique d'intervalles on aurait une infinité d'affectations et de valeurs possibles pour chaque variable.

L'espace de recherche est ainsi défini par les intervalles associés à chacune des variables du problème. La phase de recherche de solutions se fait ainsi en deux étapes :

- réduction des domaines contenant les valeurs,
- résolution du problème par recherche de solutions dans les domaines réduits.

III.3.4.2.3) Filtrage des domaines

Afin de gagner en performance en termes de temps de traitement, il faut réduire les domaines initiaux des variables en éliminant les parties dont on est certains qu'ils ne contiennent pas de valeurs conduisant à des solutions. Pour cela, nous avons adopté des méthodes de filtrage issues des CSP et applicables à notre problème. Par suite, la réduction des problèmes par filtrage repose sur des propriétés de consistance (ou cohérence) partielle : degré de compatibilité entre valeurs des domaines et contraintes.

En effet, un CSP (X, D, C) est dit consistant d'arc si pour tout couple de variables (X_i, X_j) de X, et pour toute $v_i \in D(X_i)$ il existe une valeur v_j appartenant à $D(X_j)$ telle que l'affectation partielle $\{(X_i, v_i), (X_j, v_j)\}$ satisfait toutes les contraintes binaires de C.

Pour mieux appréhender le déroulement du principe de filtrage, nous allons prendre un exemple de problème de placement d'objet suivant la description suivante :

« Placer l'ordinateur sur le bureau. Placer le téléphone à droite de l'ordinateur. Placer la lampe à gauche de l'ordinateur. ».

Les variables du problème sont ici :

- Un point caractéristique de l'ordinateur.
- Le volume associé au bureau relatif à la contrainte « sur ».
- Un point caractéristique du téléphone.
- Le volume associé à l'ordinateur relatif à la contrainte « à droite ».
- Un point caractéristique de la lampe.
- Le volume associé à l'ordinateur relatif à la contrainte « à gauche ».

Ces variables sont associées à des domaines de valeurs, eux-mêmes définis par des variables réelles auxquelles sont associés des intervalles.

Le domaine initial du point caractéristique P_{ordi} de l'ordinateur correspond à un volume de placement défini par le produit cartésien des intervalles associés aux trois coordonnées définissant P_{ordi}. Ces coordonnées ont comme domaine initial $D(P_i)=[-\infty,+\infty]$.

La première étape de filtrage consiste à déterminer le domaine dans lequel P_{ordi} peut être instancié, tout en conservant la consistance de la contrainte « *Placer l'ordinateur sur le bureau.* ». Le domaine, après cette étape de filtrage, peut être visualisé comme sur la Figure 42. Le point caractéristique pourra être instancié n'importe où dans le domaine, permettant le placement de l'ordinateur sur la surface du bureau.

Vue de dessus :

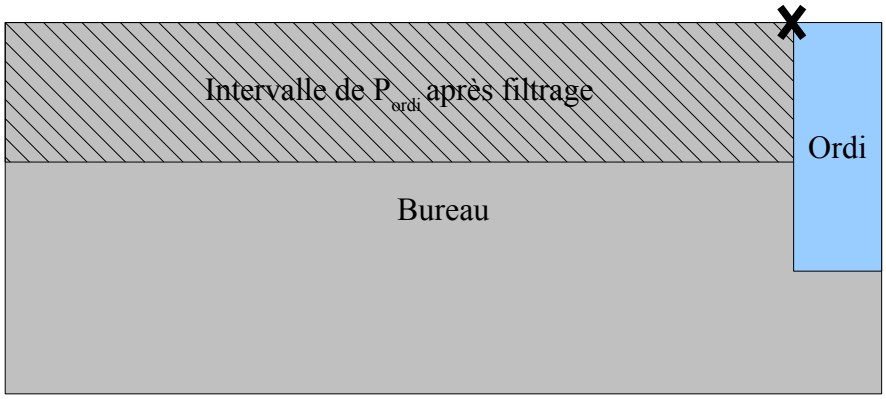


Figure 42: Domaine du point P_{ordi} après la première étape de filtrage

Vue de dessus :

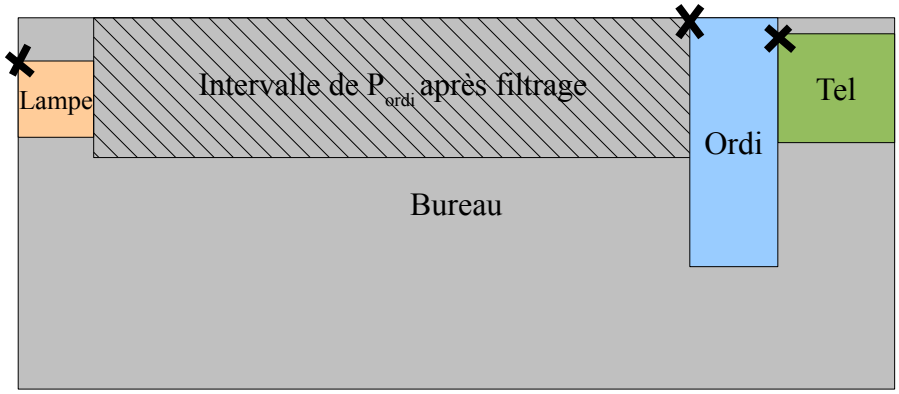


Figure 43: Domaine du point P_{ordi} après filtrage

La prise en compte des propriétés *« Placer le téléphone à droite de l'ordinateur. Placer la lampe à gauche de l'ordinateur. »* va réduire le domaine de P_{ordi} car le filtrage devra tenir

compte du volume nécessaire pour pouvoir placer les deux objets sur le bureau, avec le téléphone à sa droite et la lampe à sa gauche.

Le résultat après ces étapes de filtrage peut être visualisé comme sur la Figure 43. On constate que le domaine permet une instanciation de P_{ordi} qui laisse dans tous les cas la possibilité de placer la lampe à gauche et le téléphone à la droite de l'ordinateur.

Le nouveau CSP obtenu après cet étape de filtrage est inclus dans le CSP initial, c'est-à-dire que son espace de recherche a été réduit tout en conservant l'ensemble des solutions valides. En effet, seules les valeurs dont on est sûrs qu'elles ne permettaient pas le placement des trois objets sans collision ont été supprimées du domaine associé à P_{ordi}.

Après chaque modification ou réduction du domaine D_i d'une variable X_i par la procédure de filtrage, il faut répercuter cette modification et tester si les domaines des variables qui sont contraintes par rapport à X_i sont toujours arc-consistants. Cette procédure sera répétée pour les autres variables du problème, et selon la façon dont les objets sont contraints entre-eux, nous obtiendrons un espace de recherche plus réduit que celui défini par les domaines initiaux.

L'ordre de choix des variables peut être pris en compte pendant l'étape de filtrage des domaines, afin de prévoir le plus tôt possible l'inexistence de solutions. En effet, quand un domaine devient vide pendant l'étape de filtrage, cela signifie qu'aucune solution respectant les contraintes ne pourra être trouvée.

Toutefois, la procédure de filtrage n'est pas suffisante pour résoudre des contraintes. Car selon la façon dont les objets sont contraints entre-eux, il peut rester dans les domaines des valeurs qui ne seront pas solution.

Dans l'exemple précédent, si l'énoncé avait été : *« Placer l'ordinateur sur le bureau. Placer le téléphone sur le bureau. Placer la lampe sur le bureau. ».* Le filtrage n'aurait pas réduit les domaines de ces trois objets non contraints entre-eux, et des valeurs entrainant des collisions après instanciation se trouveraient encore dans les domaines.

III.3.4.2.4) Métaheuristiques

Dans le cas de problèmes fortement contraints, les méthodes exactes telles que les CSP peuvent engendrer des difficultés, car après l'étape de filtrage, le temps nécessaire pour trouver des solutions peut croître exponentiellement avec la taille du problème. Ainsi, on risque de ne pas trouver une solution satisfaisant l'ensemble des contraintes dans un temps « raisonnable ».

Pour remédier à ce problème, les métaheuristiques nous offrent la possibilité d'obtenir une solution qui peut ne pas être optimale, afin d'accélérer le temps de résolution du problème. Donc, à la fin de la procédure de filtrage, la recherche de solutions proches de l'optimum, qui ne satisfont pas forcément toutes les contraintes, peut se faire en un temps correct pour l'utilisateur, grâce à l'utilisation de métaheuristiques. Une étude des techniques de résolutions basées sur les métaheuristiques pourra être trouvée dans [38].

Pour procéder à la recherche de solutions, les métaheuristiques utilisent une fonction appelée « fonction objectif » ou parfois aussi « fonction de coût », qu'on doit minimiser à chaque étape du traitement pour évaluer la qualité de la solution obtenue.

En effet, dans un problème Np-difficile, il est possible de rencontrer des valeurs ou des solutions voisines de la solution exacte, comme l'illustre la figure 44.

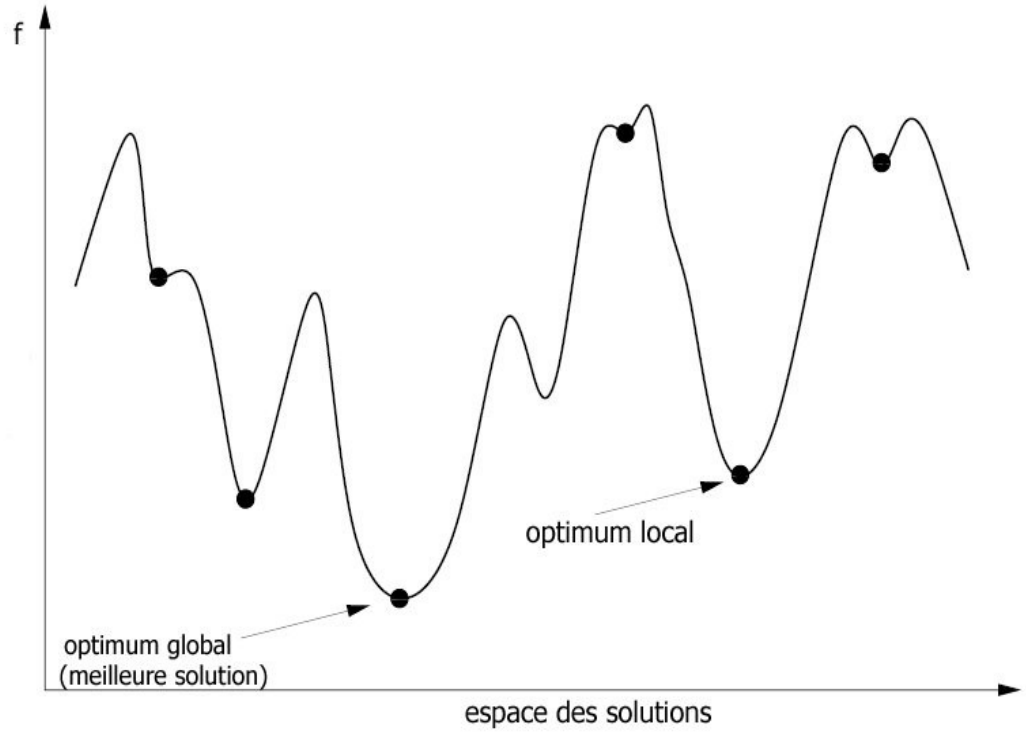


Figure 44: Représentation de l'espace des solutions à partir de la fonction objectif

Ce qu'il faut éviter avec les métaheuristiques, c'est d'arrêter la recherche sur une solution loin d'une solution qui est un optimum local. Il faut disposer de mécanismes permettant de continuer l'exploration de la solution jusqu'à ce qu'on ait atteint un objectif donné, ou qu'on ait atteint la meilleure solution dans un temps limite fixé a priori.

La fonction de coût *f* que nous avons choisie a comme objectif principal de minimiser la somme des distances (ou profondeur de collision) entre les différentes entités, afin d'optimiser autant que possible le placement de chaque entité. Ainsi, nous avons vu précédemment que la contrainte de collision est vérifiée si :

$$d(O_i, O_j) = \sqrt{(xo_j - xo_i)^2 + (yo_j - yo_i)^2 + (zo_j - zo_i)^2} \leqslant (R_i + R_j)$$

tel que $O_i et O_j$ représentent les centres des sphères enveloppant les objets à placer, et $R_i et R_j$ représentent leurs rayons respectifs.

La fonction coût est alors définie comme suit :

$$f : D_1 \times D_2 \times \ldots \times D_n \to \Re$$

$$S \to f(S) = \sum (|d(O_i, O_j) - (R_i + R_j)|)$$

Dans le cas du projet DRAMA, nous avons utilisé une métaheuristique appelée, « recuit simulé ». Cette méthode se base sur le processus utilisé depuis longtemps dans la métallurgie, qui consiste à chauffer le morceau de métal avant de le laisser se refroidir très lentement pour l'obtention d'un alliage de qualité. L'algorithme de cette méthode est décrit comme suit :

Procédure *recuit_simulé* (solution initiale *s*)

1 : Poser *T* ← *T0* //Initialiser la température

2 : Répéter :

3 : Choisir aléatoirement *s'= ϖοισιν δε s*

4 : Générer un nombre réel aléatoire *r* dans [0,1]

5 : Si $r < e^{\frac{f(s)-f(s')}{T}}$ alors Poser *s* ← *s'*

6 : Décroître *T* *//On peut faire T = T x α , ou α < 1*

7 : Jusqu'à ce que T=0

8 : Fin

III.3.5) Instanciation de la scène

L'instanciation de la scène consiste à obtenir les valeurs contenues dans la solution optimale obtenue précédemment et de les affecter à des variables de position et d'orientation pour définir le placement et les contraintes géométriques de chaque entité.

Une fois les variables instanciées, les entités peuvent être placées dans la scène.

Nous avons utilisé un moteur physique appelé « Bullet » pour gérer les interactions physiques entre les objets, et plus particulièrement les collisions. En effet, Bullet peut gérer les collisions grâce à l'algorithme appelé GJK et développé par Gilbert, Johnson et Keerthi [23]. Grâce à cet algorithme, les objets qui sont en collision sont disposés automatiquement dans la scène de façon à ne plus être en collision.

De plus, ces placements peuvent être aussi soumis à des contraintes correspondant à la description, ce qui correspond tout à fait à notre attente.

En effet, à partir des descriptions stockées dans notre base de connaissance (au format XML), nous pouvons récupérer les données correspondant aux volumes et aux repères définissant les contraintes associées à chaque entité. Une fois que ces paramètres sont disponibles, nous pouvons créer des liaisons entre les différents objets qui sont contraints entre-eux.

Avec le moteur physique Bullet, nous pouvons gérer jusqu'à 6 degrés de liberté (voir Figure 45) qui permettent de limiter facilement les amplitudes de translations et de rotations autour des trois axes. De ce fait, les zones ou volumes correspondant aux contraintes peuvent donc être définis en utilisant des fonctions disponibles dans Bullet.

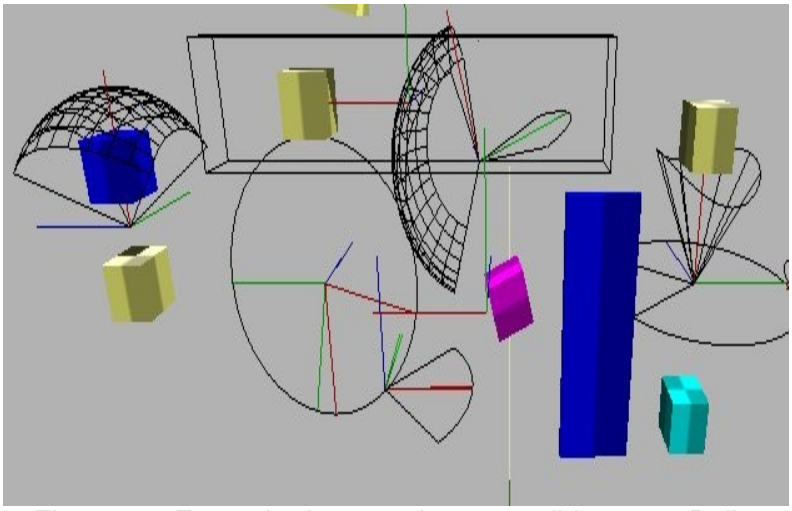


Figure 45: Exemple de contraintes possibles avec Bullet

En résumé, une fois que toutes les variables caractérisant le placement des entités dans la scène sont instanciées, nous obtenons un système « masse-ressort » qui est géré par Bullet.

Ceci permet au système de se mettre à jour automatiquement quand nous modifions la configuration de la scène ou quand nous ajoutons des nouvelles contraintes.
L'étape suivante est la visualisation de la scène résultat, obtenue après le filtrage, la résolution des contraintes et l'instanciation des variables du problème.

III.3.6) Affichage des résultats

III.3.6.1- Moteur 3D Ogre

Afin que l'utilisateur puisse choisir la scène qu'il juge la plus proche de la scène décrite dans le texte, nous avons également développé dans le projet DRAMA un module qui permet d'afficher la scène résultant de la description en trois dimensions. Pour cela, nous avons eu besoin d'un moteur de rendu 3D qui nous permette à la fois de visualiser et aussi de naviguer à l'intérieur de notre scène finale.
En fait, un moteur de rendu 3D est une bibliothèque, constituée par une série de classes qui nous offre la possibilité de réaliser des applications permettant d'afficher des scènes 3D complexes. Un moteur 3D est une couche logicielle (ou interface externe) qui permet de s'interfacer avec des interfaces de programmation de plus bas niveau d'abstraction (API 3D : OpenGL ou DirectX) .
Grâce aux fonctions déjà incluses dans le moteur 3D, nous pouvons réaliser facilement ce qu'il aurait été fastidieux de programmer avec les API. Donc le moteur 3D nous sera utile pour gérer tout l'aspect visuel dans notre scène 3D, ainsi que la gestion des événements permettant à l'utilisateur d'interagir avec la scène 3D.
Dans le cas de notre projet, nous avons donc opté pour le moteur « Ogre »[34] vu les divers avantages qu'il peut nous offrir. Effectivement, Ogre un moteur très complet qui a beaucoup de fonctionnalités et qui possède des modules additionnels vraiment intéressants et évolutifs. Il est libre et peut être installé sur tous les systèmes grâce à un rendu en DirectX ou OpenGL et il est aussi fourni avec les codes sources, ce qui le rend facilement adaptable à tous nos besoins.
L'un des points forts de Ogre est aussi sa façon de gérer les objets 3D. En effet, avec cette bibliothèque, nous avons pu mettre en place une banque d'objets 3D qui stocke séparément les maillages dans des fichiers contenant les coordonnées des sommets qui correspondent à la géométrie de l'objet, et les fichiers matériaux qui caractérisent toutes

[34] http://www.ogre3d.org

les propriétés visuelles de l'objet. Ainsi, grâce à cet aspect multi-couche des objets dans Ogre, nous pouvons personnaliser les objets ou introduire parmi les contraintes existantes des propriétés non géométriques telles que les textures, les couleurs, etc.

Et enfin l'avantage majeur est qu'il s'associe à Bullet grâce à un module appelé OgreBullet qui nous permet d'accéder directement aux classes et méthodes de Bullet à partir de Ogre.

Néanmoins, l'un des gros inconvénients de Ogre est sa complexité. C'est un gros moteur de rendu et donc moins simple à manipuler que d'autre moteurs 3D plus adaptés à de petits projets. Il n'est pas très simple à prendre en main et demande beaucoup de temps d'apprentissage avant de pouvoir écrire un programme permettant de visualiser une scène 3D.

Après avoir écrit une application intégrant simultanément Ogre et Bullet, nous avons mis en place la possibilité de représenter visuellement les contraintes entre les objets pour avoir une confirmation visuelle de la satisfaction des contraintes. Cela nous a également permis de développer une autre interface utilisateur qui permet de créer les contraintes de placement entre deux objets, de façon simple et graphique, directement à partir des résultats de DRAMAscène.

III.3.7) Évaluation des résultats

Afin de vérifier l'efficacité des méthodes que nous avons adoptées, nous proposons l'expérience suivante inspirée par les travaux d'Olivier le Roux. Ainsi, le placement de *n* objets dans un plan peut être simulé par le placement de *n* disques se trouvant dans un espace 2D rectangulaire de longueur w et de largeur L, sachant que chaque disque représente la projection au sol du volume d'un objet.

Les objets, ou plutôt les disques, ne doivent pas se chevaucher, ni dépasser le bord de la scène, soit la limite de l'espace rectangulaire. Ce problème peut être formulé comme suit. En prenant le point milieu du rectangle représentant la surface de la scène (w, L) comme origine de notre système de coordonnées cartésiennes, les coordonnées du centre et le rayon du i-ème disque sont désignés respectivement par (x_i, y_i) et R_i .

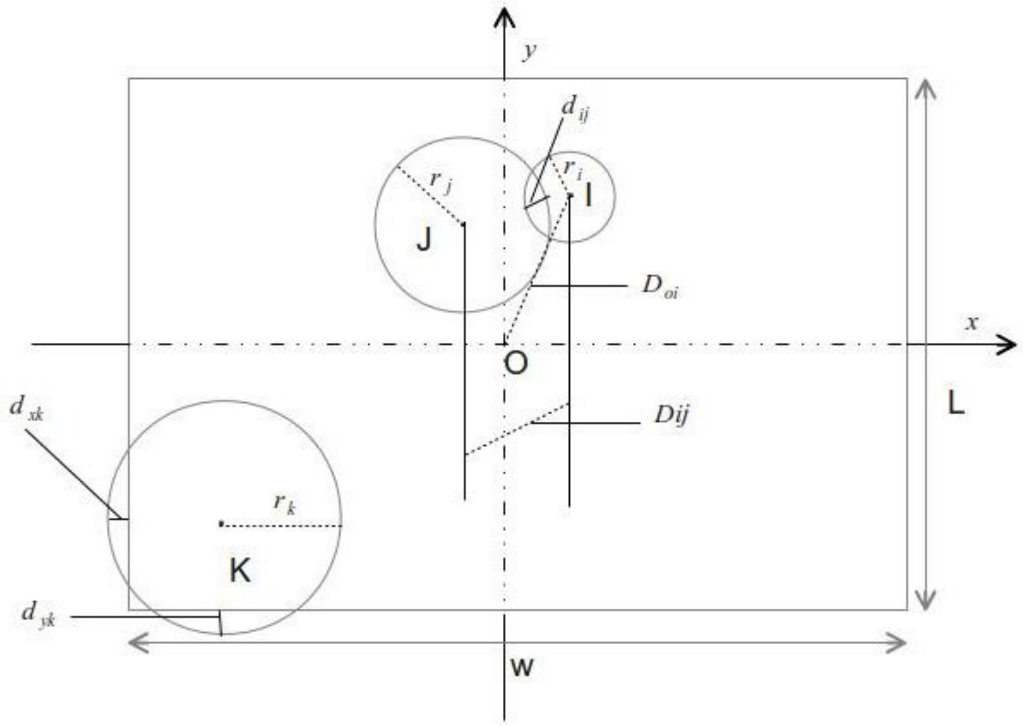


Figure 46: Présentation des contraintes de chevauchement entre les disques, ainsi qu'entre les disques et les bordures

Chaque disque doit respecter les conditions suivantes :

$$\begin{cases} r_k - \frac{w}{2} < x_k < \frac{w}{2} - r_k \\ r_k - \frac{L}{2} < y_i < \frac{L}{2} - r_k \\ d_{ij} = d(c_i, c_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < r_i + r_j, \quad tels\,que \quad i, j, k = 1,2,\ldots,N \; et \; i \neq j \end{cases}$$

Ainsi, pour éviter la collision entre le disque I et J, il faudrait donc annuler la distance de pénétration d_{ij} entre les deux disques. Dans le monde simulé par Bullet, cette collision provoquera une force élastique qui repoussera les deux disques. Notre problème devient donc un problème d'optimisation, dont l'objectif sera de minimiser la fonction d'énergie potentielle :

$$E(X) = E(x_1, y_1, x_2, y_2, \ldots, y_n, x_n) = \sum_{j=1}^{N} E_j$$

où,

$$Ej = \sum_{i=0\,;\,j\neq i}^{n} e_{ij}, \quad j = 1,2,\ldots,n \quad e_{ij} = d_{ij}^2$$

E_j désigne l'énergie potentielle obtenue par la collision (ou chevauchement) d'un objet (disque) avec les autres objets, ainsi qu'avec les bordures.

La distance de chevauchement (pénétration) d_{ij} , entre deux disques peut être calculée à partir du système d'équations suivant :

$$\begin{cases} d_{ij} = r_i + r_j - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, si\ D_{ij} < r_i + r_j \\ d_{ij} = 0, sinon \end{cases}$$

tel que, $D_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ désigne la distance séparant les centres des deux disques I et J.

La distance de débordement, que nous notons par d_{dk} , du disque k par rapport à la bordure est obtenue par :

$$d_{dk} = \sqrt{{d_{xk}}^2 + {d_{yk}}^2}$$

tels que, d_{xk} et d_{xk} soient les projections de d_{dk} sur l'axe des x et des y.

$$\begin{cases} d_{xk} = r_k - \frac{w}{2} - x_k, si \quad x_k < r_k - \frac{w}{2} \\ d_{xk} = -x_k + \frac{w}{2} - r_k, si\ x_k > \frac{w}{2} - r_k \\ 0, sinon \end{cases}$$

$$\begin{cases} d_{yk} = r_k - \frac{L}{2} - y_k, si \quad y_k < r_k - \frac{L}{2} \\ d_{yk} = -y_k + \frac{L}{2} - r_k, si\ y_k > \frac{L}{2} - r_k \\ 0, sinon \end{cases}$$

Si nous appliquons la méthode de recuit simulé pour résoudre ce problème, nous suivons l'algorithme suivant :

1) Générer une configuration initiale X, et initialiser la température T_0 ,
2) Générer une nouvelle configuration X' , soit $dE = E(X') - E(X)$,
3) Si $dE < 0$, exécuter 4), sinon,

   si $e^{\frac{-dE}{T_0}} \leqslant \lambda$ (λ un nombre aléatoire compris entre 0 et 1), aller vers 2),
4) $X = X', E(X) = E(X')$,
5) Vérifier si la valeur minimale de la somme de l'énergie potentielle est obtenue, sinon retourner au point 2),
6) Diminuer la température $T_0 = \alpha * T_0$. Si la température finale est atteinte, alors arrête ; sinon, revenir en 2).

La nouvelle configuration X' (voisinage de X) est obtenue en effectuant des modifications dans l'ancienne configuration X de solutions. Ces modifications nous permettront de déplacer les disques qui se sont chevauchés.

En prenant l'exemple montré dans la figure 46, nous pouvons déplacer le disque I suivant un vecteur de translation colinéaire à $\vec{IJ}$, dont les composantes, $d_{xj} et d_{yj}$, sont calculées comme suit :

$$d_{xj} = \frac{(x_i - x_j)}{D_{ij}} * d_{ij} \quad \text{et} \quad d_{yj} = \frac{(y_i - y_j)}{D_{ij}} * d_{ij}$$

En effet, d_{ij} désigne toujours la distance de pénétration comme nous l'avons définie plus haut. d_{xj} est la projection de d_{ij} sur l'axe des x et d_{yj} sa projection sur l'axe des y. Soit θ l'angle entre l'axe des abscisses (axe des x) et $\vec{IJ}$, nous obtenons :

$$\cos(\Theta) = \frac{(x_i - x_j)}{D_{ij}} = \frac{d_{xj}}{d_{ij}} \quad \text{et} \quad \sin(\Theta) = \frac{(y_i - y_j)}{D_{ij}} = \frac{d_{yj}}{d_{ij}}$$

Ce qui nous donne les valeurs de d_{xj} et d_{yj} ci-dessus. Pour un disque k qui dépasse la bordure de la scène, il faut lui appliquer une translation suivant un vecteur de composantes d_{xk} et d_{yk} défini par :

$$\begin{cases} d_{xk} = r_k - \frac{w}{2} - x_k & , si\ x_k < r_k - \frac{w}{2} \\ d_{xk} = \frac{w}{2} - x_k - r_k & , si\ x_k > \frac{w}{2} - r_k \\ d_{yk} = r_k - \frac{L}{2} - y_k & , si\ y_k < r_k - \frac{L}{2} \\ d_{yk} = \frac{L}{2} - y_k - r_k & , si\ y_k > \frac{L}{2} - r_k \end{cases}$$

Pour choisir les objets (disques) à déplacer, il est préférable de traiter en premier les objets de petite taille (disques de petit rayon dans notre exemple) pour accélérer la vitesse de traitement du problème de placement.

III.3.7.1- Évaluation

Pour évaluer la performance de la méthode que nous avons proposée précédemment, nous avons effectué des tests sur une machine avec un processeur Intel Core 2 Duo cadencé à 2,6 Ghz et 2Go de RAM. Le tableau suivant montre les résultats des différentes simulations que nous avons réalisées.

Nombre d'objet	5	15	30	50	75	90	100
Saturation (%)	3.927	11.781	23.561	39.263	58.904	70.68	78.54
Énergie finale	0	0	9.87E-4	0.851	3.701	6.432	11.761
Temps (ms)	1	1	61	119	175	190	264

Tableau 10: Résultat du placement d'objets soumis aux seules contraintes de non-collision

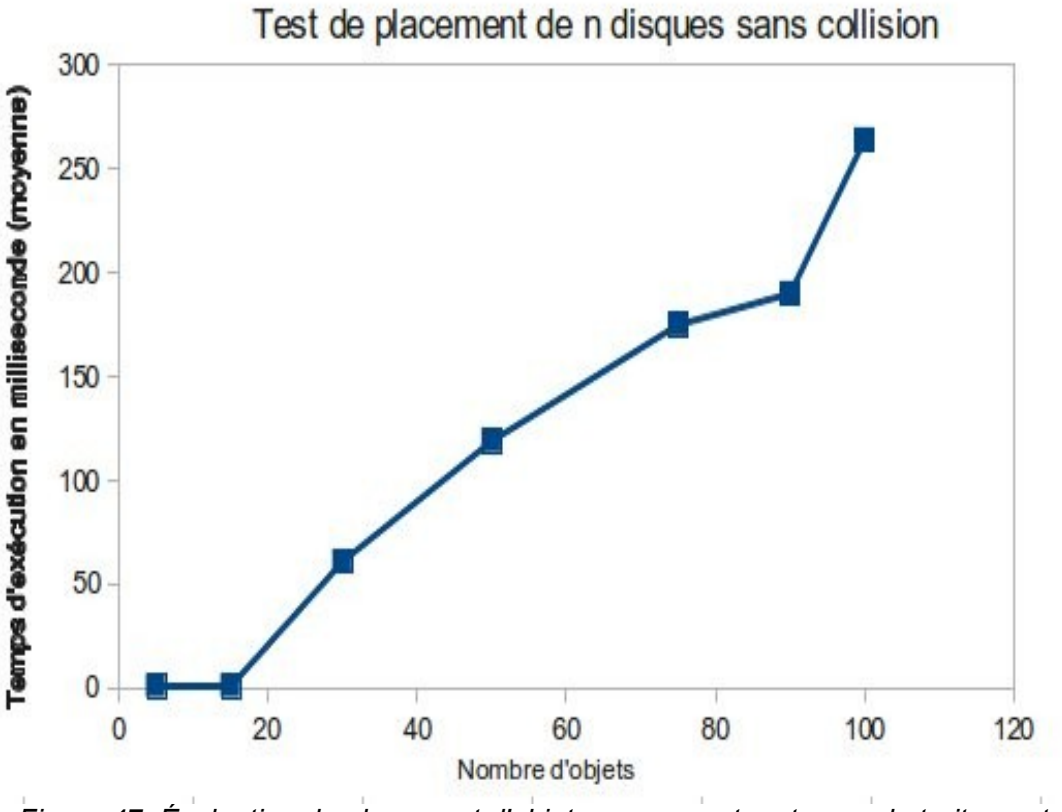


Figure 47: Évaluation du placement d'objets par rapport au temps de traitement

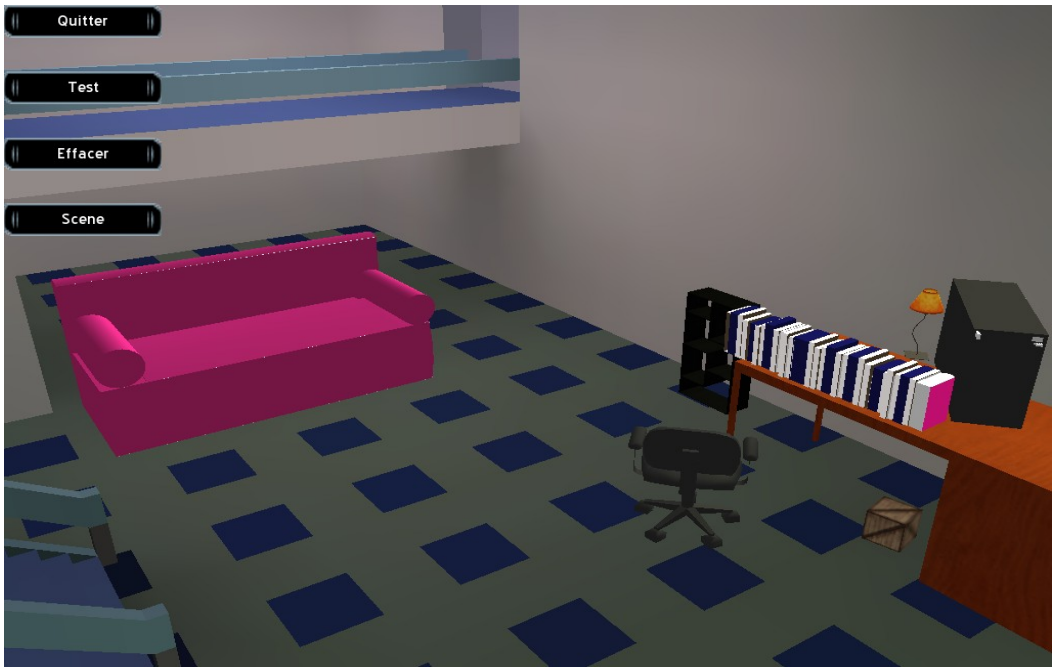


Figure 48: Exemple de scène finale générée

III.3.7.2- Scène résultat

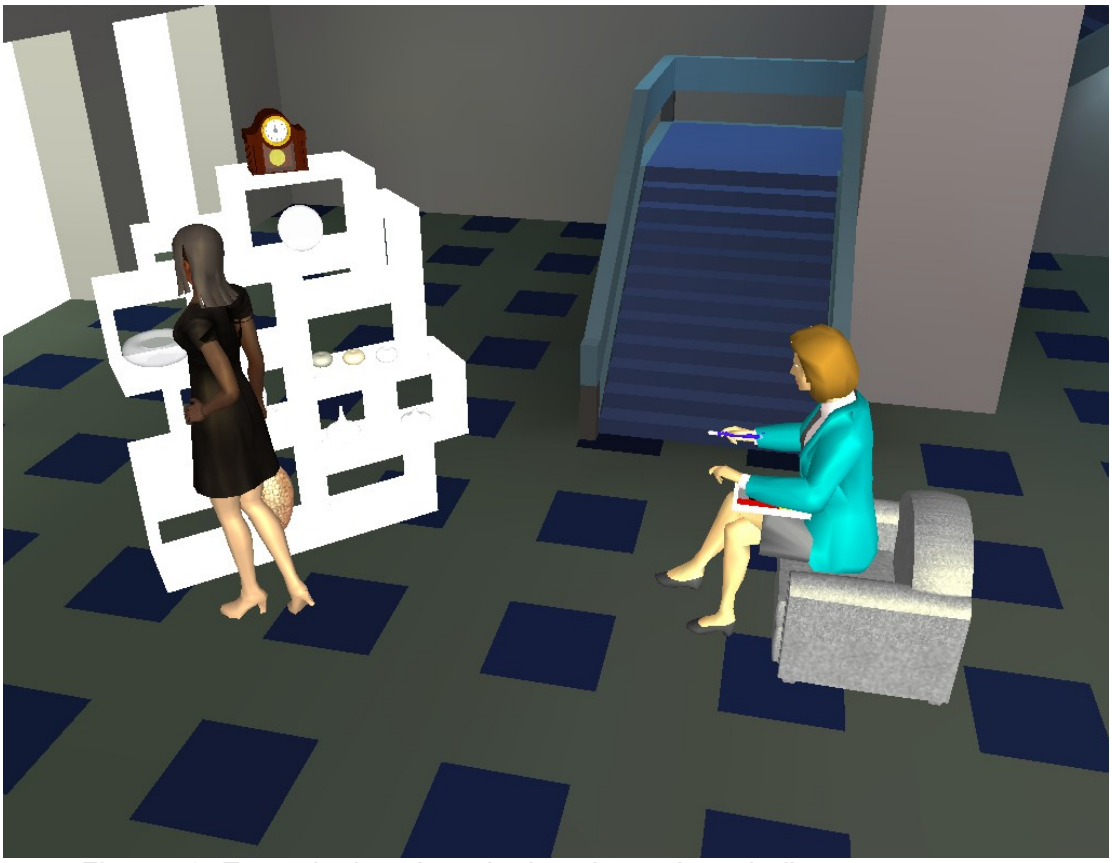


Figure 49: Exemple de scène résultat obtenu à partir d'un texte

La scène que nous présentons ci-dessus (Figure 49) a été générée à partir de la didascalie vue dans Texte 1. La version balisée (Texte 4) obtenue à partir de cette didascalie nous a permis de déduire :

- le type de l'environnement (<MSP> salon bourgeois </MSP>),
- la liste des personnages (<ACH>Mrs Peacock </ACH>,<ACH>Rose</ACH>),
- la liste des accessoires et objets (fauteuil, stylo plume, pendule à coucou, etc.),
- ainsi que d'autres informations telles que les propriétés descriptives (vielle dame, <MPOS>assise </MPOS>dans son <MS TYPE= " Fixed">fauteuil</MS>, etc.)

III.4- Conclusion

Dans cette partie, nous avons présenté les méthodes qui nous ont permis de traiter les propriétés contenues dans le texte descriptif pour produire et résoudre les contraintes. Nous avons détaillé chaque étape indispensable pour atteindre notre objectif initial, en commençant par la création des balises dédiées aux textes de théâtre, et jusqu'à l'affichage des scènes finales.

Nous avons également introduit des notions sur la spécification et le traitement de contraintes, utiles pour la résolution et le placement de chaque entité, de façon à respecter les contraintes. Au travers des études et des résultats obtenus, nous avons pu constater que la résolution proprement dite peut ne pas être adaptée à la nature du problème si on utilise seulement les CSP.

Mais nous conservons l'étape de filtrage qui est nécessaire afin d’éliminer dans les domaines les valeurs qui ne conduisent pas à des solutions, et qui permet ainsi d'accélérer l'étape suivante d'instanciation d'une solution.

Partie IV- Conclusion générale

IV.1- Conclusions et perspectives

Le travail présenté dans cette thèse a été guidé, du début à la fin, par le concept de génération de scènes virtuelles 3D à partir de descriptions extraites de textes de théâtre, et ceci dans le cadre du projet DRAMA.

L'objectif de cette thèse était d'étudier et de proposer des techniques informatiques facilitant le travail de mise en scène à partir d'un texte théâtral, pour permettre ensuite la génération de scènes virtuelles à partir d'éléments de description.

Cet objectif a été atteint dans le sens où toutes les briques de l'architecture de DRAMA ont été conçues et développées. Même s'il est certain que l'ergonomie de l'outil développé reste encore à améliorer et devra faire l'objet de travaux futurs, en relation avec les utilisateurs finaux et avec des personnes spécialistes des interfaces homme-machine.

Pendant cette étude, nous sommes intéressés dans un premier temps aux travaux antérieurs ayant un rapport avec notre projet. Ainsi, nous avons passé en revue les différentes techniques ayant un rapport avec nos travaux, afin de mieux cerner notre sujet d'étude. En particulier, nous nous sommes intéressés aux recherches antérieures concernant la modélisation déclarative, au traitement des informations contenues dans un texte menant à l'interprétation des connaissances, et au domaine de la génération par contraintes.

Après cette étude, nous avons débuté notre partie contribution au projet DRAMA par la recherche de nouvelles approches, permettant d'extraire du texte d'une pièce de théâtre les informations jugées utiles pour la mise en scène.

Pour mener à bien ces recherches, nous avons commencé par étudier les différentes particularités des textes théâtraux, notamment la structure et la forme des textes au format d'édition. A partir de cette étude, nous avons pu mettre en place un nouvel outil de balisage qui utilise le langage XML, qui permet de repérer les termes importants dans un texte de théâtre, afin de développer le premier module DRAMAtexte du projet DRAMA. Nous avons commencé par créer des nouvelles balises qui vont, non seulement permettre de repérer des informations pertinentes, mais aussi, de préciser les caractéristiques ou les propriétés de chacun des éléments identifiés à l'aide de données associées.

Le processus de balisage se déroule en deux étapes qui sont le balisage automatique et manuel. Nous avons classé ces balises en trois catégories distinctes :

- les balises automatiques qui sont mises en place à partir de la détection d'informations grâce à des règles typographiques ou grâce à la structure même du texte,
- les balises manuelles ajoutées par un lecteur,
- et les balises dédiées au metteur en scène pour lequel nous avons choisi qu'il serait le seul habilité à modifier le texte de l'auteur.

Ainsi, dans cette première phase, nous avons montré comment ces nouvelles balises, créées en suivant un formalisme XML, nous permettent de repérer et d'extraire les informations utiles pour le travail de mise en scène, mais également les propriétés indispensables pour la génération automatique des scènes virtuelles. Notons aussi que nous avons pris en considération l'aspect ergonomique de l'outil que nous avons développé, afin que des utilisateurs non spécialistes des modeleurs géométriques puisse l'utiliser sans un trop gros effort d'apprentissage, même si ce point est perfectible.

En plus des les balises prédéfinies initialement, chaque utilisateur a la possibilité de créer de nouvelles balises afin d'enrichir les possibilités offertes par l'outil. Ceci va permettre de disposer, au fur et à mesure de l'utilisation de DRAMAtexte, d'une diversité de balises et de textes balisés qui vont servir de base de données de textes théâtraux consultables et interrogeables par une communauté d'utilisateurs.

Ensuite, à partir des choix effectués pour DRAMAtexte, nous avons commencé l'étude de DRAMAscène, un autre module de DRAMA destiné à gérer les aspects graphiques dans DRAMA, notamment la partie génération des contraintes et visualisation de scènes virtuelles.

Pour se faire, nous avons récupéré les propriétés descriptives issues de DRAMAtexte afin de les interpréter et donc de les traduire en contraintes géométriques et spatiales. Ces contraintes varient selon le type de l'objet à placer et son rôle (objet à placer ou objet référence), car à chaque contrainte correspond une zone de placement et un repère de contrainte, permettant de déterminer l'emplacement et l'orientation de chaque objet à placer dans la scène. Ces informations concernant ces contraintes sont aussi stockées dans un fichier XML, afin de construire une base de connaissances sur la façon de définir les relations géométriques entre les différents types d'objets.

A l'issue de cette interprétation des propriétés descriptives en contraintes, nous disposons de la modélisation de notre problème sous la forme d'un ensemble de contraintes à résoudre. Il a donc été nécessaire d'étudier de de choisir des techniques nous permettant de résoudre un ensemble de contraintes géométriques, dans un contexte spécifique qui est la génération d'un espace scénique. Pour cela, nous avons appliqué des algorithmes basés sur le filtrage des CSP pour réduire l'espace de recherche, avant de lancer une recherche de solution effective. Les variables manipulées dans DRAMA, spécifiant la position et l'orientation de chaque objet, prennent leurs valeurs dans des domaines continus, et plus il y a de contraintes et d'objets à placer, plus la recherche de solution peut être longue. Ainsi, nous avons choisi une métaheuristique basée sur le recuit simulé, technique permettant de trouver une solution en un temps acceptable pour l'utilisateur.

Une fois cette solution trouvée, les valeurs des variables sont utilisées pour placer les objets dans la scène. En effet, ces valeurs sont transmises à un moteur physique appelé Bullet qui va à son tour gérer les éventuelles collisions entre objets. La partie visualisation est assurée par le moteur 3D Ogre.

Ces travaux ont été très enrichissants car ils m'ont permis de travailler sur la conception de DRAMA en collaboration avec Matthieu Pouget qui est metteur en scène. Cette collaboration a entrainé de nombreux échanges très intéressants et très fructueux, parfois passionnés...

Les travaux présentés dans ce mémoire vont pouvoir être réutilisés lors de nouvelles recherches, notamment des travaux consistant à travailler sur l'interprétation des propriétés à partir de données contenues dans une base de connaissances.
Nous avons développé un outil permettant d'extraire les propriétés concernant les entités : les objets, les accessoires, et le décor.
En ce qui concerne les personnages, nous nous sommes limités à extraite et traiter les propriétés concernant leur placement dans la scène. Mais il est également intéressant d'effectuer des recherches sur la façon de codifier et de représenter les acteurs virtuels, de façon à restituer leur état émotionnel, au travers de leur expression ou de leur posture. D'ailleurs, ce point est actuellement étudié dans le cadre d'un doctorat.
Une autre piste de recherche pourra concerner l'ajout d'animations et de déplacements des personnages dans la partie visualisation de scènes. Ceci afin de permettre la prise en compte d'aspects matériels lors des déplacements, mais aussi afin d'ajouter plus de réalisme sur l'étude des textes et en particulier des textes de théâtre.

De plus, on pourrait aller au delà de l'étude de textes pour générer des scènes à partir d'éléments autres que textuels (par exemple, la parole ou les graphiques), en s'appuyant sur les méthodes basées sur les langages naturels ou l'analyse d'images pour l'extraction de propriétés.
Il pourra être également intéressant d'étudier et développer un espace de travail collaboratif pour DRAMA afin que plusieurs utilisateurs puissent travailler ensemble sur un même texte, à distance, ce qui est particulièrement intéressant pour les premières étapes du travail de mise en scène où les personnes n'ont pas encore besoin de se rencontrer physiquement.

DRAMA a été un projet fédérateur pour des chercheurs dans des domaines très différents, issus du monde littéraire, théâtral, et informatique. Les échanges et les apports mutuels ont été importants, et enrichissants comme le sont généralement les travaux effectués dans un cadre multidisciplinaire.

Partie V- Annexes

V.1-Outils de développement

L'application DRAMA a été développée avec l'outil Eclipse IDE[35], libre, extensible, universel et polyvalent qui permet potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java, grâce à des bibliothèques spécifiques SWT[36]. Il est également utilisé aussi pour écrire des extensions à partir du plateforme Eclipse RCP[37].
Grâce à SWT, nous avons pu, dans DRAMAtexte, offrir des interfaces graphiques qui, que ce soit sur windows, Linux ou MacOs, ne sont pas différentiables de celles des autres applications. Par ailleurs, l'aspect natif de SWT offre aussi d'autres avantages :

- support des boîtes de dialogues standards de chaque système d'exploitation (ouverture d'un fichier, impression d'un document, etc.).
- possibilité d'utiliser des composants externes (navigateur, explorateur de fichier, etc.) en faisant appel aux fonctionnalités des applications les plus courantes de chaque système d'exploitation (Internet Explorer pour Windows, Mozilla/Firefox pour Linux ou Safari pour MacOS). Dans le cadre d'applications Eclipse RCP, ce composant peut s'avérer intéressant pour faire coopérer applications de types 'client riche' et 'client léger'.

En complément de Eclipse JDT[38], l'éditeur java sous Eclipse IDE, nous avons aussi utilisé Eclipse CDT(C/C++ developpement tools) pour développer DRAMAscène. En effet, DRAMAscène a été codé en C++ pour garder la rapidité et la fluidité de l'affichage de la scène 3D.

[35] Integrated Development Environment
[36] Standard Widget Toolkit
[37] Rich Client Plateform
[38] Java Developpement Tools

En résumé, nous avons fait appel aux librairies et outils de développement suivants pour la réalisation de l'ensemble du projet DRAMA :

- Jodconverter, bibliothèque java pour la conversion des documents textes en d'autres formats (HTML, XML, etc.),
- JDOM, pour manipuler les fichiers XML nécessaires dans le module DRAMAtexte, notamment pour la gestion des balises et interrogation des textes XML balisés.
- SWT et JFACE, pour la création des interfaces et objets graphiques (boutons, fenêtres, etc.) intégrés dans DRAMAtexte,
- Ogre, plateforme pour le développement des rendus 3D dans DRAMAscène,
- Bullet, pour l'utilisation des propriétés physiques des objets 3D (collision, gravité, masse, etc.),
- OgreBullet, pour l'intégration du moteur physique Bullet dans Ogre,
- CEGUI, bibliothèque pour l'utilisation d'objets visuels 3D (bouton, zone de liste, formulaire,etc.)
- TinyXML, nécessaire pour l'analyse de document XML et aussi pour modifier l'arborescence d'un fichier XML,
- RealPaver, solveur de contraintes utilisé pour réduire le domaine de recherche (filtrage) avant la phase de génération.

V.2-Outils mathématiques de base

Dans les moteurs 3D actuels, on utilise souvent de nombreux outils mathématiques pour améliorer la qualité du rendu et faciliter le placement et l'orientation d'un objet dans une scène. Parmi ces outils, nous présentons ici les matrices de rotation et les quaternions.

V.2.1) Matrice de rotation

La rotation d'un objet est donnée par les 3 matrices de rotation suivantes :

- $Ro_x(\Theta)$, rotation d'angle Θ autour de l'axe $\vec{X}$,
- $Ro_y(\Theta)$, rotation d'angle Θ autour de l'axe $\vec{Y}$,
- $Ro_z(\Theta)$, rotation d'angle Θ autour de l'axe $\vec{Z}$,

tels que,

$$Ro_x(\Theta)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & -\cos(\Theta) & \sin(\Theta) & 0\\ 0 & \sin(\Theta) & \cos(\Theta) & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$Ro_y(\Theta)=\begin{pmatrix}\cos(\Theta) & 0 & \sin(\Theta) & 0\\ 0 & 1 & 0 & 0\\ -\sin(\Theta) & 0 & \cos(\Theta) & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$Ro_z(\Theta)=\begin{pmatrix}\cos(\Theta) & -\sin(\Theta) & 0 & 0\\ \sin(\Theta) & \cos(\Theta) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

V.2.2) Les quaternions

Les quaternions sont très utiles pour calculer des rotations en 3D et pour permettre de représenter l'orientation d'un objet. Si l'on souhaite modifier l'orientation de cet objet, il suffit de construire le quaternion associé à la rotation souhaitée puis d'appliquer cette rotation à l'orientation initiale de l'objet.

Par définition, un quaternion est de la forme :

$$Q(a,b,c,d) = a + i.b + j.c + k.d \quad \text{où} \quad a,b,c \in \mathbb{R} \quad \text{et,}$$

$$i^2 = j^2 = k^2 = -1 \;,\; i*j = -j*i = k \;,\; j*k = -k*j = i \;,\; k*i = -i*k = j$$

Soient Q_0, Q_1 deux quaternions, tels que :

$$Q_0 = a_0 + i.b_0 + j.c_0 + k.d_0 \quad \text{et} \quad Q_1 = a_1 + i.b_1 + j.c_1 + k.d_1$$

nous avons,

$$Q_0 + Q_1 = (a_0 + a_1) + (b_0 + b_1)i + (c_0 + c_1)j + (d_0 + d_1)k$$

et,

$$Q_0 * Q_1 = (a_0.a_1 - b_0.b_1 - c_0.c_1 - d_0.d_1) + (a_0.b_1 + b_0.a_1 + c_0.d_1 - d_0.c_1).i$$
$$+ (a_0.c_1 - b_0.d_1 + c_0.a_1 + d_0.b_1).j + (a_0.d_1 - b_0.c_1 + c_0.b_1 + d_0.a_1).k$$

La norme d'un quaternion Q(a, b, c, d) vaut :

$$\|Q\| = \sqrt{a^2 + b^2 + c^2 + d^2}$$

Si la norme d'un quaternion vaut 1, on dit que le quaternion est normalisé. C'est ce genre de quaternion qui va nous intéresser en 3D. En effet, tout quaternion normalisé peut être associé à une rotation vectorielle 3D. Cette rotation vectorielle est une rotation autour d'une droite passant par l'origine du repère. Pour cela, il faut trois valeurs pour définir la direction de l'axe de rotation et une autre pour spécifier l'angle de rotation.

Prenons une rotation autour d'un axe orienté selon le vecteur $\vec{v}$, vecteur normalisé, de coordonnées (x, y, z) et d'angle α, le quaternion associé vaut :

$$Q = \cos\frac{(\alpha)}{2} + i.x.\sin\left(\frac{\alpha}{2}\right) + j.y.sin\left(\frac{\alpha}{2}\right) + k.z.sin\left(\frac{\alpha}{2}\right)$$

Par ailleurs, modifier l'orientation d'un objet revient à faire pivoter l'objet autour d'un ou plusieurs axes. Cependant, la composée de 2 rotations passant par le même point (ou l'origine) est une rotation passant par ce point. Donc, quelle que soit l'orientation d'un objet dans l'espace, on peut définir cette orientation par une rotation.

De plus, les quaternions sont plus rapides à multiplier que les matrices de rotation. Le gain n'est pas négligeable, si en plus on considère que pour mémoriser un quaternion il faut seulement 4 valeurs contre les 16 de la matrice, les quaternions prennent donc beaucoup moins de place en mémoire.

Partie VI- Bibliographie

[1] Andriamarozakaniaina T., Pouget M., Zaragoza J., and Gaildrat V. Dramatexte., Indexation et base de connaissances. Premier Colloque international sur la notation informatique de personnage, may 16-17, 2008 , Toulouse, France.

[2] Alavi M., Leidner D., *Knowledge management and knowledge management systems: Conceptual foundations and research issues*. In MIS Quarterly, p.107-136 , June 2001.

[3] Cadoz ., Luciani A., Florens J., *CORDIS-ANIMA : a modeling and simulation system for sound and image synthesis - the general formalism*. ACM Interactive 3D Graphics Conference, p.189-196 , 1993.

[4] Bonnefoy P-F., *Techniques de satisfaction de contraintes pour la modélisation déclarative. Application à la génération concurrente de scènes*. Thèse de doctorat, Université de Limoges, Juin 1999.

[5] Breay P., *The social ontology of virtual environments - criticisms and reconstructions*. The American Journal of Economics and Sociology, 269-282 , January, 2003.

[6] Champciaux L., *Introduction de techniques d'apprentissage en modélisation déclarative*. Thèse de doctorat, Université de Nantes, 1999.

[7] Charman P., *Gestion des contraintes géométriques pour l'aide à l'aménagement spatial*. Thèse de doctorat. École nationale des ponts et chaussées, Novembre 1995.

[8] Colin C., Desmontils E., Martin J., Mounier J.. *Modèle utilisateur d'un modéleur déclaratif, on* on Computational Graphics and Visualisation Techniques *(*Compugraphics'97) , 1997.

[9] Coyne B., Sproat R., *WordsEyes: an automatic text-to-scene conversion system*. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, p.487-496, SIGGRAPH '01, 2001.

[10] Daniel L. Rubin, Natalya F. Noy, MarK A. Musen. *Protege: A Tool for Managing and Using Terminology in Radiology Applications*. Journal of Digital Imaging, J Digit Imaging , 2007.

[11] Davies N., Mehdi Q., Gough N., *Virtual crime scene reconstruction with integrated animated characters*. Proceedings 18th european simulation multiconference, 2004.

[12] Desmontils E., Pacholczyk D., *Interprétation du vague dans la description linguistique d'une scène en modélisation déclarative*, avril 1997.

[13] Dorme G., *Etude et réalisation de techniques de prise de connaissance de scènes tridimensionnelles*. Thèse de doctorat, Université de Limoges, 2001.

[14] Dragicevic P., Fekete J-D., *Support for Input Adaptability in the ICon Toolkit*. Proceedings of the 6th international conference on multimodal interfaces, ICMI '04 , October 2004.

[15] Dragonas J., *Modélisation déclarative collaborative. Systèmes collaboratifs pour la modélisation déclarative en synthèse d'image*. Thèse de doctorat, Université de Limoges, France, 2006.

[16] Dupuy S., Egges A., Legendre V., Nugues P., *Generating a 3D simulation of a car accident from a written description in natural language: The Carsim system*. In proceedings of the workshop on temporal and spatial information processing - volume 13, TASIP '01, Stroudsburg, PA, USA , 2001.

[17] Essert C., *Sélection dans l'espace des solutions engendrées par un plan de construction géométrique,* thèse de doctorat, Université Louis Pasteur - Strasbourg I Strasbourg, 2001.

[18] Faugeras O., Papadopoulo T., *Grassmann-Cayley algebra for modeling systems of cameras and the algebraic equations of the manifold of trifocal tensors,* Technical Report - 1NR1A 3225 , 1997.

[19] Fellbaum C., *WordNet: an electronic lexical database*. Proceedings of 11th eurographics workshop on rendering, MIT Press, Cambridge, MA, USA , 1999.

[20] Gaildrat V.. *Modélisation déclarative d'environnements virtuels : Création de scènes et de formes complexes par l'énoncé de propriétés et l'emploi d'interactions gestuelles*. Habilitation à diriger des recherches, Université Paul Sabatier, Toulouse, 2003.

[21] Gaildrat V., *Declarative modelling of virtual environments, overview of issues and applications*. 3Ia'07, Athènes, Grèce, Mai 2007.

[22] Gaildrat V., Pouget M., Andriamarozakaniaina T., Zaragoza J ., Velonoromanalintantely R., Etape d'indexation d'un texte théâtral dans le cadre du projet DRAMA, dans : La notation informatique du personnage théâtral, Carnières-Morlanwelz, Lansman éditeur, 2010, p. 51-70.

[23] Gino B. *Collision detection in interactive 3D computer animation*. Eindhovern: Eindhovern University of Tehnology, 1999.

[24] Giudicelli V., Lefranc M., *Ontology for immunogenetics: The IMGT-ONTOLOGY*. Bioinformatics, p.1047-1054, Laboratoire d'ImmunoGénétique Moléculaire, UPR CNRS 1142, Institut de Génétique Humaine, Montpellier, France , 1999.

[25] Glass K., Bangay S., Alcock B., *Mechanisms for multimodality : taking fiction to another dimension*. Proceedings of the 5th international conference on computer graphics, virtual reality, visualisation and interaction in Africa, AFRIGRAPH '07, Grahamstown, South Africa, 2007.

[26] Golopentia S., Martinez-Thomas M., Pouget M., Les étiquettes du personnage de théâtre, dans : La notation informatique du personnage théâtral, Carnières-Morlanwelz, Lansman éditeur, 2010, p. 83-94.

[27] Grimm S, Hitzler P., Abecker A., *Knowledge representation and ontologies. Logic, ontologies and semantic web languages*.Studer et al. Semantic Web Services: Concepts, Technology and Applications, p.51-106, Institute AIFB, University of Karlsruhe, 2007.

[28] Grottel S., Reina G., Zauner T., Hilfer R., Ertl T.. *Particle-based rendering for porous media*. Proceedings of the annual SIGRAD conference, p.45-51 , 2010.

[29] Gutierrez-Garcia J-O., Koning J-L., Corchado F., *An obligation approach for exception handling in interaction protocols*.Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03, WI-IAT '09, Washington, DC, USA , 2009.

[30] Hamilton-Smith, N., *A Versatile Concordance Program for a Textual Archive.* In R. A. Wisbey (ed.), The Computer in Literary and Linguistic Research (pp. 235–44). Cambridge University Press. 1971.

[31] Hegron G., *Conception architecturale et modélisation déclarative*. *Thèse* de *doctorat* Sciences pour l'Ingénieur, option Architecture. CERMA UMR CNRS 1563, École d'Architecture de Nantes , 2005.

[32] Hockey, S. and I. Marriott. *The Oxford Concordance Project (OCP) – Part 1*. ALLC Bulletin 7. 1979.

[33] Hockey, S. and I. Marriott. *The Oxford Concordance Project (OCP) – Part 4*. ALLC Bulletin 8.

1980.

[34] Huot S., *Reconstruction de bâtiment 3D à partir de photographies*. École des mines de Nantes Département informatique, Nantes, 2000.

[35] Huot S., Dumas C., Dragicevic P., Fekete J-D., Hégron G.. *The MaggLite Post-WIMP Toolkit: Draw It, Connect It and Run It*. Proceedings of the 17th annual ACM symposium on User interface software and technology, p.257-266,UIST '04,Santa Fe, NM, USA , October 2004.

[36] Knuth D. E., *Tau Epsilon Chi, a system for technical text*. American Mathematical Society, 1979. ISBN: 0-8218-0209-7.

[37] Kwaiter G., *Modélisation déclarative de scènes : étude et réalisation de solveurs de contraintes*. Thèse de doctorat, Université Paul Sabatier, Toulouse, 1998.

[38] Larive M., Le Roux O., Gaildrat V. *Using Meta-Heuristics for Constraint-Based 3D Objects Layout.* International Conference on Computer Graphics and Artificial Intelligence (3IA 2004), may 2004.

[39] Le Roux O., Gaildrat V., Caubet R., *Constraint-based 3D isothetic object layout for declarative scene modeling*. International conference on imaging science, systems, and technology (CISST 2003), Las Vegas , 2003.

[40] Le Roux O., *Modélisation déclarative d'environnements virtuels : contribution à l'étude des techniques de génération par contraintes*. Thèse de doctorat, Université Paul Sabatier Toulouse, 2003.

[41] Le Roux O., Gaildrat V., Caubet R.. *Using constraint satisfaction techniques in declarative modelling*.Geometric Modeling: Techniques, Applications, Systems and Tools. Muhammad Sarfraz (Eds.), p. 1-20 , 2004.

[42] Lucas M., Desmontils E.. *Les modeleurs déclaratifs*.Revue internationale de CFAO et d'informatique graphique, Vol. 10, N°6, p.559-585 , décembre 1995.

[43] Hervé Luga. *Vie artificielle et synthèse d'images : étude des mécanismes évolutionnistes pour la synthèse de formes et de comportements*. Thèse de doctorat, Université Paul Sabatier, juillet 1997.

[44] Martinez-Thomas M., *Premières pistes pour la spectalecture: DRAMA à l'épreuve de la dramaturgie textuelle*, in Le théâtre de l'Occident à l'Orient, CHER, Strasbourg, 2010.

[45] Pantelia, M., Peevers, R., *Thesaurus Linguae Graecae*, University of California, Irvine, California, 2004.

[46] Parish, Y. I.H., Muller, P., “Procedural modeling of cities”, SIGGRAPH'01, Los Angeles, pp. 301-308, 2001.

[47] Plemenos D., *Contribution à l'étude et au développement des techniques de modélisation, génération et visualisation de scènes, - Le projet Multiformes,* thèse de doctorat, Université de Nantes, 1991.

[48] Plemenos D., Ruchaud W., Tamine K., *Interactive techniques for declarative modelling*.

International conference 3IA'98, Limoges (France) , 1998.

[49] Pouget M.. *Drama texte : pour une analyse informatisée de la dramaturgie textuelle*. Université Toulouse le Mirail , 2005.

[50] Pouget M. *Projet Cahier de Scène Numérique : premier pas vers le cahier de scène numérique.*

Thèse de doctorat en Arts du Spectacle, Université Toulouse II, 2010.

[51] Pouget M., Rabiafaranjato V., Gaildrat V., Sanza C., Martinez-Thomas M., *Codifying emotions in a theatrical context with symbolic rendering (short paper),* International Conference on Computer Graphics and Artificial Intelligence (3IA), Athènes, mai 2011.

[52] Sanchez S., Le Roux O., Luga H., Gaildrat V. *Constraint-Based 3D-Object Layout using a Genetic Algorithm*. International Conference on Computer Graphics and Artificial Intelligence (3IA), 2003.

[53] Sellinger D., *La modélisation géométrique déclarative, interactive : le couplage d'un modeleur déclaratif et d'un modeleur géométrique classique,* Thèse de doctorat, Université Paul Sabatier, Toulouse, 2003.

[54] Smelik R., Tutenel T., Kraker K., Bidarra R.. *Declarative terrain modeling for military training games*. International journal of computer games technology volume 2010, 2:1--2:11, 2010.

[55] Sosnov A., Macé P., Hégron G., *Semi-metric formal 3D reconstruction from perspective sketches*. Proceeding ICCS '02 proceedings of the international conference on computational science-part II , 285--294,ICCS '02, London UK , Avril 2002.

[56] Sosnov A., Huot S., Macé P., Hégron G., *Rapid incremental Architectural modeling from imprecise perspective sketches and geometric constraints*.Actes de la 12ème international conference on computer graphics graphicon'2002, Novgorod , 2002.

[57] Stolte N. *Espaces discrets de haute résolutions : une nouvelle approche pour la modélisation et la synthèse d'images réalistes*. PhD thesis, Paul Sabatier University, Toulouse, France, 1996.

[58] Tching L., T*raitement générique des interactions haptiques pour l'assemblage d'objets issus de CAO*. Thèse de doctorat, INSA de Rennes, février 2010.

[59] Thomas R., Gruber. *A translation approach to portable ontology specifications, knowledge acquisition*.Special issue: current issues in knowledge modeling , 199-220,Volume 5 Issue 2, London, UK, June , 1993.

[60] Uschold M., Grüninger M., *Ontologies:Principles, methods and applications*. Knowledge Engineering Review Volume 11 Number 2, p.93-136 , June 1996.

[61] Vandeloise C., *L'espace en français*. Seuil, Paris , 1986.

[62] Vassilas N., Miaoulis G., Chronopoulos D., Konstantinidis E., I. Ravani, D. Makris, D. Plemenos, *MultiCAD-GA : A System for the Design of 3D Forms based on Genetic Algorithms and Human Evaluation*, Lecture Notes in Artificial Intelligence, Springer. Vol. 2308, pp. 203-214, 2002.

[63] Weyland D., *Simulated annealing, its parameter settings and the longest common subsequence problem*. Proceedings of the 10th annual conference on genetic and evolutionary computation, p 803-810, GECCO '08, Atlanta, GA, USA , 2008.

[64] Xu K., *Automatic object layout using 2D constraints an semantics*. Computing and information technologies exploring emerging technologies proceedings of the international conference, 2001.

[65] Winter A., Strübing A., Ißler L., Brigl B., Haux R.. *Ontology-based assessment of functional redundancy in health information systems*. Models in Software Engineering, p.213-226, Springer-Verlag, Berlin, Heidelberg , 2009.