Présentations multimédia synchronisées pour le WWW

Franck Rousseau

Institut National Polytechnique de Grenoble

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Informatique : Systèmes et Communications

présentée et soutenue publiquement

par

Franck ROUSSEAU

le 21 janvier 1999

Présentations Multimédia Synchronisées pour le WWW

Directeur de thèse : Andrzej DUDA

Jury :	M. Jean-Pierre Verjus,	président
	M. Guy Bernard,	rapporteur
	M. Bernard Merialdo,	rapporteur
	M. Vania Joloboff,	examinateur
	M. Andrzej Duda,	directeur de thèse

À mes parents
et Virginie.

Je tiens particulièrement à remercier Andrzej Duda et Vania Joloboff qui m'ont encadré pendant ces dernières années et m'ont permis de mener à bien cette thèse, ainsi que Paul Jacquet de m'avoir accueilli au laboratoire LSR et Jacques Febvre de m'avoir accueilli à l'OSF-RI, aujoud'hui devenu The Open Group Research Institute.

Je remercie Guy Bernard et Bernard Merialdo d'avoir accepté d'être les rapporteurs de ce travail et Jean-Pierre Verjus le président du jury.

Je remercie Bull S.A. pour le soutien financier, et mes différents interlocuteurs au sein de cette entreprise, notamment Najah Naffah et Michel Habert.

Finalement je tiens à remercier tous ceux qui ont contribué au bon déroulement de ces années de thèse, sur le plan privé, ma famille et mes amis, ainsi que sur le plan professionnel, toute l'équipe de l'Open Group Research Institute de Grenoble.

Table des matières

Liste des figures

Chapitre 1

Introduction

In *Le Dictionnaire Francophone En Ligne*[1], Hachette :

> **multimédia** 1. n. m. Technique permettant de rassembler sur un même support des moyens audiovisuels (textes, sons, images fixes et animées) et des moyens informatiques (programmes, données) pour les diffuser simultanément et de manière interactive ; équipement, industrie se rapportant à cette technique. 2. adj. Qui concerne, qui utilise plusieurs médias. *Un groupe de communication multimédia.* (V. mass media.) 3. adj. Qui concerne, qui utilise le multimédia. *Un dictionnaire multimédia.* (V. encycl. information.)

Ces dernières années le terme *multimédia* est devenu incontournable. Paradoxalement, il est assez difficile de donner une définition exacte et précise recouvrant toutes ses acceptions. L'exemple ci-dessus en définit quelques unes. Ce terme est utilisé dans de nombreux contextes puis est devenu en quelque sorte un élément *marketing* au fil du temps : plus une seule publicité pour un produit informatique n'omet d'employer *multimédia* dans son énoncé ; mais que doit-on entendre par *ordinateur multimédia* ? Si l'on se réfère à la définition première de multimédia[11, 69, 104], à savoir la capacité à manipuler simultanément plusieurs types de média — texte, son, image, vidéo, etc. — il y a longtemps que les ordinateurs sont multimédia.

[1] `http://www.francophonie.hachette-livre.fr/`

Il faut donc, avant d'aller plus loin, définir clairement quels sont les domaines du multimédia qui nous intéressent et quelle est la problématique associée.

Le multimédia est un domaine de recherche très jeune dans le monde de l'informatique. Son développement avait été entrevu par divers visionnaires de la recherche, dès l'après-guerre par Vannevar Bush[15] dans son célèbre article *As We May Think* datant de juillet 1945, dans lequel il imagine un appareil pouvant regrouper les informations et les communications emmagasinées par son utilisateur et capable de les restituer de façon simple et rapide.

La recherche multimédia est née et a bénéficié des progrès accomplis dans des domaines de plus bas niveau comme le matériel et les systèmes d'exploitation, les techniques de traitement du signal et de codage, la modélisation des données complexes, etc. Le but ultime du multimédia est de fournir un moyen d'interaction avec des données en tout genre qui soit simple et rapide comme Bush l'avait imaginé en faisant abstraction de la technique informatique nécessaire à l'obtention du résultat. Nous sommes encore loin de cet idéal, bien que les interfaces graphiques aient fait des progrès et que l'hypertexte rende la navigation dans les bases documentaires plus aisée, mais la prise en compte de la parole ou du regard dans les interactions avec la machine devrait nous faire franchir un nouveau pas d'ici quelques années.

1.1 Domaines d'application

On peut distinguer deux grandes catégories d'application du multimédia. D'un côté, le domaine de la téléconférence multipartie dont le but est de fournir des outils de communication audio et vidéo permettant à plusieurs personnes en des lieux distincts de participer, en temps réel, à une réunion virtuelle. De l'autre côté, on peut identifier le domaine des présentations multimédia. Il s'agit ici de pouvoir définir, stocker et restituer des présentations formées d'un ensemble d'éléments pouvant être des médias de base comme le son ou la vidéo numérique, ou eux-mêmes d'autres présentations.

Nous emploierons dans ce mémoire l'expression *présentation multimédia*, car c'est le domaine d'études concerné ici. Toutefois la distinction faite ci-dessus n'empêche pas une certaine imbrication des deux domaines ; d'une part, parce que de nombreux problèmes leur sont communs et d'autre part,

parce que l'on peut vouloir faire une présentation multimédia lors d'une vidéoconférence ou inversement inclure un flot vidéo en temps réel dans une présentation, par exemple dans le cadre de l'enseignement à distance.

1.2 Problématique

Les récentes avancées techniques aussi bien dans les domaines du logiciel que du matériel ont provoqué l'explosion du multimédia dans l'informatique personnelle. Dans ce sens multimédia se réfère communément à la capacité de l'ordinateur, de ses périphériques — carte son, carte vidéo 3D, carte de compression, écran de grandes dimensions, enceintes acoustiques — et de ses logiciels à manipuler et restituer des données autre que le texte et l'image, principalement le son et la vidéo numériques ainsi que les environnements virtuels en trois dimensions — dans les jeux par exemple.

Un nouvel acteur est venu s'ajouter à cela, le réseau mondial Internet, via le *World Wide Web*. Par sa simplicité d'utilisation, le Web est devenu, aux côtés du CD-ROM, un médium de choix dans la diffusion du multimédia, rendant l'accès à une quantité croissante de données aussi simple qu'un *clic* de souris.

Quoique simpliste, cette vision nous permet toutefois d'extraire la problématique liée au multimédia et de présenter succinctement les domaines de recherches qui y sont associés.

1.2.1 Les données

Le multimédia implique la création, la manipulation et la restitution de médias de base sous forme numérique. Les types de médias utilisés dépendent en premier lieu des possibilités d'interaction de l'homme avec sa machine. L'utilisation du toucher, de l'odorat et du goût dans ce domaine étant très marginale, bien qu'étudiée au sein de laboratoires comme le MediaLab au MIT et le CLIPS à Grenoble par exemple, il nous reste l'ouïe et la vue. La première étape du multimédia est donc la création, la manipulation et la restitution de données *sonores* et *graphiques* sous les formes les plus diverses.

On peut également distinguer ces données suivant leur mode de création.

Elles peuvent être produites soit par *numérisation* d'un signal analogique soit directement par *synthèse* à partir de modèles mathématiques et physiques. Ces deux domaines ont donné lieu à des recherches portant sur la théorie du signal et l'échantillonnage des données ainsi que sur la perception psychosensorielle de celles-ci par l'homme lors de leur restitution.

La taille importante des données numérisées a aussi amené à l'étude du codage et permis de développer des techniques basées sur des propriétés psychosensorielles, comme le phénomène de masquage des fréquences en audio par exemple[85], ou intrinsèques au média, comme la compensation de mouvement en vidéo qui prend en compte le fait que des images successives d'une séquence contiennent énormément d'information redondante.

1.2.2 La représentation

Dès lors que l'on dispose des médias de base, se pose le problème de les associer pour créer des *présentations multimédia*. Une présentation est la restitution organisée des différents médias de base suivant un scénario défini par un auteur.

On peut distinguer deux approches. Tout d'abord, la plus simple qui consiste à créer un nouveau média brut à partir des données de base et revient à enregistrer la présentation sous la forme de données brutes, vidéo par exemple. On peut comparer cela à la réalisation de trucages en images de synthèse dans un film, que l'on superpose à la prise de vue originale pour créer une version définitive. La conséquence de cela est que l'on n'a plus accès aux données originales lorsque l'on ne dispose que de la version finale. Cette méthode est donc peu adaptée au domaine numérique dans lequel la réutilisation est un but important — c'est toutefois un bon moyen de protéger ses droits alors que ce problème reste ouvert pour ce qui est des médias numériques. Cependant, le principal avantage est la simplicité du résultat, constitué par un flot de données unique qu'il suffit de jouer, mais on se situe ici à la limite du domaine du multimédia puisqu'il ne reste en réalité qu'un seul média, et l'on rejoint plutôt la problématique du codage et de la restitution des données vue en 1.2.1.

La seconde approche est réellement une approche de création multimédia dans laquelle on considère la définition d'une présentation à partir de médias de base, qui peuvent être une présentation complexe déjà existante. Cette méthode soulève alors le problème majeur du multimédia, la gestion de

la dimension temporelle. Il va falloir spécifier, par un moyen quelconque, quand et comment les différents éléments d'une présentation devront être présentés. Les études entreprises dans ce domaine ont permis de définir plusieurs *modèles temporels* permettant la spécification des relations de composition et de synchronisation entre médias temporels à un niveau d'abstraction plus ou moins élevé.

L'utilisation de ces modèles pour la spécification de présentations multimédia peut se faire de deux manières. La première consiste à intégrer les différents modules de restitution des médias sous la forme d'une application à l'aide d'un langage de programmation. La seconde méthode s'attache à décrire la structure de la présentation multimédia, sous une forme programmatique ou déclarative. Cette deuxième approche facilite grandement le processus d'édition d'une présentation, puisqu'il se borne à la description et n'inclut pas les mécanismes nécessaires à la réalisation de la présentation.

1.2.3 La restitution

Afin de restituer la présentation telle que son auteur l'a souhaitée, des mécanismes de contrôle du bon déroulement temporel de celle-ci sont nécessaires. On divise communément la gestion du temps en trois tâches[69] : la *synchronisation intra-média*, la *synchronisation inter-média* et l'*ordonnancement* ou *scheduling*. Cela concerne respectivement la gestion du temps pour les échantillons d'un média, le contrôle de la simultanéité de présentation entre plusieurs médias devant être présentés en même temps comme une vidéo et sa bande-son, et le démarrage ou l'arrêt des médias au moment approprié pour qu'ils soient présentés aux instants convenus.

Ce domaine de la recherche multimédia est également très prolifique. En effet la plupart des systèmes d'exploitation utilisés fréquemment ne sont pas temps réel, ce qui signifie qu'ils ne peuvent garantir aucune propriété quant à leur comportement temporel. Il faut donc développer des mécanismes et des algorithmes appropriés permettant de contourner cet inconvénient de taille pour la présentation de données temporelles et rétablir un comportement correct lorsqu'un problème survient.

1.2.4 La distribution

Un nouvel acteur est venu se greffer aux systèmes multimédia dont nous avons décrit trois aspects dans les sections précédentes, il s'agit du réseau de communication. L'interconnexion des machines par un réseau local ou le réseau mondial Internet permet désormais d'échanger des données numériques aisément, notamment en les disposant sur des serveurs les rendant disponibles depuis des machines distantes. Naturellement les systèmes multimédia doivent pouvoir tirer parti de cette possibilité par la création de présentations distribuées dont tous les éléments ne se situent plus en un lieu unique, mais sont répartis à travers le réseau.

Le caractère distribué de telles présentations soulève de nouvelles questions concernant le transport des données multimédia qui sont sensibles au temps et dans la plupart des cas très volumineuses. Il s'est développé un nouveau domaine d'études sur les *communications multimédia* dont la préoccupation est l'acheminement dans les meilleures conditions des données du serveur au client.

De nombreuses solutions ont été étudiées pour fournir une qualité de service (QoS) contrôlable et garantie dans des réseaux comme ATM ou au contraire adapter au mieux le service au caractère *best effort* de l'Internet. De nouveaux protocoles offrent déjà des possibilités intéressantes dans le cadre des présentations multimédia bien qu'ils aient été plus particulièrement destinés aux communications temps réel dans le cadre de la téléconférence multipartie.

1.3 Cadre et objectif de cette thèse

Le travail de cette thèse s'est effectué dans le cadre particulier suivant : il a été réalisé au sein de l'équipe Drakkar du laboratoire LSR-IMAG, effectué dans les locaux et avec le support de l'Open Group Research Institute de Grenoble et sponsorisé par Bull S.A.

L'objectif de ce travail a été d'étudier la possibilité de diffuser et d'accéder à des documents multimédia structurés complexes sur le *World Wide Web* d'une manière aussi transparente qu'on le fait aujourd'hui pour les documents HTML, *HyperText Markup Language*, classiques. Jusqu'à l'apparition de SMIL, *Synchronized Multimedia Integration Language*, il y a quelques

mois, les modèles existants étaient très tournés vers leur environnement d'édition et ne fournissaient pas un format de document standard et répandu. Nous nous sommes donc orientés vers l'extension du langage HTML pour permettre l'intégration de données temporelles dans l'environnement Web le plus simplement possible. Parallèlement à notre étude, le *World Wide Web Consortium* (W3C) a lancé une activité *synchronized multimedia* qui a abouti à la publication d'une recommandation pour un nouveau langage, SMIL, proposant des fonctionnalités similaires mais à l'extérieur de HTML et en utilisant un modèle temporel différent. Nous reviendrons plus longuement sur ce langage dans les sections suivantes.

1.4 Organisation du manuscrit

Cette introduction a pour but de présenter sommairement la recherche multimédia ainsi que la position où nous nous situons par rapport à celle-ci.

Le chapitre 2 présente les aspects généraux du multimédia en abordant les systèmes et applications, la problématique liée à l'utilisation de données sensibles au temps, puis l'impact sur les communications.

Le chapitre 3 aborde plus en détail notre principal intérêt, les présentations multimédia. Nous y présentons les différents modèles de représentation du temps et de spécification de la synchronisation. Des systèmes existants ainsi que les principaux standards sont décrits.

Après cette présentation du domaine dans lequel notre travail s'inscrit, le chapitre 4 expose notre contribution. Nous exprimons notre objectif en détail et proposons une solution consistant à étendre le langage HTML au domaine temporel. Ces extensions permettent de spécifier des présentations multimédia pour le World Wide Web.

Ensuite, nous décrivons, dans le chapitre 5, les réalisations effectuées afin de rendre opérationnelles les extensions proposées. Une architecture d'exécution a été conçue et développée en Java. Nous abordons également le contrôle de la qualité de présentation et des communications.

Finalement nous concluons sur ce travail dans le chapitre 6. Un bilan du travail effectué est présenté et les perspectives d'études futures sont exposées.

Chapitre 2

Le multimédia

2.1 Applications et systèmes multimédia

Nous n'allons pas faire ici un recensement des systèmes et applications multimédia depuis l'apparition du domaine, chose qui a été faite dans de nombreux ouvrages de référence dont certains sont cités[11, 69, 104] et cela nous éloignerait quelque peu du sujet de cette thèse. Intéressons-nous plutôt à quelques applications typiquement qualifiées de multimédia pour ensuite aborder les grands problèmes qui sont ainsi soulevés.

2.1.1 Quelques applications majeures

Les CD-ROM

Commençons par une petite provocation ! Un CD-ROM n'a rien d'une application multimédia puisqu'il s'agit évidemment d'un support optique destiné au stockage des données numériques. La plupart des logiciels ou des données numériques sont aujourd'hui distribués sous la forme de CD-ROM. C'est un support de grande capacité et d'un coût de revient très faible bien adapté à cette utilisation.

Cependant ce terme est passé dans le langage courant pour désigner le contenu que l'on trouve principalement sur les CD-ROM, c'est-à-dire les applications du type encyclopédie multimédia ou musée virtuel. Quasiment

toutes les grandes publications généralistes proposent aujourd'hui une rubrique indifféremment appelée *multimédia* ou *CD-ROM*. Quelques CD audio sont aussi distribués avec une piste multimédia supplémentaire accessible si l'on possède un ordinateur ou directement avec un CD-ROM rajouté dans le boîtier sur lesquels on trouvera, généralement au format QuickTime, des extraits de clip vidéo ou de concert pour les artistes les moins créatifs, mais pouvant aller jusqu'à une vraie présentation multimédia créée spécialement pour l'occasion.

On peut extraire deux grandes caractéristiques de ce genre d'application :

- elles utilisent de façon conséquente l'hypertexte pour naviguer à travers la masse importante d'information qu'elles représentent, et cela n'a pas toujours un effet positif, l'utilisateur pouvant rapidement se perdre dans un véritable labyrinthe ;
- rares sont les applications intégrant réellement les différents médias, elles sont le plus souvent calquées sur un modèle traditionnel de type encyclopédie ou livre, auquel on a ajouté du son et de la vidéo ainsi que des liens hypertextes.

Les jeux

Les jeux vidéo pour ordinateur constituent à eux seuls une catégorie particulière, bien qu'ils soient eux aussi distribués sur CD-ROM, et cela tient sûrement au fait qu'ils existent depuis les débuts de l'informatique personnelle et qu'ils ont été le premier type d'application à intégrer réellement des données multimédia. Cela a commencé avec des animations et du son, de qualité souvent élevée pour créer des bruitages les plus réalistes possible, pour au fur et à mesure ajouter des séquences vidéo, des images de synthèse, puis récemment des environnements en trois dimensions des plus réalistes.

Il est intéressant de noter que les jeux sont sûrement les applications courantes qui demandent les plus importantes ressources à un ordinateur personnel. Ils nécessitent une carte son de bonne qualité, un lecteur CD-ROM suffisamment rapide pour transférer les énormes quantités de données utilisées et finalement un affichage graphique rapide et de bonne résolution, une carte 3D étant même souvent conseillée pour soulager le processeur des lourdes tâches comme le calcul de texture.

Il faut aussi noter un autre aspect très intéressant, la communication. Les premiers jeux offraient la possibilité de jouer à plusieurs personnes, généralement deux, sur une même machine. Puis, on a pu connecter plusieurs machines ensemble, simplement à l'aide d'un câble ou sur un véritable réseau local, toujours pour jouer à quelques personnes, chacune devant son ordinateur. Aujourd'hui, on peut se confronter à d'autres joueurs en se connectant à des serveurs dédiés sur Internet. Toutefois, le problème est simplifié par le fait que chacun des joueurs possède le même logiciel et qu'un jeu est virtuel. Il n'y a donc pas besoin d'échanger de données volumineuses, un codage approprié permettant de reconstituer les scènes virtuelles sur chacun des sites. La difficulté principale vient de la latence dans les communications qui peut éventuellement créer des incohérences.

La vidéoconférence

Abordons maintenant la vidéoconférence. Ce terme désigne plus précisément la téléconférence multipartie, c'est-à-dire la mise en présence de plusieurs personnes situées dans des endroits distincts et éloignés par transmission de la parole et de l'image. Parfois, cela s'accompagne d'un "tableau blanc", espace partagé sur lequel chacun peut écrire. Certaines applications vont plus loin en recréant l'espace virtuel dans lequel les personnes se rencontrent, simulant ainsi une salle de réunion en trois dimensions où chacun a sa place à l'aide du son spatialisé[68].

On retrouve, dans la vidéoconférence, les ingrédients déjà cités précédemment, son et vidéo numériques ainsi que communication, mais il s'agit cependant d'un cas vraiment singulier dans l'univers du multimédia. D'abord, c'est une application dont l'origine est ancienne et qui visait au début à agrémenter le téléphone analogique de la vidéo, également analogique. Depuis, elle est passée dans le domaine numérique et les communications ne sont plus uniquement point-à-point, entre deux interlocuteurs, mais multipartie. Ensuite, parce qu'elle fait collaborer deux acteurs qui ont toujours eu des approches différentes, voire antagonistes, les télécommunications et l'informatique. Finalement, parce que la transmission de son et de vidéo numérique en temps réel est une vraie difficulté dès que le débit des réseaux est limité, comme sur les lignes téléphoniques ou Internet, et que leur fiabilité diminue, Internet surchargé. Ce domaine a donc soulevé beaucoup de problèmes auxquels la recherche s'est intéressée :

- le *multicast*, permettant la diffusion des données d'un interlocuteur vers tous les autres d'une façon la plus efficace possible, en évitant au possible d'avoir à envoyer les données autant de fois qu'il y a de personnes pour économiser la bande passante ;
- la communication temps réel, qui a requis le développement de protocoles spécifiques suffisamment légers pour le transport rapide des données sur les réseaux IP ;
- la compression vidéo, pour diminuer fortement le volume de données transportées, avec le développement de formats comme H.261[49, 107], appelé aussi p×64[65], spécifiquement conçu pour être utilisé sur ISDN et B-ISDN[1] qui offrent des débits multiples de 64 kb/s ;
- les techniques de contrôle et de correction d'erreur, permettant de diminuer l'impact de la perte de données ou de l'altération de celles-ci pendant le transport ;
- la gestion de la qualité de service (QoS), permettant de garantir une qualité de communication plus ou moins stable lors d'une communication à travers le développement des réseaux ATM[2] ou de protocoles comme RSVP[3][8].

La recommandation H.323[50] publiée par l'ITU[4] spécifie les systèmes de vidéoconférence sur réseaux à commutation de paquets en s'appuyant sur d'autres de ses recommandations concernant le codage du son et de la vidéo, les terminaux multimédia ainsi que l'établissement des communications.

Télé-enseignement, télé-médecine

Partant de ce concept de vidéoconférence, on peut imaginer toute une séries de déclinaisons en applications diverses. Les deux plus intéressantes par les problèmes qu'elles soulèvent sont le télé-enseignement et la télé-médecine.

Dans le premier cas on veut proposer un accès distant à l'enseignement se constituant en deux parties. La possibilité de consulter des cours sous

[1] Integrated Services Digital Networks et Broadband ISDN, soit RNIS, Réseau Numérique d'Intégration de Services et RNIS large bande

[2] Asynchronous Transfer Mode

[3] Resource reSerVation Protocol

[4] International Telecommunication Union (`http://www.itu.ch/`)

une forme multimédia, incluant par exemple des commentaires audio et des extraits vidéo de cours magistraux, ainsi que la possibilité de réaliser des cours en direct à une audience répartie en divers endroits, tout en autorisant l'intervention des participants qui peuvent interrompre le cours et poser des questions. Il faut de plus la possibilité d'échanger des documents, par exemple des exercices et des contrôles, avec une possibilité d'authentification et d'annotations de ceux-ci.

Dans le second cas la problématique est semblable mais avec une différence importante. Le but est de permettre à des médecins de réaliser une consultation sur des radiographies par exemple, en les annotant et en effectuant d'éventuelles recherches dans des bases documentaires importantes. Dans ces circonstances la qualité et la fiabilité deviennent de tout premier ordre pour ne pas perturber un diagnostic par des éléments extérieurs qui auraient altéré les données.

Commerce électronique

Le commerce électronique n'est pas en-soi une application multimédia mais concerne plutôt les domaines du paiement électronique sécurisé et de l'authentification. Cependant, pour que la vente par voie électronique soit attractive il faut qu'elle offre un service supplémentaire à la traditionnelle vente sur catalogue. Aussi bien les grandes entreprises de vente par correspondance que les concessionnaires automobiles ou les agences de voyage sont désireux de pouvoir offrir à l'acheteur potentiel une vue la plus précise et réaliste possible des produits ainsi que des commentaires vantant leurs mérites. Ce domaine d'activité semble être un bon débouché pour la réalité virtuelle ou la vidéo sur le réseau, permettant de faire visiter la voiture dernier modèle jusque dans ses moindres recoins ou offrant un aperçu des merveilles qui seront visitées lors d'un tour du monde.

Vidéo à la demande

Nous allons terminer ce rapide tour d'horizon par une application qui suscite un grand intérêt économique : la vidéo à la demande. Grâce à l'arrivée du numérique dans le monde télévisuel il va être possible d'offrir au téléspectateur le choix de ses programmes et de remplacer ainsi la traditionnelle location de cassettes vidéo. Ce domaine de la recherche est vraiment

très spécifique parce qu'il est proche de la télévision et requiert donc une qualité d'image et de son très élevée. De plus il ne s'agit pas à proprement parler de création multimédia mais plutôt de diffusion de programmes sur des canaux classiques tels que le câble ou le satellite.

2.1.2 Le World Wide Web

Le Web doit être aujourd'hui incontestablement le plus grand support de publication de documents. N'importe qui peut, à condition de posséder un ordinateur équipé d'un modem et de prendre un abonnement chez un fournisseur d'accès Internet, publier des documents au format HTML qui seront accessibles partout dans le monde. On peut alors à l'aide d'un *browser*, logiciel de navigation permettant de parcourir l'espace des documents au gré des liens hypertexte, consulter cette vaste base documentaire. De plus, l'hypertexte permet de référencer et de réutiliser du contenu déjà présent et de types très variés. Toutefois, il ne s'agit pas à proprement parler d'un environnement multimédia car l'intégration est pratiquement inexistante. Les données sont accessibles à travers des liens et il est alors possible de les ramener localement pour les consulter avec le logiciel adéquat.

Un pas vers une intégration plus importante a été fait avec l'utilisation de *plug-ins*[5] et d'applications de support (*helper applications*). Dans le premier cas le browser se repose sur un module extérieur pour traiter certains types de données, comme de la musique au format MIDI ou de la vidéo qui sera affichée directement dans la page consultée. Le second cas, plus rudimentaire, permet seulement de lancer une application tierce en lui transmettant les données qui ont été récupérées.

De nombreux plug-ins tels *Shockwave Flash*, *Headspace Beatnik*, *VXtreme Web Theater* ou *VDOLive* permettent d'ajouter des animations, du son et de la vidéo aux pages HTML. Toutefois, les possibilités d'interaction entre ces différents composants, afin de créer de véritables présentations multimédia, sont nulles. Malgré cela, le succès de Java, un nouveau langage de programmation qui a la particularité d'avoir été créé pour être portable et s'exécuter de façon sécurisée, permet d'envisager le développement d'un support pour l'exécution coordonnée d'applications au sein d'un browser (voir en 5.2.1).

[5]Module logiciel venant se greffer sur un browser pour ajouter de nouvelles fonctionnalités.

En dépit de cette faible intégration des différents médias, mais en raison de son succès phénoménal et de sa facilité d'utilisation le Web nous semble être le vecteur futur du multimédia et c'est précisément dans cette direction que nous avons orienté nos recherches.

2.1.3 Anatomie des systèmes multimédia modernes

À travers ce rapide aperçu de quelques applications multimédia on peut extraire facilement les caractéristiques requises pour un système multimédia devant les supporter.

L'ordinateur personnel restera sûrement à moyen terme le support central du multimédia. Il lui faudra une capacité de stockage importante et d'accès rapide au regard de la taille des données à manipuler. Des cartes d'extension spécialisées dans le traitement de certains types de données sont aussi nécessaires : audio, affichage haute résolution, traitement 3D, compression et décompression vidéo, etc. Enfin il devra y avoir la possibilité de se connecter à un réseau avec un débit raisonnablement élevé.

Ces dernières années les fabricants de microprocesseurs tels Intel, Sun et Hewlett Packard se sont lancés dans le développement et l'intégration à leur processeurs d'un jeu d'instructions spécialisées dans le traitement du signal, qui a pour caractéristique l'application d'une même opération à un grand nombre de données[6]. Cela a donné par exemple la technologie MMX chez Intel. Ces instructions spécialisées, qui étaient jusqu'alors réservées aux DSP[7], permettent de réaliser efficacement les traitements coûteux induits par les données et les communications multimédia, autorisant par exemple le décodage vidéo uniquement par logiciel.

On trouvera d'autres systèmes multimédia, du moins en apparence, par exemple sous la forme de bornes interactives, mais il s'agira uniquement de l'habillage spécifique d'un ordinateur classique, éventuellement agrémenté d'un écran tactile.

Même si certains systèmes d'exploitation classiques montrent rapidement leurs limites[13, 103], il est peu probable que le développement de systèmes multimédia basés sur des systèmes temps réel, mieux adaptés au traitement

[6]Parallélisme SIMD, Single Instruction, Multiple Data.
[7]Digital Signal Processor

des données temporelles, se fasse à grande échelle. La solution du "surdimensionnement", consistant à disposer de ressources matérielles et logicielles en excès pour éviter d'atteindre leur limites, semble être la voie prépondérante pour pallier les problèmes de performances. En effet la puissance des machines ne cesse de croître et leur prix de baisser. Le maillon faible reste cependant le réseau : à moins de pouvoir pratiquer la même politique, ce qui est faisable sur un réseau local en utilisant des technologies comme ATM ou Gigabit Ethernet, on doit se plier aux caprices d'Internet lors des heures de surcharge. L'éventuel déploiement des techniques de réservation évoquées en 2.1.1 permettra peut-être un jour de disposer d'une qualité de service garantie, et ce avec facturation en conséquence !

2.1.4 Conclusion

On peut extraire de cette présentation succincte du multimédia les deux caractéristiques principales des données multimédia qui vont conditionner toute la recherche dans ce domaine, et en déduire les conséquences.

- Les données sont *volumineuses*, posant des problèmes de stockage et d'accès. Des techniques de compression ont été développées, pour tenter de répondre au premier problème mais la contrepartie négative est leur tendance à gêner l'accès en rendant la structure et le codage des données plus complexes. Enfin le volume constitue aussi un handicap lors du transport de données sur des réseaux à faible bande passante, comme peut l'être Internet.

- Les données sont *temporelles*, ce qui soulève le problème de leur manipulation sous deux aspects. D'abord au travers de leur représentation, qui va nécessiter l'introduction de modèles temporels. Ensuite toujours du point de vue du transport, puisque cette dépendance vis-à-vis du temps va imposer de nouvelles contraintes très fortes sur les conditions de celui-ci, ce qui, lié au problème précédent de la taille importante de données, constitue la problématique centrale des communications multimédia.

2.2 Le facteur temps

Comme nous l'avons vu le facteur *temps* est une des deux sources de difficulté du multimédia avec la taille importante des données à manipuler. Cela est donc tout naturellement devenu un secteur majeur de la recherche multimédia.

2.2.1 Concepts de base

Classes de données

On peut tout d'abord définir deux classes de médias en fonction de leur relation vis-à-vis du temps :

- les médias *discrets* ou *statiques*, qui n'ont pas de dimension temporelle, par exemple du texte ou une image fixe ;
- les médias *continus*, qui ont un comportement temporel intrinsèque, par exemple une vidéo.

Pour pouvoir manipuler conjointement ces deux types de média il faudra évidemment ajouter une dimension temporelle aux médias discrets. Cela se fera en leur attribuant une durée, soit explicitement soit implicitement par l'interaction qu'ils auront avec les autres éléments. Alors que l'on peut distinguer les médias individuellement par ce critère, une fois intégrés dans le multimédia ils deviendront tous temporels du fait de leur durée non nulle. Un média de durée nulle est équivalent à l'absence de média du point de vue du résultat puisqu'il ne sera jamais rendu — entendu ou vu en l'occurrence.

Une seconde distinction de classe peut se faire en considérant la nature de cette relation au temps. Deux nouvelles classes sont alors définies :

- les médias *déterministes*, dont on connaît le comportement a priori, donc la durée a priori, par exemple un fichier son dont on détermine la durée à travers son en-tête ou une vidéo à partir du nombre d'images et de sa fréquence ;

- les médias *indéterministes*, par opposition aux premiers, dont on ne connaît pas le comportement a priori ; on en connaîtra la durée qu'a posteriori, c'est-à-dire une fois leur fin atteinte, par exemple un signal vidéo reçu en temps réel à travers le réseau lors d'une vidéoconférence.

Flots

Si la différence entre médias discrets et continus disparaît lors de leur restitution, car une durée est finalement associée à tout média discret, celle-ci reste bien présente au niveau fonctionnel.

Un média discret constitue un tout indivisible et il faudra disposer de l'intégralité des données le constituant avant de pouvoir le restituer. Cela n'est pas tout à fait exact, par exemple dans le cas des images JPEG à affichage progressif qui utilisent un codage hiérarchique spécifique permettant l'affichage d'une première ébauche du résultat dès les premières données disponibles, cette dernière étant ensuite améliorée au fur et à mesure que de nouvelles données sont disponibles. Toutefois on peut considérer que le média initial est fidèlement restitué uniquement quand l'intégralité des données a été récupérée.

À l'inverse, un média continu est une succession d'éléments discrets liés les uns aux autres par une relation temporelle. Par exemple une vidéo est constituée d'une suite d'images espacées de 40 millisecondes si celle-ci comporte 25 images par seconde. Il suffit donc pour, restituer correctement une vidéo à un instant donné, de disposer uniquement des données permettant de reconstituer la prochaine image à afficher.

L'approche traditionnelle de l'accès aux données sous forme de fichiers stockés localement sur un disque est aujourd'hui dépassée par l'accès à travers un réseau de communication. La prédominance de ce support de communication est telle que certains constructeurs comme Sun envisagent même la disparition du disque sur les postes utilisateur, ceux-ci accédant aux données uniquement sur des serveurs répartis sur un réseau. Toutefois, pour des raisons de performances évidentes, un disque, même de taille réduite, est indispensable pour servir de cache lors d'accès à des serveurs distants autorisant ainsi des stockages temporaires efficaces.

Cette vision locale des données, qui permettait de les envisager comme un tout accessible de façon désordonnée, disparaît donc au profit d'une approche

par *flot*[8], dans laquelle les données sont vues comme une suite d'éléments discrets, les uns remplaçant les autres suivant une séquence bien précise. Cette approche s'adapte bien aux médias temporels, dont il sera possible de restituer les premiers éléments dès leurs données disponibles alors que les suivantes seront en train d'arriver. Nous verrons toutefois en 2.2.1, avec l'exemple de MPEG, que cela n'est pas aussi simple à cause des éventuelles interdépendances temporelles dues au codage et à la compression.

Unités d'information

Les médias continus sont constitués d'une séquence temporelle d'*unités d'information*, appelées *Logical Data Units* (LDU) par Steinmetz[104], unités de données logiques. La caractérisation de ces LDU n'est pas unique pour un média donné. On peut en distinguer plusieurs types, qu'il est possible d'assembler en hiérarchies, et dont l'utilité dépendra de l'application et du niveau auquel on se trouve dans celle-ci.

Des applications d'édition mettront à la disposition de l'utilisateur une hiérarchie de LDU utile pour le type de travail auquel elle sont destinées, c'est-à-dire la manipulation d'une structure logique. Une application de montage vidéo pourra par exemple définir la vidéo, la scène, la séquence et l'image comme type de LDU, dont on peut voir la hiérarchie sur la figure 2.1. On remarque que la caractérisation des LDU n'est pas nécessairement intrinsèquement liée à la représentation des données comme c'est le cas pour l'image, mais peut être définie par une sémantique externe comme dans le cas de la scène. Certains codages offrent la possibilité d'incorporer ce type d'informations contextuelles dans le but de simplifier la tâche aux applications qui en ont besoin, pour l'édition, l'indexation, la recherche.

Au niveau beaucoup plus bas de la restitution des données la hiérarchie des LDU sera naturellement de granularité beaucoup plus fine. Prenons toujours le cas de la vidéo qui est constituée d'une suite d'images fixes. Chaque image est un ensemble de pixels, qui dans le cas d'une représentation en couleur sont caractérisés par trois coordonnées dépendantes du type de codage utilisé, RGB, YCbCr, YUV, ou un autre. Il est usuel de distinguer encore un ou plusieurs niveaux entre l'image et le pixel, par exemple par la définition

[8]Dans certains travaux une distinction entre *stream*, flot, et *flow*, flux, est opérée. Campbell[16] caractérise un flux par "la production, la transmission et l'éventuelle consommation d'un flot donné sous la forme d'une activité intégrée répondant à une spécification de qualité de service donnée".

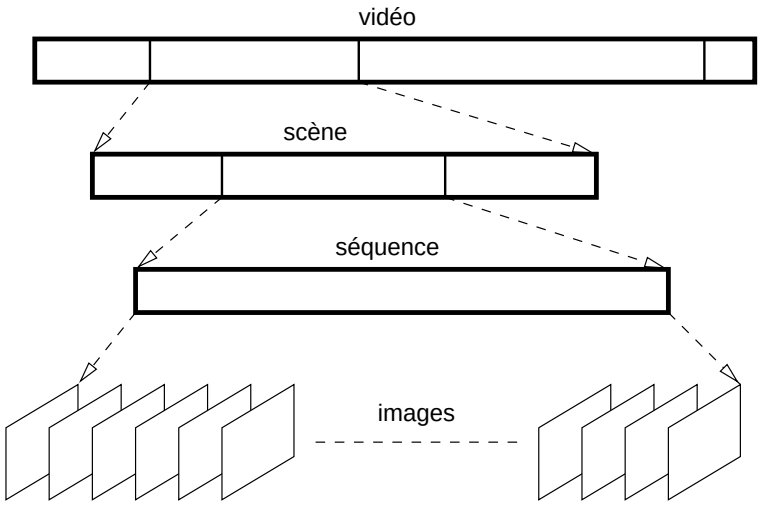


Figure 2.1: Exemples de LDU d'une vidéo

de blocs de 16×16 pixels adjacents, eux mêmes divisés en quatre blocs de 8×8 pixels, voir la figure 2.2.

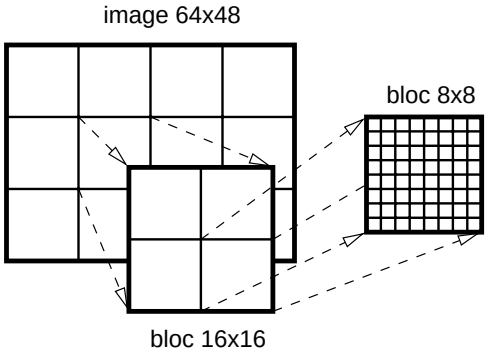


Figure 2.2: Exemples de LDU d'une image

Unités de présentation et dépendances temporelles

Les unités d'information pertinentes dans le cas de la restitution d'un média sont celles caractérisées par un instant de présentation unique, et que l'on appellera *unités de présentation*. Un média est donc constitué d'une séquence d'unités de présentation.

Pour la vidéo il s'agit de l'image. Pour le son l'échantillon satisfait à cette définition, cependant pour des contraintes de performances on ne ma-

nipule pas les échantillons individuellement. La vidéo demande en général une fréquence d'affichage autour de 25 Hz (25 images par seconde), pour l'audio cela varie de 8 000 à 48 000. Dans les couches basses du système le son est donc manipulé par groupe d'échantillons de taille fixe, constituant l'unité de présentation et correspondant en général à une durée de l'ordre de 40 ms, ce qui permet de rester en deçà des contraintes psychoacoustiques, garantissant par exemple l'effet de simultanéité entre le moment où un utilisateur arrête le son et son arrêt effectif.

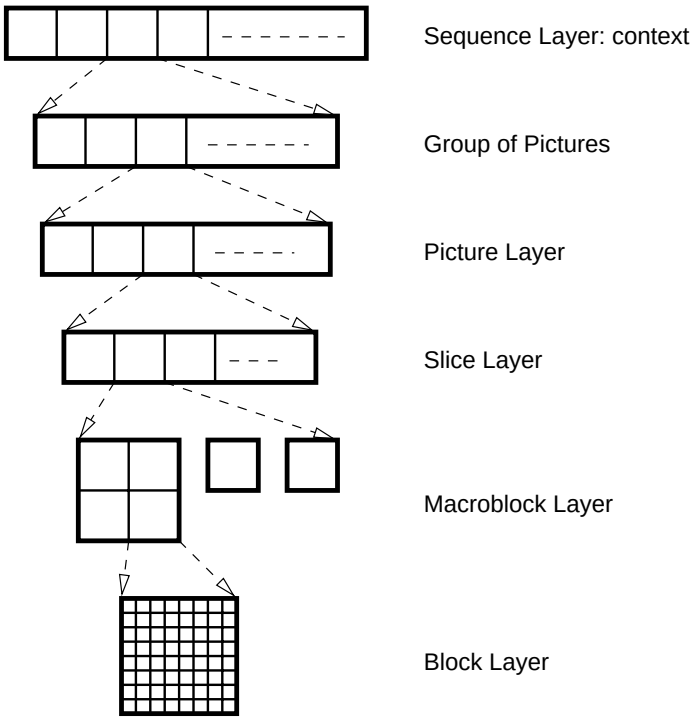


Figure 2.3: Les six couches de MPEG Video

Revenons au cas de la vidéo qui est toutefois plus complexe que ce que l'on a laissé entendre dans les paragraphes précédents. Considérons l'exemple de la vidéo MPEG[47, 62]. La compression MPEG utilise des principes de prédiction et d'interpolation qui permettent de supprimer en partie l'information redondante se trouvant dans une séquence d'images consécutives. La figure 2.3 montre la hiérarchie des six couches MPEG, définissant donc six types d'unités d'information différentes. La figure 2.4 montre les dépendances entre les images successives d'une vidéo. On constate que les images de type P, utilisant la prédiction, dépendent d'une image passée, ce qui ne crée pas de conflit avec l'approche flot dans le cas d'une restitution dans le sens normal.

Par contre, les images de type B, utilisant l'interpolation ou prédiction bidirectionnelle, posent un problème parce qu'elles nécessitent pour être décodées une image future, qui arrivera donc ultérieurement dans le flot. Dans l'absolu, l'unité de présentation constituée par l'image codée n'est donc pas toujours dépendante d'un unique instant de présentation, toutefois l'image décodée l'est.

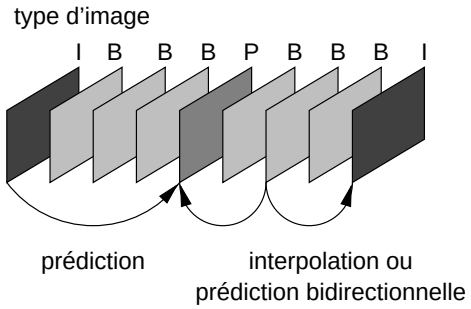


Figure 2.4: Les types d'images de MPEG Video et leurs dépendances

On peut donc distinguer la représentation de l'image avant décodage qui peut présenter de fortes dépendances temporelles, et sa représentation après décodage où l'on peut isoler des blocs de pixels indépendants. Cette particularité pose plus particulièrement des problèmes dans le cadre des communications multimédia et des solutions, consistant à modifier le flot de données qui est transmis, permettent de simplifier le traitement côté client (voir section 2.3).

Objets multimédia

Nous désignerons par *objet multimédia*, ou implicitement objet, la représentation des médias dans un système multimédia. Ce terme est plus exact et facile à manier, et permet de différencier un média, par exemple la vidéo du décollage d'Ariane IV, et l'objet le représentant, sur lequel diverses opérations sont applicables.

Nous distinguerons les objets *atomiques* représentant un média unique et les objets *composites*, désignant l'assemblage de plusieurs objets multimédia en un nouvel objet.

2.2.2 La synchronisation

Dès lors que l'on souhaite intégrer des médias temporels en vue de leur restitution, le problème de la synchronisation[9] se pose.

Il a été vu dans l'introduction que les données constituant ces médias peuvent être soit naturelles, si elles proviennent de la numérisation d'un signal, soit synthétiques, si elles sont intégralement générées par des moyens logiciels à partir de modèles, leur mélange étant par ailleurs possible. Quelle que soit la nature des données manipulées, celles-ci possèdent des propriétés temporelles intrinsèques qui, lors de la restitution, permettent de produire la sensation psychoacoustique ou psychovisuelle originale ou recherchée suivant le cas. Ces propriétés temporelles devront être impérativement respectées lors de la restitution des médias, moyennant des tolérances psychosensorielles humaines qui sont parfaitement connues. En outre s'ajoutent à cela des contraintes temporelles et psychosensorielles supplémentaires dès que plusieurs médias sont restitués simultanément.

On peut définir la synchronisation à deux niveaux, interne et externe aux objets multimédia.

- La synchronisation *intra-média* vise à assurer la continuité de la restitution des unités de présentation consécutives d'un objet atomique donné. Ce type de synchronisation est local à un objet et souvent réalisé dans les couches basses du système. Cela consiste, par exemple, à assurer que l'espacement entre les images consécutives d'une vidéo est toujours de 40 ms si celle-ci compte 25 images par seconde, ou dans le cas du son, que les blocs d'échantillons sont bien envoyés à la bonne période vers le périphérique de sortie. Dans ce dernier cas il faut à tout prix éviter d'envoyer les données trop vite ce qui occasionnerait leur perte et la disparition de portions du son, ou inversement trop lentement ce qui produirait des craquements ou au mieux des blancs.

- La synchronisation *inter-média* garantit le respect des relations et contraintes temporelles entre différents objets. Cela consiste, par exemple, à assurer qu'une vidéo et sa bande-son associée démarreront bien simultanément, et qu'une fois leur restitution terminée une image sera affichée suivie d'un texte.

[9] Nous appliquerons le terme *synchronisation* uniquement au domaine temporel. Il est également utilisé pour signifier les dépendances de contenu et le maintien de leur cohérence.

Nous allons préciser cette définition de la synchronisation inter-média. Il y a en fait une dépendance entre les deux types de synchronisation définis précédemment et l'on peut alors séparer la synchronisation inter-média en deux composantes, une qui contient cette dépendance, l'autre non.

Lorsque deux objets sont restitués simultanément, un certain niveau de corrélation doit être maintenu en permanence entre ceux-ci. Ce niveau définit un type de synchronisation plus ou moins étroite que nous appellerons par la suite *couplage*. On appelle classiquement *lip-sync*, pour *lip synchronization*, le niveau le plus élevé, qui garantit un couplage suffisant entre une séquence vidéo dans laquelle un personnage parle et la bande-son associée restituant la parole pour qu'un spectateur ne perçoive pas de décalage. Ce type de couplage fort concerne les unités de présentation des objets et est réalisé en contrôlant étroitement la synchronisation intra-média de chaque objet de manière continue.

À l'opposé, on peut extraire une synchronisation ponctuelle ou événementielle, ne s'appliquant pas sur une séquence d'unités de présentation, mais sur une unité seulement. Il s'agit de l'*ordonnancement* ou *scheduling* des objets.

La figure 2.5 illustre cette distinction. Il nous semble important de pouvoir distinguer le fait qu'un état d'un objet — une vidéo à sa 150^e image — ou un événement — la fin d'un flot audio — engendre un autre événement ponctuellement — l'affichage d'un texte — et le fait de contrôler plus ou moins étroitement le déroulement simultané de deux flots — la vidéo et le commentaire sur la figure. Il y a des situations où une synchronisation continue entre deux objets est inutile, par exemple entre une vidéo et une musique de fond, dans ce cas on ne se préoccupera que de l'ordonnancement en veillant à démarrer et arrêter les objets aux bons moments. Par contre, dans le cas d'un commentaire attaché à la vidéo il faudra en plus veiller au maintien de la synchronisation de façon fine et continue.

Nous avons distingué trois formes de synchronisation d'après les relations existant entre les objets qui en sont la cible. Une distinction supplémentaire peut se faire sur un critère additionnel, la manière dont sont définies ces relations :

- La *synchronisation directe*, ou *live*, est contrainte par la relation qui existait entre les données au moment de leur capture. Elle est donc implicitement définie. Ces données arrivent en flots continus et l'exemple typique est la vidéoconférence. Dans ce cas la marge de manœuvre est

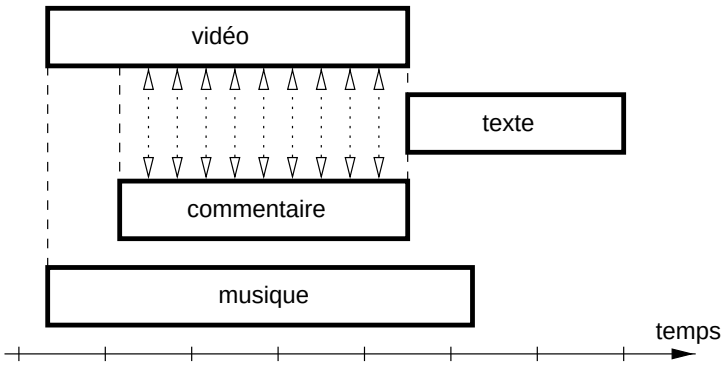


Figure 2.5: Synchronisation inter-média et ordonnancement

extrêmement réduite pour deux raisons. D'une part il faut consommer les données suffisamment rapidement pour ne pas risquer d'en perdre à cause de dépassements de capacité. Cela limite énormément les possibilités de jouer sur le stockage temporaire ou "bufferisation" de données dans le but de limiter les effets causés par les discontinuités dans les flots. D'autre part, lorsque des interactions sont permises comme c'est le cas pour la vidéoconférence, le délai entre la saisie des données et leur restitution doit être minimisé pour ne pas occasionner de gêne, lors de dialogues par exemple.

- La *synchronisation synthétique* est explicitement définie par la spécification des relations entre les différents objets multimédia à restituer. Ce type de définition exige des modèles temporels plus complexes que la simple coïncidence de marqueurs temporels utilisés dans le premier cas. De plus, les modèles de représentation peuvent être de deux types : indépendants des données, il faudra alors, pour restituer les objets correctement, analyser une spécification et s'y conformer, ou directement inclus dans le modèle de données, comme c'est le cas pour un flot MPEG incluant vidéo et audio. Ce second cas est en fait très proche de la synchronisation *live*, puisque les données sont alors multiplexées et positionnées par rapport à des marqueurs temporels.

2.2.3 Mise en œuvre de la gestion du temps

Dans le système

La gestion de la synchronisation dans les systèmes multimédia reste aujourd'hui encore une tâche délicate. Une des difficultés majeures vient du fait que la notion de temps est très délicate à manier dans un système informatique classique. La gestion de toutes les ressources, y compris le temps, passe par le système d'exploitation, or celui-ci doit exécuter une multitude de tâches et traiter une multitude d'interruptions simultanément — du moins doit-il en simuler l'effet puisque cela est impossible à moins de disposer d'un processeur par tâche — et ce processus est donc fondamentalement asynchrone. Une application tournant au dessus d'un tel système ne peut donc pas avoir la certitude que les actions qu'elle souhaite exécuter le seront bien en temps voulu. Certains travaux proposent des solutions pour améliorer la gestion des applications multimédia[80].

La solution idéale serait évidemment d'utiliser un système temps réel, avec lequel des propriétés temporelles bien précises pourraient être garanties, permettant ainsi aux applications de s'exécuter dans de bonnes conditions. Cependant, ce genre de système est complexe, coûteux et peu répandu dans le cadre grand public. De plus ils offrent des garanties extrêmement élevées qui sont beaucoup trop strictes dans le cadre multimédia, puisqu'une résolution de l'ordre de 40 millisecondes est suffisante — le son stéréo demande une précision de l'ordre de ± 11 μs mais n'est jamais traité de façon logicielle et laissé à la charge du périphérique matériel. Comme nous l'avons déjà vu, la solution du "surdimensionnement" prime actuellement, d'autant plus que les constructeurs sont lancés dans une course effrénée à la puissance : la solution toute simple consiste à disposer de machines suffisamment rapides pour qu'elles aient toujours le temps d'exécuter les tâches qu'on leur demande, dans le cas contraire il faut les changer pour de nouvelles plus rapides !

Une autre solution est celle de systèmes dédiés au multimédia comme *BeOS* développé par la société Be, Inc.[10]. Ce système a été développé pour les nouveaux types d'applications et incorpore un support spécifique au multimédia. La fonctionnalité la plus intéressante, qui est d'ailleurs aussi présente dans certains systèmes classiques comme *Solaris* de Sun, est la définition de deux types de processus légers ou *threads* en fonction de leurs priorités. La majorité des threads, de priorités inférieures, sont gérés classi-

[10] `http://www.be.com/`

quement, et ceux de priorités les plus hautes sont "temps réel", c'est-à-dire qu'ils préemptent tous les threads de priorités inférieures dès qu'ils veulent s'exécuter. Cela est évidemment assez dangereux et il ne faut exécuter que des tâches de courte durée, mais permet de gérer efficacement un processus de synchronisation par exemple.

Modèle de référence

Steinmetz et Nahrdstedt ont proposé un modèle de référence de la synchronisation[104] qui se compose de quatre niveaux, illustrés sur la figure 2.6. Les fonctionnalités de chaque niveau peuvent être utilisées par le niveau supérieur ou directement par les applications.

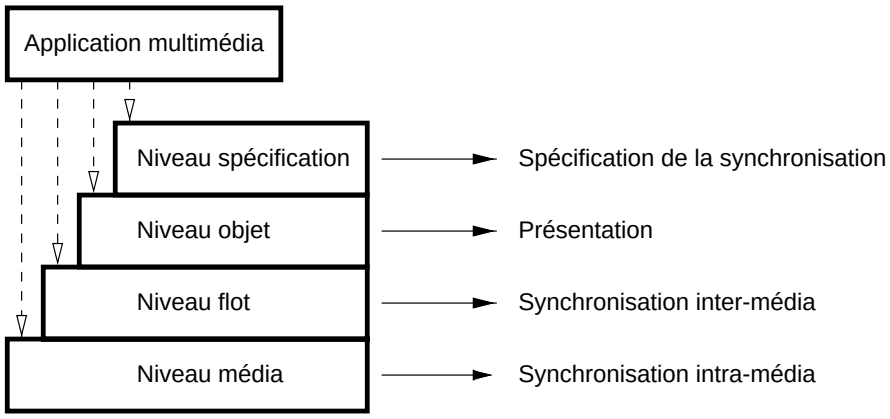


Figure 2.6: Modèle de référence de la synchronisation multimédia

- Le niveau média gère le flot continu des unités de présentation d'un média et permet d'assurer la synchronisation intra-média. Dans le cas d'un système distribué les auteurs incluent également le support réseau à ce niveau. Cela est discutable parce qu'il peut être nécessaire d'effectuer un ensemble de tâches spécifiques dépendant des protocoles utilisés, par exemple un réordonnancement de paquets ou un démultiplexage des données, qui auraient leur place dans une couche transport à part entière.

- Le niveau flot fournit l'abstraction de niveau supérieur permettant de manipuler plusieurs flots de façon transparente, à travers des opérations du type *start*, *stop*, *pause*.

- Le niveau objet propose une abstraction commune aux objets discrets et continus et gère l'ordonnancement à partir de la spécification. Aucune opération de synchronisation ne fait partie de ce niveau, par contre il est chargé de l'éventuelle initialisation des couches sous-jacentes, afin que la synchronisation se déroule correctement. Nous pouvons également discuter le fait que cette dernière fonctionnalité se trouve à un niveau aussi élevé sachant qu'elle dépend directement de niveaux beaucoup plus bas comme les conditions de transport, l'allocation de ressources, etc.

- Le niveau spécification fournit les moyens, abstraits et pratiques, pour spécifier le comportement temporel des divers objets multimédia manipulés.

2.3 Communications multimédia

Le domaine des *communications multimédia* constitue aujourd'hui un pan important de la recherche. Ce domaine s'attache à résoudre les difficultés soulevées par les deux principales caractéristiques des données multimédia que sont leur taille importante et leur dépendance temporelle.

Ces données volumineuses, lorsqu'elles sont acheminées par les moyens traditionnels, induisent des durées de transfert disproportionnées en regard de leur sensibilité au temps. La solution à ce problème peut se trouver dans le réseau et le mode de transport mais également en analysant les spécificités des différents types de média à transporter et en adaptant leur mode de représentation.

2.3.1 Transport temps réel

Une caractéristique importante pour un protocole de transport multimédia est son caractère temps réel. Il est important à ce point de faire la différence entre le domaine de la recherche temps réel classique, qui concerne des applications offrant de fortes garanties de temps réel comme celles dédiés au contrôle des systèmes embarqués en aéronautique par exemple, et les communications temps réel où il faut entendre "en direct", avec un délai relativement faible.

Afin de présenter quelques traits caractéristiques des communications multimédia, nous allons nous intéresser à un protocole particulier, RTP, dans un environnement IP[11], ce qui facilitera énormément la tâche.

Le protocole RTP[101], Real-time Transport Protocol, a été défini dans l'optique des téléconférences multipartie, mais n'est pas spécifiquement lié à ce domaine et commence à être aujourd'hui assez largement utilisé sur Internet. RTCP, RTP Control Protocol, est un protocole compagnon qui permet de transmettre des informations sur les participants à une session ainsi que des statistiques liées à chacun d'eux sur les conditions de transport des données. Ces deux protocoles se situent au niveau application et peuvent supporter plusieurs types de protocole de transport, mais ils sont généralement utilisés dans le contexte Internet au-dessus d'UDP[12].

Contrairement à ce que son nom peut laisser croire, RTP ne garantit pas le transport temps réel des données, ni même la fiabilité de celui-ci et l'ordre d'arrivée des paquets. Cette tâche est éventuellement laissée aux protocoles sous-jacents. Par contre RTP fournit un marquage temporel et séquentiel des données qui offre une notion de temps et d'ordre liés au flot.

Un bon moyen de gagner en efficacité dans la vitesse de transport des données est de limiter autant que possible le délai de transport à celui introduit par le réseau en supprimant tout autre traitement. On est alors réduit à utiliser des protocoles comme UDP, qui offre cette souplesse mais ne dispose pas de contrôles raffinés comme ceux de TCP[13] qui fournit un canal de transport fiable.

UDP et RTP ne fournissent aucune garantie quand à l'arrivée des paquets à destination et à leur ordre, l'application devant s'en charger. L'absence de ces garanties coûteuses permet de gagner en efficacité car tous les flots n'en ont pas besoin. Notamment les flots audio et vidéo.

Les données audio doivent être envoyées vers leur périphérique dans le bon ordre, donc un réordonnancement s'impose. Cependant si un paquet est manquant parce qu'il a été perdu ou arrivera plus tard, il n'est pas nécessaire d'attendre son éventuelle arrivée au-delà du délai maximal qui introduirait une discontinuité ou de demander un deuxième envoi. Des techniques de codage spécifiques[110] permettent de combler les trous laissés par les paquets

[11] Internet Protocol
[12] User Datagram Protocol
[13] Transmission Control Protocol

manquants, en les remplaçant ou en les reconstituant à partir d'information redondante intelligemment placée dans le flot de données.

Pour certains codages vidéo les blocs de données formant une image peuvent être décodés dans le désordre, évitant ainsi un réordonnancement non indispensable. La perte de données est aussi moins problématique pour la vidéo que pour l'audio car des images consécutives véhiculent en général une grande quantité d'information redondante. Le fait de sauter une image ou d'en remplacer une portion par celle de l'image précédente ne crée pas de grosses perturbations dans la perception humaine. Par contre les codages supprimant cette redondance peuvent créer des difficultés. Ils présentent le gros avantage de réduire le volume des données à transporter, mais il faut veiller à transmettre régulièrement des images indépendantes car la perte ou la dégradation d'une seule suffit à perturber toute la série d'images dépendantes.

2.3.2 Contrôle

Une des clés d'un transfert de données efficace est un contrôle s'effectuant à plusieurs niveaux. D'abord au niveau transport, où des mécanismes de contrôle de débit apportent une solution satisfaisante lorsque des pertes sont observées. Il s'agit en fait de trouver un compromis "qualité de média / pertes" acceptable. Le principe repose sur le fait que le taux de pertes diminue si l'on envoie moins de paquets sur le réseau et consiste à modifier le codage des données afin de diminuer leur taille, cela allant de pair avec une diminution de la qualité. RTCP fournit un support d'analyse des pertes par son mécanisme de statistiques.

Un second type de contrôle peut s'effectuer avant même de commencer le transport des données par la négociation des formats à utiliser. Il est par exemple inutile d'essayer de transmettre une vidéo de haute qualité sur une ligne à bas débit. Il est dans ce cas plus efficace de prendre a priori une décision quant au type de données à utiliser.

2.3.3 Qualité de service

Une solution possible au problème du transport est le contrôle et la garantie de critères de qualité de service — QoS pour *quality of service* — associés aux flot de données transportées. Cette approche à pris de l'importance avec

l'apparition des réseaux ATM[53, 91] qui offrent ce type de support de façon native.

RSVP, Resource reSerVation Protocol[8], a été développé pour assurer le contrôle de la qualité de service sur les réseaux IP. Il fournit au récepteur le moyen de réserver des ressources le long du trajet le reliant à l'émetteur des données. Cela demande toutefois des routeurs adaptés et ce type de service n'est pas disponible sur Internet. De plus il reste un problème partiellement non résolu, la facturation. Car les opérateurs supportant Internet n'entendent pas fournir un tel service gratuitement, et un système fiable de quantification reste à mettre en œuvre.

La possibilité de contrôler la qualité du transport a également été une préoccupation lors de la spécification de la future version du protocole Internet, IPv6. L'adjonction d'identificateurs de flot et de classe de service doit, dans le futur, permettre d'identifier la nature des paquets transportés et ainsi autoriser un service différencié en fonction de celle-ci. Dans l'absence d'une qualité de service avec de fortes garanties, cela permet par exemple dans un routeur de faire passer des paquets identifiés comme appartenant à un flot de vidéo temps réel avant des paquets transportant de simples courriers électroniques.

2.3.4 Multipoint

Les applications coopératives et de téléconférence multipartie présentent en outre une exigence supplémentaire, qui est la communication multipoint ou *multicast*. Ce type d'application nécessite un moyen efficace pour envoyer depuis une source les mêmes informations vers de multiples destinations. Le multicasting offre en outre un second avantage important : la possibilité de communiquer en utilisant la notion de groupe d'utilisateurs, sans avoir à traiter explicitement la gestion de multiples destinations à chaque source de données. La problématique associée[22] comporte donc deux éléments importants.

- La gestion de groupe doit fournir une abstraction acceptable de groupe d'utilisateurs dans un réseau. Sur IP ceux-ci sont représentés par une classe d'adresses particulière, les adresses multicast, qui représentent un ensemble d'adresses individuelles classiques. Associés à cela il faut donc des moyens de gestion de groupe permettant diverses opérations

comme entrer et sortir d'un groupe, et contrôlant efficacement les droits et la sécurité.

- Le routage multicast est la technique permettant d'acheminer les paquets transmis en mode multicast aux destinations faisant partie du groupe cible. Les réseaux IP supportent bien le multicast de façon locale, mais l'interconnexion de tels réseaux pose de nombreux problèmes dans le maintien de cette propriété.

Depuis 1992 existe le *MBone*[21, 30], IP Multicast Backbone, qui offre les propriétés multicast sur Internet par l'intermédiaire de tunnels multicast reliant entre elles des machines capables de réaliser le routage multicast. De nombreux logiciels peuvent aujourd'hui utiliser ce support, comme *vat*, *vic*[74], *FreePhone*[110], *Rendez-Vous* des outils d'audio et vidéoconférence ou *MiMaze* un jeu distribué.

2.4 Accès aux données

Maintenant que l'on peut créer et transporter des données multimédia de plus en plus volumineuses et complexes, se trouvant réparties à très grande échelle sur le réseau Internet, il se pose le problème d'y accéder. Pour cela il faut d'abord les stocker efficacement, puis éventuellement les indexer pour les rechercher, enfin les référencer et les localiser.

Les bases de données traditionnelles sont dépassées par l'utilisation des nouveaux types de données multimédia et il a fallu étendre celles-ci à leur nature temporelle et développer des modèles permettant l'interrogation de telles bases et la manipulation des objets qu'elles contiennent.

L'indexation des données multimédia constitue aujourd'hui un défi majeur. La masse croissante d'informations non textuelles échappe aux outils d'indexation traditionnels basés sur le texte. Il est possible de contourner cette difficulté en utilisant de la méta-information textuelle — de l'information sur le contenu — associée aux médias mais cette méthode n'est pas automatisable et nécessite l'intervention humaine, ce qui est impensable à réaliser à grande échelle, par exemple si l'on souhaite indexer l'ensemble des programmes télévisuels mondiaux. Des méthodes d'analyse des données de type audio et vidéo pour l'indexation par le contenu[35, 76, 71, 34] sont nécessaires, cependant celles-ci sont très complexes et coûteuses.

Des opérations aussi simples pour l'œil humain que la segmentation vidéo à partir des changements de scènes peuvent se révéler délicates à automatiser[117]. Le problème typique lié à ce genre d'opération est la fiabilité des méthodes d'analyse. Ces dernières peuvent produire des résultats erronés en ajoutant ou en oubliant des références. Le résultat satisfaisant est alors obtenu en sélectionnant la bonne méthode et en adaptant quelques paramètres en fonction du contenu du média à analyser.

Enfin, il n'est pas suffisant de rendre des données accessibles sur le réseau, il faut aussi pouvoir le signaler aux éventuels usagers. Le cas de la diffusion multicast sur le MBone est intéressant. Lors de la diffusion d'un événement, comme le décollage de la navette spatiale américaine, il faut pouvoir annoncer au public sa diffusion, les modalités de connexion, permettre la réalisation de cette connexion et décrire les types de données utilisés. Divers protocoles sont développés dans ce but et sont à ce jour encore à l'état d'Internet Draft : SDP[39], Session Description Protocol, SAP[38], Session Announcement Protocol, et SIP[92], Session Initiation Protocol. L'annonce de sessions peut se faire en diffusant des messages SAP contenant une description SDP. SIP se charge de l'établissement des communications en localisant les entités et en déterminant les types de médias à utiliser.

Chapitre 3

Les présentations multimédia

Dans le chapitre précédent, nous avons fait un tour d'horizon rapide du multimédia et présenté la problématique de base liée à l'introduction du temps dans la réalisation d'applications : données temporelles, synchronisation, communications. Nous allons maintenant approfondir le domaine qui nous intéresse plus particulièrement, celui des présentations multimédia.

Le but est ici de s'élever à un niveau d'abstraction supérieur. Nous avons vu quels sont les mécanismes de base mis en œuvre dans les applications multimédia. Nous souhaitons maintenant pouvoir en tirer parti de la façon la plus évidente qui soit en utilisant des applications fournissant ces services de base plutôt qu'en développant de nouvelles applications.

Une présentation multimédia est le fruit d'un travail d'édition de haut niveau consistant à organiser et intégrer des médias de natures variées afin de pouvoir les présenter dans l'espace et dans le temps d'une manière définie par l'auteur (voir figure 3.1). Elle dépend donc de deux applications, qui peuvent éventuellement être confondues : un éditeur et une application de présentation.

3.1 Spécification de présentations

Nous appelons *présentation multimédia* un ensemble de données regroupant plusieurs médias temporels de types variés dans le but d'être présentés un nombre indéfini de fois d'une manière coordonnée spécifiée par un auteur

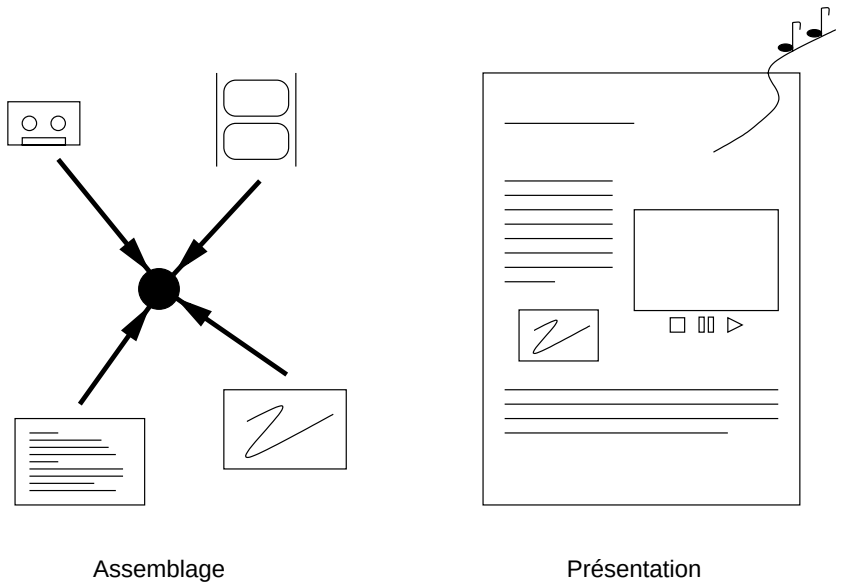


Figure 3.1: Composition des présentations multimédia

au moyen d'un procédé quelconque. Deux exécutions de la présentation ne produiront pas nécessairement le même résultat — ceci est vrai par exemple lorsqu'il y a des interactions avec l'utilisateur ou l'utilisation de flots délivrés en direct.

La modélisation multimédia s'intéresse principalement à l'aspect temporel de la spécification des présentations multimédia. Nous appelons *scénario* cette partie purement temporelle de la spécification. Toutefois, le terme présentation est souvent utilisé dans le sens de scénario, car il est vrai que le principal intérêt réside dans l'étude des nouveaux problèmes posés par l'introduction du temps.

Un scénario spécifie le comportement temporel souhaité des objets, c'est-à-dire leur synchronisation. Nous avons abordé la synchronisation et sa mise en œuvre dans la section 2.2. Dans ce chapitre, nous allons principalement traiter la spécification de la synchronisation.

3.1.1 Analyse du problème

L'apparition du facteur temporel avec les données multimédia a imposé de nouvelles contraintes et de nouvelles approches dans la réalisation des

applications. Celles-ci vont donc naturellement se répercuter à un niveau d'abstraction plus élevé avec la définition des présentations multimédia. La spécification de scénarios temporels de présentations multimédia nécessite de nouveaux supports formels.

Nouveaux modèles

D'abord, il faut être capable de prendre en compte le temps comme nouvelle dimension, ce qui nécessite un *modèle temporel* afin de le conceptualiser et le manipuler. La notion de temps est omniprésente dans l'informatique, sous la forme la plus basique qui soit, d'une horloge battant à une fréquence fixe. L'exécution des instructions par le processeur, le transfert des données sur le bus, sont soumis à ce type de contrainte. Mais, nous nous trouvons au niveau le plus bas, il s'agit dans ce cas de manipuler des données basiques elles aussi, sous la forme de bits et de façon séquentielle. À l'opposé, les systèmes multimédia manipulent une quantité importante de données, de toute nature, ayant des rapports différents au temps, et de manière concurrente puisque plusieurs médias peuvent être présentés simultanément. Une abstraction de haut niveau est donc nécessaire à la spécification, qui sera finalement transformée dans la seule représentation que connait la machine, citée plus haut, au passage à travers les couches logicielles lors de la présentation.

Ensuite, basé sur ce modèle temporel, il faut définir un *modèle de synchronisation* permettant de définir les dépendances temporelles entre les objets multimédia, offrant ainsi la possibilité de spécifier la synchronisation. Le modèle peut autoriser la définition de toute dépendance temporelle susceptible d'exister entre les objets ou, de façon plus restrictive, uniquement un sous-ensemble de celles-ci que l'on aura jugé suffisant.

Comme il a été vu en 2.2.2, la synchronisation se constitue de plusieurs sous-problèmes. La synchronisation intra-média ne sera pas directement considérée dans un modèle de synchronisation puisqu'elle est étroitement liée au média en question et gérée par la partie logicielle en charge de sa restitution. Par contre, elle pourra être contrôlée indirectement de plusieurs façons, par exemple par l'utilisation d'un facteur de vitesse ou par extension/contraction en jouant sur la durée totale du média.

Par contre, la définition de la synchronisation inter-média est centrale. Nous avons introduit en 2.2.2 une distinction entre couplage et ordonnancement (ou scheduling), définissant ainsi deux autres aspects de la synchroni-

sation.

- La composition temporelle spécifie l'ordonnancement des objets, c'est-à-dire leur position relative dans le temps. Il y a alors la possibilité de définir précisément le déroulement d'une présentation de son début à sa fin.
- Le mode de couplage entre objets définit qualitativement la nature des dépendances temporelles entre des objets présentés simultanément. Ils peuvent être totalement indépendants, faiblement couplés dans le cas d'un commentaire audio associé à une séquence d'images, ou fortement couplés dans le cas de la synchronisation entre un personnage parlant dans une vidéo et le doublage de sa voix (*lip-sync*).

Plus en détail

On peut isoler trois étapes nécessaires à la réalisation d'une présentation multimédia, en partant de médias indépendants sous forme brute pour aller jusqu'à la restitution.

- L'*édition*, c'est-à-dire la création de la présentation multimédia et son éventuelle modification. Cette phase doit permettre de spécifier le comportement désiré des objets constituant la présentation, aussi bien temporel, spatial, que vis-à-vis des interactions externes. On distingue ici deux caractéristiques importantes :
  - le niveau d'interaction avec l'utilisateur est très élevé ce qui nécessite l'utilisation de modèles de synchronisation de haut niveau en vue d'offrir des abstractions satisfaisantes facile à conceptualiser et à manipuler mentalement ;
  - l'édition est un processus incrémental qui demande des modèles adaptés à ce type d'approche, consistant, suivant les cas, à partir d'éléments précis et procéder par construction, ou au contraire à définir une structure globale qui est petit à petit remplie.
- Le *stockage* et le *transfert* de la présentation multimédia. Dans la majorité des cas une présentation est créée dans le but d'être présentée ultérieurement et ce plusieurs fois. Il faut donc pouvoir la stocker et la transférer sur un réseau le cas échéant. Plusieurs approches sont possibles, dont les deux extrêmes sont les suivantes :

  - d'une part, l'approche intégrée, dans laquelle les médias seront stockés soit sous la forme du résultat attendu de la présentation en les multiplexant comme on peut le faire avec *QuickTime* par exemple, et dont le résultat constitue plus un nouveau média qu'une présentation multimédia comme nous l'avons définie, soit sous la forme de code exécutable associé aux données, créant ainsi une application réalisant la présentation ;
  - d'autre part, l'approche par référence, dans laquelle les médias et la spécification de la présentation sont tous stockés de façon indépendante et éventuellement distribuée, laissant à la charge de l'application de restitution le soin de récupérer les données déclarées dans la spécification.

  Dans le premier cas, toute trace du modèle de synchronisation utilisé lors de l'édition disparaît, puisqu'il ne reste que des données brutes sous forme binaire. Dans le second cas, la spécification des données étant séparée des données, il est plus facile de la conserver sous une forme structurée fidèle aux modèles utilisés.

- La *présentation*, qui désigne ici non pas le contenu mais l'action de présenter, que l'on appellera également exécution ou restitution pour éviter répétitions et confusions. Celle-ci doit reposer sur les fonctionnalités fournies par le système multimédia sur laquelle elle est effectuée. Ce système peut être simplement un système d'exploitation dans le cas où la présentation est en fait une application à part entière, ou alors se présenter sous la forme d'une application complexe qui restituera la présentation multimédia à partir de l'analyse de sa spécification et de la synthèse de la synchronisation. Certains modèles reposent sur une notion de temps élastique, c'est-à-dire que les objets n'ont pas de durée fixe, mais celle-ci est bornée dans un intervalle. Dans ce cas, des mécanismes de formatage temporel — par analogie au formatage spatial — sont nécessaires pour déterminer les durées associées aux objets en fonction des contraintes auxquelles ils sont soumis.

Nous verrons plus loin que les modèles temporels, de synchronisation et de données utilisés dans ces trois étapes ne sont pas nécessairement les mêmes, chacune d'elles ayant des contraintes différentes. Les modèles utilisés lors de l'édition seront généralement de haut niveau alors que ceux utilisés pour la présentation seront plus proches de la machine, qui impose notamment une certaine approche du temps. Toutefois, ces deux phases sont

particulières puisqu'elles ont recours dans leur déroulement à des modèles et à des représentations internes utilisées par les applications d'édition et de présentation.

C'est l'opposé en ce qui concerne le stockage et la transmission : la représentation des présentations devant être traduite sous la forme de fichiers ou de flots. Nous verrons que l'approche utilisée ici sera de plus ou moins haut niveau suivant que le problème aura été considéré du point de vue de l'édition ou de la présentation. C'est finalement la représentation utilisée, dans ce cas, qui correspond à la définition donnée en début de section 3.1 et constituera la présentation multimédia.

3.1.2 Caractéristiques recherchées

Nous avons vu précédemment que les présentations multimédia constituent une manière d'aborder la réalisation d'applications multimédia à plus haut niveau. Les modèles proposés doivent fournir des abstractions permettant de mettre en œuvre d'une manière intuitive et simple des fonctionnalités couramment utilisées dans la réalisation de présentations. Cependant, afin d'être vraiment une alternative intéressante à la conception d'applications multimédia de bas niveau, ces modèles ne doivent pas être simplistes et offrir au niveau supérieur les caractéristiques trouvées dans les applications.

L'*interactivité* est indispensable pour plusieurs raisons. D'abord sur le plan de la conception, cela laisse à l'auteur la possibilité de structurer ses présentations de façon à les rendre attractives en provoquant des interactions avec l'utilisateur ou en lui laissant un contrôle total sur le déroulement par l'utilisation de liens hypermédia. Ainsi, les présentations ne seront plus figées mais offriront des variations dans leur restitution. Il est également souhaitable de laisser à l'utilisateur un contrôle "générique", de type magnétoscope, sur le déroulement des présentations, mais cela dépend de l'application de présentation et non des modèles utilisés.

On peut pousser encore plus loin l'idée d'interaction avec l'utilisateur par l'emploi d'objets qui ne sont plus des médias traditionnels mais des flots reçus en direct depuis le réseau ou du code exécutable pour réaliser des interactions plus complexes que de simples choix de liens, par exemple en effectuant des requêtes sur des machines distantes. Une seconde caractéristique importante apparaît donc ici.

L'utilisation d'objets de *durée inconnue* au moment de la création de la présentation. C'est le cas dans les exemples cités ci-dessus. Un flot de données reçu en direct est de durée a priori inconnue, celle-ci étant connue quand il se termine, soit par suspension de l'émission, soit par arrêt de l'utilisateur. De même, l'exécution de code est de durée inconnue et cela à plus forte raison si une communication sur le réseau est établie. Cette caractéristique est absente de certains systèmes multimédia car il sera vu par la suite que certains modèles de synchronisation ne peuvent manipuler que des objets dont la durée est fixe et connue a priori.

Ces deux caractéristiques confèrent, aux présentations multimédia les utilisant, un comportement indéterministe dû à l'utilisation d'objets multimédia au comportement indéterministe, comme il a été vu en 2.2.1, ou alors à l'utilisation d'événements extérieurs dont la position dans le temps, voire l'existence, sont inconnus a priori, par exemple les interactions utilisateur (un utilisateur peut presser une touche à n'importe quel moment de la présentation ou ne pas toucher au clavier du tout).

Les événements indéterministes ne pouvant, par définition, pas être connus statiquement, il faut traiter leur cas de manière dynamique au moment de la présentation. Leur gestion, et plus particulièrement dans le cas des interactions avec l'utilisateur, reste aujourd'hui le problème le moins bien maîtrisé. La méthode le plus souvent employée reste le recours au cas particuliers que l'on sait traiter, les autres configurations étant refusées.

3.1.3 Structuration de l'information

Les données numériques brutes, sous la forme d'un ensemble de bits, sont bien sûr incompréhensibles pour l'homme. Au fil de l'évolution de l'informatique, des abstractions d'un niveau élevé ont été développées afin d'offrir la possibilité de structurer toujours plus efficacement l'information manipulée. On peut citer par exemple les systèmes de fichiers, les bases de données, les langages de programmation, l'approche orientée objet, la notion de document, etc.

La structuration de l'information spécifiant une présentation est fondamentale dans sa perception et sa manipulation par un utilisateur humain. Il faut lui offrir des abstractions suffisamment puissantes pour faciliter la représentation mentale de cette information et ainsi pouvoir la manipuler efficacement. La *modularité* offre des réponses à ces exigences. Elle permet d'en-

capsuler l'information de manière hiérarchique dans des entités autonomes. Celles-ci peuvent alors être manipulée comme des objets à part entière et ce au niveau d'abstraction voulu. L'approche modulaire est indispensable au processus d'édition car elle offre deux nouvelles caractéristiques primordiale à l'efficacité de celui-ci.

- L'édition *incrémentale*, qui consiste à procéder par étapes successives, est un gage de clarté et d'efficacité. Cela permet à l'auteur de choisir la méthode qu'il souhaite pour construire sa présentation, en partant d'une trame générale et en raffinant de manière itérative les différents éléments ou en construisant par le bas les différents modules pour ensuite les regrouper en modules plus complexes par exemple.

- La *réutilisation* de travaux déjà existants est un moyen permettant de profiter efficacement de ce qui a déjà été réalisé. La modularité rend cette technique facile à mettre en œuvre puisqu'il devient possible de prélever des parties autonomes dans des présentations existantes afin de les réinjecter dans de nouvelles. Lorsqu'une approche par référence est utilisée, cela permet également de référencer des sous-ensembles d'autres présentations, ce qui est une manière indirecte de les réutiliser.

Dimensions logique et spatiale

Dans l'approche document structuré classique, on distingue la dimension logique et la dimension spatiale qui se retrouvent exprimées dans le document par la structure logique et la structure physique[90].

- La *structure logique* divise un document en une hiérarchie d'éléments dont la sémantique est liée au type de document. Ces éléments sont des entités autonomes qui confèrent au document une structure modulaire. Par exemple, on peut structurer un rapport sous la forme d'un ensemble de chapitres, chacun étant composé de sections, etc.

- La *structure physique* d'un document dépend de deux types de paramètres. Les premiers sont liés au document et définissent son style, comme par exemple l'interligne, et les seconds sont propres au support physique de représentation, comme par exemple les dimensions de la page.

Dans ce cadre, une représentation physique d'un document est en fait une projection de sa structure logique sur le domaine physique, qui est dans ce cas de nature spatiale, en général la page.

Les documents électroniques se sont vus enrichis d'une nouvelle structure. La structure logique d'un document étant hiérarchique, elle est en quelque sorte fermée, l'accès se faisant à partir du sommet. L'utilisation des liens hypertextes permet de contourner cet inconvénient en autorisant des relations transverses dans la structure du document, c'est-à-dire en ne respectant pas sa nature hiérarchique, et également des relations vers l'extérieur — d'autres documents ou des portions. Un lien hypertexte établit une relation directe entre une ancre origine et une ancre destination, et permet de rejoindre directement la destination depuis l'origine lorsqu'il est activé par l'utilisateur. De cette manière, il est possible de naviguer au sein d'une grande quantité de documents de façon plus ou moins anarchique sans utiliser d'organisation hiérarchique traditionnelle.

- La *structure hypertexte* d'un document est partagée entre les deux aspects logique et physique. Un lien hypertexte est une entité abstraite du point de vue sémantique, dans le sens où aucune sémantique précise ne lui est associée en dehors de la référence. Un auteur peut donc l'utiliser dans un but de structuration logique, par exemple pour définir des références bibliographiques ou des notes. Mais les liens peuvent également être utilisés dans un but purement de navigation, comme par exemple quand un document est divisé en plusieurs pages avec un lien en bas de chacune pour passer à la suivante, chose couramment rencontrée sur le Web.

Dimension temporelle

Afin de définir un modèle adapté au multimédia, il faut ajouter les notions de temps et d'hypermédia. Une solution consiste à ajouter aux deux dimensions logique et spatiale vues précédemment, une dimension temporelle, ce qui a été fait dans *Madeus*[51, 60], système sur lequel nous reviendrons plus loin. Cette approche suppose une indépendance entre les aspects logiques, spatiaux et temporels des documents.

Dans le cadre multimédia, la dimension temporelle a des répercutions sur les liens hypertextes. Ces derniers une fois munis d'une composante tem-

porelle deviennent des liens hypermédia permettant de naviguer à la fois à travers l'espace et le temps. Un lien est activable lorsque l'élément auquel il est attaché est lui-même actif, ce qui définit l'ancre d'origine comme étant un domaine spatial, par exemple la fraction d'écran où est présentée une vidéo, et un intervalle de temps, la période pendant laquelle est montrée la vidéo. De la même façon, une notion de temps est associée à la destination, en général un instant singulier de l'élément cible, par exemple le début d'une séquence audio.

En plus des structures physiques et logiques vues plus haut, apparaissent les deux structures suivantes (figure 3.2).

- La *structure temporelle* définit les dépendances temporelles entre les éléments. Ces dépendances peuvent exister entre des objets temporels ou statiques. Dans ce dernier cas, une durée peut y être associée soit explicitement, soit implicitement comme résultante des autres dépendances.
- La *structure hypermédia*, qui spécifie des liens vers des portions de la même présentation ou vers d'autres entités extérieures. La sémantique de ces liens est à l'appréciation de l'auteur (référence, note, voir aussi, etc.).

Cette approche consistant à isoler les quatre dimensions logique, spatiale, temporelle et hypermédia vise à conserver l'approche document initiale. La structure logique est très proche de la structure spatiale, ce type de document étant initialement destiné à être formaté pour un support physique. Bien qu'elle soit hiérarchique, la structure logique exprime la disposition linéaire d'éléments en relation avec le précédent et le suivant qui seront juxtaposés lors de la mise en page. La représentation spatiale est implicitement présente dans la structure logique : une section composée des paragraphes §1, §2 et §3 sera représentée par la succession de ces paragraphes, moyennant quelques aléas liés à la mise en page provoquant éventuellement un changement de colonne, de page ou l'insertion d'un élément flottant comme une figure.

Structure des présentations multimédia

La dimension temporelle est prépondérante dans le cadre des présentations multimédia et elle exprime en partie les relations sémantiques entre les différentes entités. En outre, l'organisation spatiale demande plus de dynamicité.

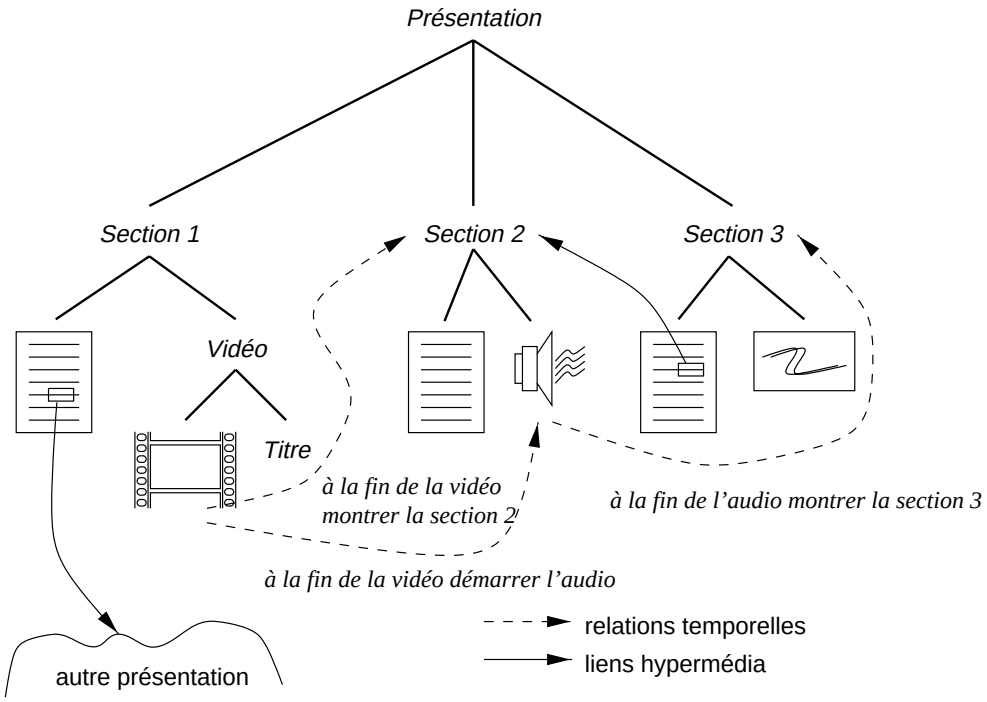


Figure 3.2: Exemple de structure d'une présentation

Les nombreux types de médias disponibles remplacent en partie le texte et la disposition des éléments sous la forme d'un flot n'est plus primordiale. Le formatage statique est incompatible avec l'utilisation du mouvement pour créer divers effets de transition ou afin d'attirer l'attention de l'utilisateur.

La structure logique d'un document dépend donc beaucoup de son aspect temporel. Par exemple, un diaporama, dont chaque image est présentée avec un commentaire parlé, et dont les transitions sont marquées par un effet sonore en rapport avec l'image suivante. Cet exemple a un aspect linéaire semblable à un document classique, il serait d'ailleurs facile de le convertir en une suite d'images sous lesquelles figurerait le commentaire écrit. La structure logique est ici très proche du déroulement temporel et non plus de la disposition spatiale, dans ce cas très basique puisque chaque nouvelle image remplacera la précédente.

On peut voir une présentation multimédia comme une structure logique complexe dont la restitution est la projection sur la dimension physique. La structure logique définit les dépendances sémantiques entre les éléments, qui sont en partie basées sur les dépendances temporelles. La projection sur la dimension physique consiste à associer aux éléments des ressources physiques,

c'est-à-dire des zones d'espace où s'afficher et des zones de temps pendant lesquelles s'activer.

Comme nous le verrons plus loin, certains modèles, comme *CMIF*[109] ou *Menkalinan*[55] exposent une structure logique temporelle qui est une hiérarchie d'opérateurs. Par exemple, dans le cas de *CMIF*, celle-ci repose sur deux opérateurs, séquentiel et parallèle, spécifiant si des objets sont présentés à la suite ou simultanément. Ceci permet de spécifier grossièrement la structure de la présentation qui est raffinée en utilisant un autre moyen.

3.2 Spécification de la synchronisation

3.2.1 Modèles temporels

Avant d'aborder les principaux modèles servant à spécifier les relations de synchronisation entre les objets multimédia, nous allons nous intéresser à la représentation du temps. On distingue deux approches[114], suivant que l'on représente le temps comme un ensemble d'*instants* ou d'*intervalles*. Le terme *point* est également utilisé pour instant. Il découle de ces deux approches deux manières de spécifier les relations temporelles que nous exposerons également.

Par instants

Un instant spécifie une position dans le temps à laquelle on peut attribuer une valeur et possède une durée nulle. On peut exprimer la position relative de deux instants a et b à partir de trois relations : a peut être avant b ($a < b$), égal à b ($a = b$) ou après b ($a > b$). Cet ensemble est suffisant pour ordonner des instants connus, donc passés. Toutefois, dans le cadre de la définition de scénarios, il est intéressant de pouvoir disposer de relations indéfinies, qui permettent de spécifier les dépendances temporelles entre des instants dont les valeurs ne seront connues précisément que lors de l'exécution de la présentation. Pour cela, on considère les relations obtenues par disjonction, soit $<$, $\leq$, $=$, $\geq$, $>$, $\neq$, $\emptyset$ et ? qui signifie $\{<, =, >\}$, soit la relation indéfinie. Sachant que de petits écarts sont imperceptibles lors d'une présentation, on ignore les relations $\leq$, $\geq$ et $\neq$, qui ne se distinguent respectivement de $<$, $>$ et ? qu'en un seul point. De plus, on souhaite définir des scénarios cohérents donc $\emptyset$ est également exclu. Il reste finalement quatre relations $<$, $=$, $>$, et

?.

Ce modèle est utilisé dans la définition de scénarios en définissant des points particuliers sur les objets, le début et la fin étant les plus évidents, et en spécifiant un ensemble cohérent de relations constituant un graphe d'instants.

Par intervalles

Un intervalle est, quant à lui, défini par deux instants, un de début et un de fin, et possède une durée non nulle lorsque ces deux instants sont distincts[1]. L'approche par intervalles se prête bien à la conception multimédia car elle permet de représenter les différents éléments de façon purement temporelle en faisant abstraction de leur nature réelle. L'intervalle représente alors la durée de présentation de l'élément en question.

Les relations entre deux intervalles sont au nombre de treize[37], mais on peut les ramener à sept en supprimant les relations inverses de six d'entre elles, l'égalité étant son propre inverse. On les connaît sous cette forme sous le nom de *relations d'Allen*[2]. La figure 3.3 illustre ces relations, les indices *beg* et *end* signifiant respectivement les instants de début et fin des intervalles. Elles sont de deux types suivant qu'elles expriment la séquentialité des deux intervalles, pour *before* et *meets*, c'est à dire $A_{end} \leq B_{beg}$, ou leur parallélisme, les relations restantes.

3.2.2 Commentaires

De nombreux modèles utilisent la notion d'intervalle pour représenter les objets dans le domaine temporel. Toutefois, cela n'implique pas l'utilisation de relations sur les intervalles. Il est en effet possible de spécifier des contraintes basées sur les instants en utilisant des points singuliers des intervalles, le début, la fin ou des instants internes définis au préalable.

Les relations exposées ci-dessus, sur les instants comme sur les intervalles, ne permettent pas en temps que telles de définir des scénarios précis. Elles sont basées sur des relations d'ordre et permettent de décrire grossièrement

[1] On peut également définir un intervalle à partir d'un seul instant et de sa durée.

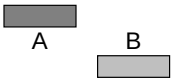
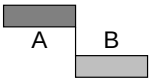
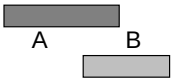
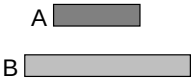
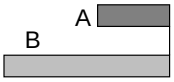
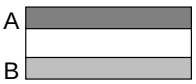
Relation	Représentation	Relations sur les instants de début et fin
A before B		$A_{end} < B_{beg}$
A meets B		$A_{end} = B_{beg}$
A overlaps B		$A_{beg} < B_{beg} < A_{end} < B_{end}$
A during B		$B_{beg} < A_{beg} < A_{end} < B_{end}$
A starts B		$A_{beg} = B_{beg}$
A finishes B		$A_{end} = B_{end}$
A equals B		$(A_{beg} = B_{beg}) \wedge (A_{end} = B_{end})$

Figure 3.3: Les relations d'Allen

une configuration temporelle mais pas de spécifier clairement les comportements recherchés. Par exemple, $(a < b)$ ne permet pas de savoir comment disposer ces instants ni quelle est la durée les séparant. De même pour (A during B) on ne connaît pas les délais entre début et fin des deux intervalles. Pour ces raisons, de nombreux modèles, enrichissants ceux-ci, ont été proposés afin de permettre la spécification de la synchronisation dans un scénario.

3.2.3 Méthodes de spécification

Parallèlement à ces deux représentations du temps, il existe diverses méthodes de spécification de la synchronisation. Ces méthodes peuvent reposer explicitement sur un des deux modèles temporels vus ci-dessus, comme les graphes d'instants, ou alors être utilisables indifféremment avec l'un des deux. Nous présentons rapidement ci-dessous les principales méthodes de spécification de la synchronisation que nous détaillerons lors de la description de certains modèles de présentation multimédia dans la section suivante.

Axe temporel

La spécification à l'aide d'axes temporels, ou *time-lines*, est la méthode la plus simple et la première utilisée. Elle se différencie des méthodes suivantes parce que la spécification ne se fait pas par définition de relations entre les différents objets mais en positionnant de manière absolue sur un axe temporel les différents éléments constituant une présentation. Cette méthode est directement comparable au mixage audio réalisé en studio : un ensemble de pistes est défini et, sur chacune d'elles, le comportement d'objets multimédia est spécifié (figure 3.4).

Il n'y a pas de spécification de relations entre les différents objets. À chaque point a appartenant à un média est associée une valeur du temps t_a relative au début de la présentation. La présentation s'exécute juste en parcourant l'axe à la vitesse du temps et en effectuant les actions nécessaires à la restitution des éléments rencontrés lors de cette progression.

L'avantage majeur de ce modèle est sa simplicité extrême et donc sa facilité d'appréhension due au fait qu'il calque la réalité, l'ordonnancement des événements étant immédiat lorsque l'on regarde une représentation du

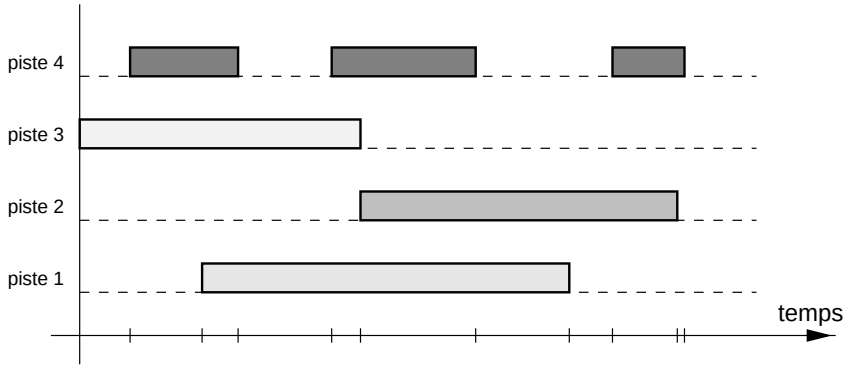


Figure 3.4: Exemple de spécification de type *time-line*

scénario.

En contrepartie, les inconvénients sont nombreux. Le positionnement étant absolu, si l'on déplace un objet sur l'axe et que l'on souhaite conserver une coïncidence particulière avec un autre objet, il faudra également déplacer ce dernier. Lorsqu'une multitude d'objets est concernée, cela devient fastidieux. Cependant, cet obstacle peut être contourné par une interface utilisateur adéquate permettant de lier temporairement la position d'un groupe d'objets et de les déplacer d'un seul bloc. Ce même problème est plus difficilement contournable quand on modifie la durée d'un élément par exemple. Dans ce cas, le repositionnement à la main peut s'avérer inévitable.

Un second inconvénient majeur est l'impossibilité d'inclure des objets au comportement indéterministe : la date des événements les concernant étant inconnue, il n'est pas possible de les placer de manière absolue sur l'axe. Toutefois, le cas particulier des interactions utilisateur est soluble comme nous le verrons par la suite.

Une solution pour amener un peu de souplesse dans l'édition à base de time-lines est de manipuler plusieurs axes. Une présentation peut être découpée en plusieurs parties indépendantes, chacune d'elles possédant sa propre base de temps. Lorsqu'une transition d'une partie vers une autre s'effectue, il y a également changement d'axe temporel. Les modèles utilisant le *time-line* arborescent et le *time-line* multiple qui seront vus plus loin sont de ce type.

Graphes d'instants

La méthode reposant sur les graphes d'instants consiste à relier des instants d'objets distincts à l'aide d'une ou de plusieurs relations. Dans le premier cas, on parle de point de synchronisation. Un point de synchronisation exprime une relation de simultanéité entre plusieurs instants, l'égalité temporelle. Ainsi, placé entre la fin d'un objet A le début de B il exprimera que B doit démarrer lorsque A termine. L'utilisation d'objets vides permet dans ce cas de simuler des délais entre les événements. La figure 3.5 illustre l'utilisation des points de synchronisation.

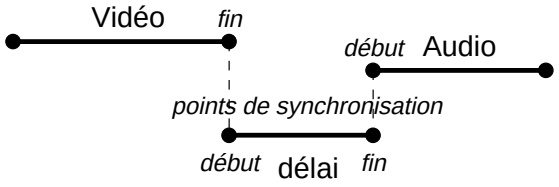


Figure 3.5: Points de synchronisation

Il est possible d'enrichir le graphe en ajoutant les relations de précédence et succession à la relation de simultanéité. Cela permet de supprimer les délais introduits artificiellement sous la forme d'objets mais complique le graphe puisque les délais seront ajoutés aux relations, voir figure 3.6. Il est possible de ne conserver qu'une relation munie d'un délai δ pouvant être négatif pour exprimer la précédence, nul pour la simultanéité et positif pour la succession.

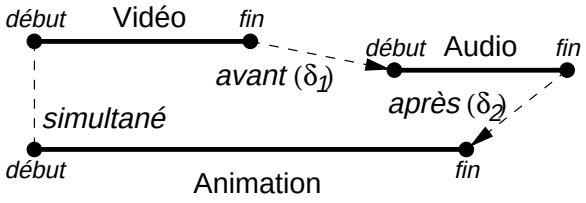


Figure 3.6: Graphe d'instants

Les avantages des graphes d'instants sont multiples. D'abord, il s'agit d'une manière très intuitive de spécifier des relations entre des objets à partir de points de références qui sont faciles à appréhender comme le début ou la fin d'un objet. De plus, la synchronisation peut se faire en des points quelconques des objets qu'il suffit de définir pour ensuite les référencer. Contrairement au modèle à axe temporel, il est possible d'inclure des objets

au comportement indéterministe dans la spécification puisque l'on peut les référencer sans notion explicite de temps.

Un inconvénient est le manque de structuration qui peut amener à des graphes énormes et ingérables lorsque les objets et les relations sont nombreuses. Il y a toutefois moyen d'encapsuler des objets et leurs relations dans un objet abstrait afin de clarifier la spécification. Mais, l'inconvénient principal est la possibilité de spécifier des scénarios incohérents, ce qui arrive si l'on ajoute la relation (Animation.fin avant(δ_3) Audio.début) par exemple. Des techniques de vérification devront donc être associées pour garantir qu'une présentation en accord avec le scénario défini est bien possible.

Événements

La synchronisation par événements fonctionne de la manière suivante. Un événement provoque une action sur l'objet auquel il est destiné. Par exemple, démarre ou arrête. Les événements sont générés par d'autres objets ou une horloge en réponse à une action ou à une transition d'état. Par exemple, pour démarrer B lorsque A se termine, un événement "démarre" destiné à B est associé à l'état fin de A. Lorsque A arrive à sa fin l'événement est envoyé à B provoquant son démarrage. Ce type de synchronisation est utilisé dans le standard PREMO qui sera vu plus loin.

Relations sur les intervalles

Il est également possible de spécifier la synchronisation en utilisant des relations temporelles sur les intervalles. De nombreux modèles on été proposés en vue d'améliorer les relations d'Allen, en introduisant des délais ou en définissant un ensemble plus restreint de relations, mais plus expressives par exemple. Les objets sont dans ce cas représentés par un ensemble d'intervalles sur lesquels un ensemble de contraintes est exprimé. Dans le cas de l'exemple de la figure 3.6 et avec les relations d'Allen, cela donne :

- Vidéo before Audio
- Vidéo starts Animation
- Animation overlaps Audio

Ce type de modèle a l'avantage d'offrir une abstraction de haut niveau en représentant tous les objets d'une façon homogène par un intervalle. En contrepartie, l'inconvénient est l'impossibilité de synchroniser des objets sur des événements internes. Par exemple, si l'on désire synchroniser un objet sur la transition entre deux séquences dans une vidéo, il faudra diviser cette vidéo en deux objets. Cela devient réellement problématique lorsqu'il faut répéter cette manœuvre à grande échelle, par exemple pour synchroniser des sous-titres avec une vidéo.

Comme pour le modèle précédent, celui-ci tolère les objets indéterministes, un intervalle pouvant représenter une durée a priori inconnue, et il est possible de spécifier des scénarios incohérents, par exemple en ajoutant la relation (Animation before Audio).

Réseaux de Petri temporisés

Les réseaux de Petri temporisés, ou TPN pour *Timed Petri Nets*, sont une extension des réseaux de Petri classiques auxquels on ajoute la notion de durée. À chaque place est associée une durée représentant le temps que le jeton doit rester bloqué avant qu'il ne puisse transiter. Une transition a lieu lorsque tous les jetons d'entrée sont libres, dans ce cas un jeton est ajouté à toutes les places de sortie. La figure 3.7 présente l'exemple déjà utilisé précédemment.

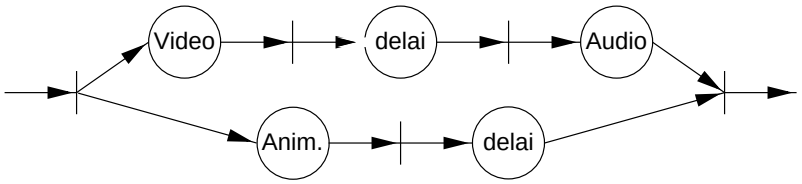


Figure 3.7: Réseaux de Petri temporisés

L'utilisation des réseaux de Petri dans la modélisation multimédia est très répandue. Le but est de réutiliser les connaissances acquises en vérification de systèmes à l'aide de la théorie associée aux réseaux de Petri dans le cadre du multimédia. La définition de modèles de synchronisation utilisant ce formalisme rend cette tâche assez aisée bien qu'au bout du compte les modèles obtenus soient assez lourds et peu intuitifs comme nous le verrons.

Ces modèles visent à prouver les spécifications multimédia, c'est-à-dire à vérifier qu'elles sont cohérentes et exécutables. Ils présentent toutefois plu-

sieurs inconvénients. D'abord, ils sont abscons, et leur utilisation directe est difficile. Une couche d'adaptation vers un modèle de plus haut niveau est indispensable pour pouvoir les utiliser dans le cadre de présentations de grande taille. Ces modèles, utilisant l'approche par intervalles, présentent les avantages et les inconvénients déjà cités plus haut.

Hiérarchie d'opérateurs

La spécification de synchronisation par opérateurs utilise également une approche par intervalles. Le principe est de composer des intervalles à l'aide d'opérateurs. L'application d'un opérateur sur des intervalles produit un nouvel intervalle dont les caractéristiques dépendent des opérandes et du type d'opérateur appliqué. Un scénario s'exprime ainsi sous la forme d'une structure hiérarchique.

Le modèle le plus simple repose sur les deux opérateurs séquentiel et parallèle. Le premier spécifie la mise en séquence de plusieurs intervalles et le second leur exécution en parallèle. La figure 3.8 présente l'exemple déjà utilisé auparavant. D'autres modèles proposent une plus grande variété d'opérateurs, ce qui apporte plus d'expressivité, mais on retrouve toujours le schéma séquentiel/parallèle.

Ce type de spécification offre deux avantages majeurs. Le premier est la garantie de la cohérence des scénarios. Cette propriété découle du principe de construction qui exclut l'introduction d'incohérences puisque chaque opérateur produit un intervalle valide. Le second avantage est la modularité et la cohérence de la spécification. Il est possible de manipuler des sous-parties du scénario sans se préoccuper de son contenu et de son organisation. Elles apparaissent simplement comme des intervalles classiques, munis d'un début et d'une fin.

Toutefois, cette approche amène deux inconvénients importants. Le premier est le fait que, pour introduire une nouvelle relation entre deux objets, il est possible qu'il soit nécessaire de remanier la structure hiérarchique. Le second, plus important encore, est l'impossibilité de spécifier certains scénarios.

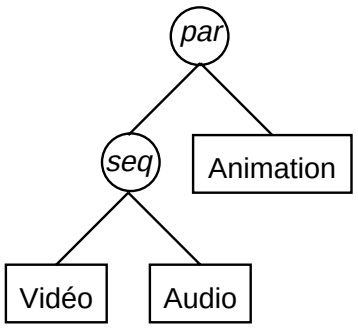


Figure 3.8: Hiérarchie séquentiel/parallèle

Scripts

La spécification à l'aide de scripts peut s'appuyer sur une ou plusieurs des méthodes déjà citées. Ils représentent souvent un moyen d'apporter de nouvelles fonctionnalités et un peu de souplesse à des modèles restreints comme le *time-line*. Dans ce cas, ils peuvent servir à traiter les interactions utilisateur ou programmer le comportement d'objets particuliers comme des boutons ou des choix à l'aide des structures conditionnelles classiques de la programmation. Ils permettent également d'associer des actions complexes décrites sous la forme d'un script à certains événements, qui lorsqu'ils se produisent, déclenchent l'exécution du traitement.

L'utilisation des scripts peut également se faire au sein d'un véritable environnement de programmation. Cela permet de développer des présentations sous la forme d'applications. Il devient alors possible de créer tous les comportements voulus, de contrôler tous les paramètres. Cependant, on s'éloigne du formalisme déclaratif qui est une constante de toutes les méthodes précédentes pour entrer dans le monde procédural et impératif. Malgré la grande souplesse que cela apporte, il en résulte une plus grande difficulté dans la création de présentations même basiques.

3.2.4 Cohérence et formatage temporels

La production de présentations multimédia est une tâche complexe car il faut représenter la dimension temporelle de façon concrète et statique, ce qui est déjà une gageure, mais il est également souhaitable de pouvoir s'assurer que la spécification produite pourra être respectée, c'est-à-dire vérifier sa

cohérence, et finalement produire le résultat attendu par *formatage* temporel.

Ces deux problèmes sont proches car ils consistent tous les deux à partir d'une spécification de scénario et, dans le premier cas, à prouver qu'au moins une solution existe, dans le second, à trouver une solution, optimale si possible. Ces problèmes ont été traités dans de nombreux travaux comme *Firefly*[9, 10] ou *Isis*[57] et dernièrement dans *Madeus* par Nabil Layaïda lors de son travail de thèse[60] de façon précise et en apportant des solutions très satisfaisantes.

Cohérence temporelle

Nous avons vu plus haut quelques exemples d'introduction d'incohérences dans un scénario. Les modèles *time-line* utilisant le temps absolu ne peuvent pas générer de scénarios incohérents puisque des dates précises sont assignées aux objets et qu'il n'y a aucune évaluation à faire, mais seulement à réaliser la présentation aux instants définis. Les incohérences peuvent surgir si l'on utilise la notion de temps relatif, qui permet par exemple de spécifier (A avant B) et (B avant A). Comme nous l'avons vu, l'utilisation d'opérateurs prémunit contre la spécification d'incohérences : un opérateur permet de définir un nouvel intervalle à partir de plusieurs autres, ces derniers ne seront plus alors manipulés directement, uniquement l'intervalle résultant pourra être utilisé par la suite.

On peut définir deux types d'incohérences[60] suivant leur nature qualitative ou quantitative. Dans le premier cas, l'incohérence relève des relations elles-mêmes, indépendamment des durées liées aux objets, comme dans l'exemple cité au paragraphe précédent. Dans le second cas, l'incohérence relève des valeurs de temps associées aux objets. La spécification est qualitativement correcte mais les durées s'opposent à son respect. L'approche choisie dans certains systèmes[9, 57, 36] est de modifier les durées des objets suivant certaines contraintes spécifiées par l'auteur afin de trouver une solution au problème. On parle alors de temps élastique ou *elastic time*.

La méthode la plus simple pour vérifier la cohérence est empirique et consiste à réaliser une évaluation de la présentation pour savoir si celle-ci est possible ou non. Il s'agit toutefois d'une méthode lourde qui impose de réaliser une présentation que l'on pourrait appeler fantôme puisqu'elle n'est pas présentée.

Une approche plus fiable est l'utilisation d'outils théoriques afin de vérifier la cohérence et éventuellement d'identifier les points d'incohérences s'il en existe ou de proposer une solution dans le cas contraire. De manière générale, lorsque des formalismes à base de relations sur les intervalles sont utilisés pour la spécification de scénarios, ces derniers sont toujours ramenés dans le formalisme par graphe d'instants. Cela permet d'utiliser des techniques de résolution de contraintes et de programmation linéaire déjà bien éprouvées.

L'inconvénient majeur de cette dernière approche est sa lourdeur. Les méthodes de vérification de graphes sont en général coûteuses et de complexité élevée, ce qui les rend lourdes à appliquer sur des présentations de grande taille. Il en résulte deux choses, lorsqu'elles sont utilisées dans les outils d'édition : soit la vérification est faite en fin d'édition ou lorsque l'auteur la demande, ce qui ne correspond pas à la nature incrémentale de l'édition et peut amener à détruire une présentation correcte sans savoir où et quand la ou les erreurs ont été commises, soit la vérification est faite à chaque ajout ou modification d'une relation ou d'une durée, ce qui rend alors le processus d'édition extrêmement fastidieux. Une solution est apportée dans *Madeus* qui propose un mécanisme de vérification de la cohérence qui est incrémental et localisé dans le graphe autour du point de modification, obtenant ainsi de bonnes performances y compris sur des graphes volumineux.

L'utilisation d'éléments à comportement indéterministe comme les flots en direct ou les interactions avec l'utilisateur posent toujours de sérieux problèmes dans le cadre de la vérification de la cohérence. En effet, s'ils ne gênent pas la vérification qualitative, aucune vérification quantitative ne peut être faite sachant que l'on ne connaît pas leur durée. Des mécanismes de vérification dynamique liés au formatage doivent alors être utilisés mais ils reposent souvent sur des configurations de scénarios particulières et ne traitent pas l'indéterminisme en général.

Formatage temporel

Le formatage temporel consiste à projeter une spécification temporelle sur le temps réel. Le formatage peut être statique, c'est-à-dire que des valeurs seront calculées et assignées statiquement avant présentation, ou dynamique, il se fera alors de façon continue au cours de la présentation. Certains systèmes comme *MODE*[7], *Firefly* et *Madeus* couplent les deux types de formatage. Le formatage dynamique est indispensable pour traiter l'indéterminisme.

Dans les systèmes à temps élastique comme *Firefly*, *Isis* ou *Xavier*[36] formatage et contrôle de cohérence sont conjoints. Le principe est d'essayer de construire une solution ou de déterminer l'ensemble des solutions en jouant sur les paramètres temporels des objets. À chaque objet est associé un triplet de valeurs représentant sa durée minimale, optimale et maximale. Une solution optimale est recherchée en maintenant chaque durée dans l'intervalle ainsi déterminé et le plus proche possible de la durée optimale. Dans *Firefly*, des coûts sont associés pour indiquer des préférences sur les valeurs.

3.3 Exemples de systèmes

Nous allons, dans les sections suivantes, décrire quelques systèmes ou modèles représentatifs. Précisons d'abord la terminologie. Nous classons les modèles en trois catégories, suivant qu'ils sont basés sur les *time-lines*, sur les instants ou sur les intervalles. Le fait de représenter les objets multimédia sous la forme d'intervalles ne suffit pas pour entrer dans la troisième catégorie, ce sont en fait la nature des relations temporelles qui sont prises en compte, à savoir les relations sur les instants ou les intervalles. Il est possible d'utiliser des relations sur les instants dans une représentation par intervalles en utilisant simplement les instants caractéristiques *début* et *fin* ou des instants internes définis d'une manière quelconque.

3.3.1 Modèles basés sur les axes temporels

Time-line simple

Ce modèle est utilisé par de nombreux systèmes, par exemple *QuickTime*, *Macromedia Director*, *Continuous Media Toolkit* (*CMT*)[96], *MAEstro*[24], et dans le standard ISO HyTime[48, 79].

La granularité de spécification de la synchronisation peut être extrêmement fine. Il est possible, à l'aide de certains systèmes d'édition comme *Director* de créer des animations ou des vidéos de toutes pièces en enchaînant une série d'images. De plus, des actions ou des effets peuvent être associés aux médias en certains points, permettant de réaliser des effets bien connus, comme le fondu en vidéo.

Fonction	Valeurs
Stop	$Vitesse = 0$
Avant	$Vitesse = 1$
Arrière	$Vitesse = -1$
Avance rapide $\times 2$	$Vitesse = 2$
Retour rapide $\times 3$	$Vitesse = -3$
Aller à lts	$Début = maintenant - lts$

Figure 3.9: Exemples de fonctions réalisées avec le Logical Time System

L'implémentation du *time-line* est aisée et les fonctions de contrôle facilement définies. La plupart de ces systèmes utilisent une représentation logique du temps définie ainsi : $LTS = Vitesse \times (TempsSystème - Début)$ où LTS signifie *logical time system*. Ainsi, en modifiant les deux paramètres $Vitesse$ et $Début$, on peut obtenir un ensemble de fonctions variées dont des exemples sont donnés en figure 3.9.

Time-line et script

Le modèle *time-line* seul est très restrictif dans les possibilités offertes à l'auteur de présentations multimédia. *Director* cité plus haut, qui est un système commercial largement utilisé aujourd'hui dans la création de présentations multimédia, offre la possibilité de programmer en utilisant le langage de script *Lingo* qui lui est associé. Cela permet, entre autres, de gérer plus efficacement les événements extérieurs liés aux interactions avec l'utilisateur.

Herlocker et Konstan ont proposé *TclStream*[40, 41], une extension au *Continuous Media Toolkit*. Il s'agit d'un nouveau type de média se présentant sous la forme d'un flot de commandes écrites dans le langage de script Tcl[82]. Cette extension permet de réaliser des interactions avec l'utilisateur au moyen de boutons déclarés en Tk[2] par exemple. Toutefois, le modèle de synchronisation de base reste celui de *CMT*, soit le time-line simple.

L'unité atomique dans *TclStream* est un *chunk* qui comprend cinq éléments. D'abord une valeur de temps logique, LTS (voir ci-dessus), exprimée en secondes. Celle-ci peut être absolue ou relative, ce dernier cas permettant la réutilisation de code déjà écrit dans un contexte différent sans avoir à modifier toutes les valeurs déjà calculées.

[2]Tk est un toolkit associé à Tcl permettant de créer des interfaces graphiques

Ensuite, il y a quatre fragments de code Tcl appelés respectivement *primary*, *rush-ahead*, *inverse* et *rush-behind*. Les deux premiers correspondent à la présentation normale, ($Vitesse > 0$) et les deux derniers à la présentation en sens inverse ($Vitesse < 0$). Le code *primary* est exécuté lorsque son LTS est atteint en marche avant et *inverse* lorsqu'il est atteint en marche arrière.

Les variantes *rush* sont les portions de code à exécuter à la place des *primary* et *inverse* lorsque l'exécution de ces dernières est annulée. La cause peut être double : il peut s'agir d'une resynchronisation du flot, ou simplement d'un saut dans le temps par modification de la variable *Début* (fonction Aller à). La variante *rush* adéquate des chunks qui ont été sautés doit être exécutée impérativement. Cela est utilisé pour maintenir les variables et les états dans une configuration cohérente lorsque du code qui aurait dû être exécuté ne l'est pas.

Time-line arborescent

Hirzalla[44] propose un modèle de time-line arborescent permettant de spécifier plusieurs alternatives dans un scénario. Un scénario est découpé en sous-parties indépendantes, chacune spécifiée par rapport à un axe temporel local. Il est possible de disposer sur l'axe des objets choix. Ceux-ci possèdent une certaine durée et référencent une autre sous-partie du scénario. Lorsqu'ils sont activés, la présentation de la partie courante est stoppée et celle de la partie référencée par le choix commence.

Des interactions utilisateur simples, se résumant au choix d'une alternative pendant le déroulement d'une présentation, sont alors possibles. La figure 3.10 montre un exemple de scénario. Pendant que la vidéo d'un voyage en Europe est présentée, des choix sont tour à tour autorisés en correspondance avec le pays dont il est question. L'utilisateur peut alors activer ces objets choix s'il désire lancer une sous-présentation décrivant plus précisément certaines grandes villes étapes de ce voyage.

Time-line multiple et contraintes

Nsync[5] est encore une autre variante proposée par Bailey et Konstan. Celle-ci repose également sur *CMT* pour fournir le contenu mais apporte une amélioration de taille en proposant la gestion d'interactions utilisateur

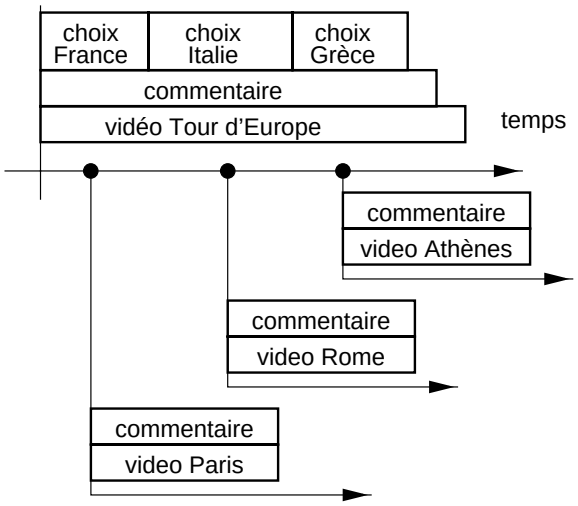


Figure 3.10: Exemple de spécification de type *time-line arborescent*

synchrones et asynchrones. Le modèle sous-jacent est *multi-time-line*. Chaque objet se voit assigner sa propre horloge et les relations de synchronisation entre ceux-ci sont définies dans un langage de spécification de haut niveau basé sur les contraintes temporelles.

Une couche de synchronisation a été ajoutée au-dessus de *CMT*[4]. La synchronisation est alors réalisée en dérivant des relations entre certaines des horloges des différents objets devant être présentés à un moment donné. Ceci permet de coordonner les objets de façon assez souple, de les présenter au moment voulu voire de ne pas les présenter du tout s'ils sont liés à un événement qui ne s'est pas produit.

Le langage décrit permet de spécifier la synchronisation en Tcl par une suite de contraintes temporelles de la forme `When {expression} {action}`. Cela signifie "lorsque `expression` devient vrai, exécute `action`". Elles sont ensuite compilées dans une représentation interne plus facile à manipuler.

Lors de l'exécution de la présentation, ces contraintes sont contrôlées pour déterminer si elles sont passées de faux à vrai pour alors réaliser l'action prévue. Comme elles peuvent contenir des contraintes temporelles sur les objets, ou plus exactement leur horloge, et que celle-ci évolue de façon continue, les contraintes peuvent devenir vraies à chaque instant. Il est impossible pratiquement d'évaluer ces contraintes de manière continue sans une perte importante de performance. Pour cette raison, l'évaluation des contraintes est faite en utilisant la *logique prédictive*. Cette logique comporte quatre va-

leurs :

- FALSE, la condition est fausse ;
- TRUE, la condition est vraie ;
- WBT(t), pour *will become true*, la condition est fausse mais deviendra vraie après un certain temps t.
- WBF(t), pour *will become false*, la condition est vraie mais deviendra fausse après un certain temps t.

On s'aperçoit en fait que TRUE = WBF(∞) et FALSE = WBT(∞). Les tables de vérité pour le OU et le ET peuvent être calculées facilement. Par exemple $\text{WBT}(t_1) \vee \text{WBT}(t_2) = \text{WBT}(\min(t_1, t_2))$.

À l'issue du calcul des valeurs des contraintes, les actions appropriées sont exécutées. Si la valeur est fausse, rien n'est fait. Si elle est vraie, l'action correspondante est exécutée. Si elle vaut WBT(t), l'action est programmée pour s'exécuter après un temps t. Si elle vaut WBF(t), la valeur sera basculée à faux lorsque le temps t sera écoulé autorisant ainsi un passage ultérieur à vrai. En cas de modification de certains paramètres des contraintes, leurs actions en attente sur des WBT(t) et WBF(t) sont annulées et les contraintes réévaluées.

3.3.2 Modèles basés sur les instants

Les modèles basés sur les instants permettent de spécifier la synchronisation par l'intermédiaire de relations entre des points temporels des différents objets. Cet ensemble de relations constitue alors un graphe d'instants.

Dans *MODE*[7] (Multimedia Objects in a Distributed Environment), proposé par Blakowski. L'élément de base de la spécification de la synchronisation est le point de synchronisation, *synchronization point*. Il s'agit d'un point de rendez-vous défini entre plusieurs objets et où chacun attendra tous les autres avant de continuer son exécution. Ce mécanisme permet de supporter les objets à durée inconnue comme les objets interactifs. Les points de synchronisation sont positionnés entre des points de référence, *reference points*, définis sur les différents objets. Un objet peut accélérer son exécution ou directement sauter au point de synchronisation suivant pour libérer d'autres

objets en attente. Dans ce système, le formatage de la spécification temporelle permettant de déterminer les actions à réaliser pour respecter celle-ci est réalisé en deux phases, une première statique qui génère un *flow graph* et une seconde dynamique réalisée lors de l'exécution de la présentation en utilisant ce graphe.

Firefly[9] est un environnement multimédia proposé par Buchanan et Zellweger permettant de générer automatiquement un ordonnancement pour des spécifications de présentations multimédia[10]. À chaque élément constituant un média sont associés des événements. Des triplets de durées minimale, optimale et maximale sont associés aux couples d'événements consécutifs d'un même média, ainsi que des coûts exprimant une préférence dans le choix de ces durées. Ces durées peuvent être spécifiées comme imprédictibles, ce qui permet d'accepter les objets au comportement indéterministe. La spécification de la synchronisation se fait alors en exprimant des contraintes temporelles de deux sortes entre un événement source et un événement destination :

- des égalités, spécifiant que deux événements doivent être simultanés ou que l'événement source doit se produire un temps donné avant l'événement destination ;
- des inégalités, spécifiant un ordre de précédence éventuellement accompagné de durées minimale et maximale.

La figure 3.11 montre un exemple de spécification. Le formatage temporel est réalisé en deux phases, comme dans *MODE*, en utilisant des techniques de programmation linéaire : une première phase de compilation de la spécification, puis une seconde phase dynamique lors de la présentation dont le fonctionnement est semblable au formatage spatial réalisé par le formateur de documents TEX[58].

Le système *FLIPS*[98], pour FLexible Interactive Presentation Synchronization, utilise un modèle basé sur les notions de barrière, *barrier*, et déclencheur, *enabler*, qui lui permet de supporter l'indéterminisme. Ce modèle utilise des objets multimédia et des événements liés à ces objets comme le début et la fin. L'événement e_i est une barrière à e_j, noté $e_i \dashv e_j$, signifie que e_j ne peut advenir qu'une fois e_i passé. À l'inverse, l'événement e_i est un déclencheur de e_j, noté $e_i \rightarrow e_j$, signifie que lorsque e_i a lieu il provoque e_j.

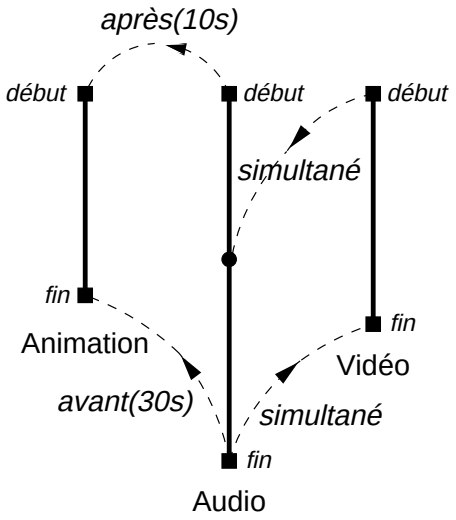


Figure 3.11: Exemple de spécification avec *Firefly*

Ces deux relations permettent de définir des comportements variés lorsqu'ils sont associés. La figure 3.12 montre plusieurs cas en exemple. Les extrémités des rectangles représentant les objets sont les événements de début et de fin. L'image et l'audio 1 s'exécutent en parallèle et le premier terminé arrête l'autre. Ensuite, l'audio 2 se comporte en maître sur la vidéo 2, la fin de cette dernière étant conditionnée par la fin de l'audio. Finalement, l'audio 3 et la vidéo 3 s'attendent mutuellement pour terminer la présentation. On retrouve ici des relations classiques sur les intervalles que nous verrons plus loin, respectivement par-min, finishes et par-max.

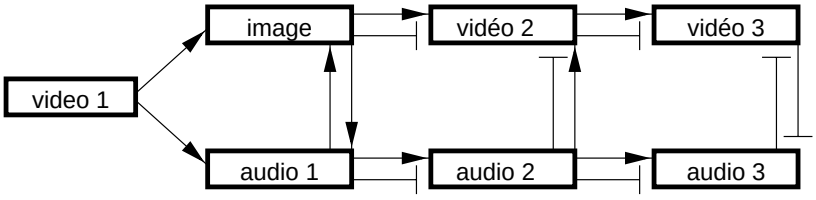


Figure 3.12: Exemple de spécification avec *FLIPS*

Mediadoc[54] basé sur le standard ISO ODA, Office Document Architecture, utilise les trois relations citées auparavant, avant, simultané et après, auxquelles un délai peut être associé. La spécification temporelle n'est pas toutefois le but principal de *Mediadoc* qui est surtout orienté vers la structuration logique et spatiale des documents.

3.3.3 Modèles basés sur les intervalles

Réseaux de Petri

Little et Ghafoor ont proposé un modèle basé sur un dérivé des réseaux de Petri temporisés, *OCPN*[66], Object Composition Petri Nets. Dans ce cas, les places représentent des objets auxquels une durée est associée et les transitions synchronisent les instants de début et de fin. La figure 3.13 montre des exemples de structure, respectivement équivalentes à (A meets B), (A finishes B), (A during B). La figure 3.14 présente un exemple de spécification de scénario. Par la suite[67] des extensions ont été proposées pour permettre un contrôle d'accès temporel efficace et couplé au stockage des données. Ce modèle ne prend pas en compte les comportements indéterministes induits par des interactions avec l'utilisateur par exemple.

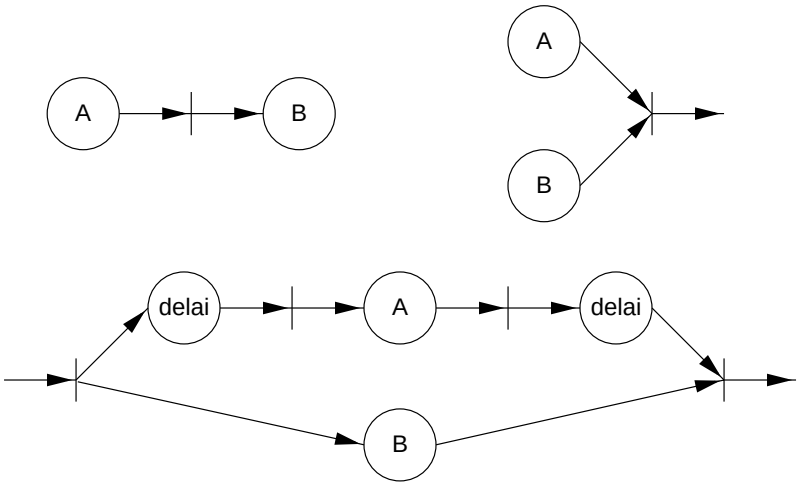


Figure 3.13: Exemple de structures *OCPN*

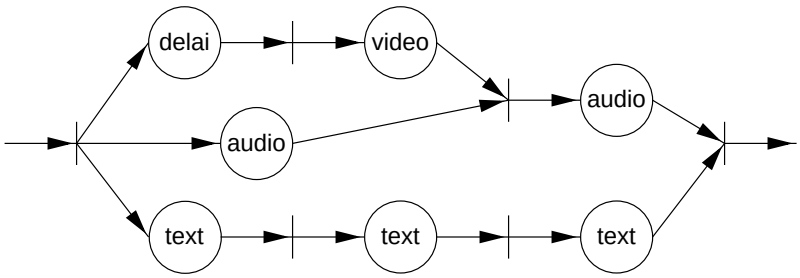


Figure 3.14: Exemple de spécification *OCPN*

Le modèle *Trellis*[106] également basé sur les TPN prend en compte la structure hypermédia des présentations et traite cet aspect d'indétermination.

Prabhakaran et Raghavan proposent une extension de OCPN en utilisant les réseaux de Petri temporisés dynamiques, *DTPN*[86], qui permettent à l'utilisateur d'interagir avec la présentation de manière asynchrone. Le principe consiste à préempter l'exécution du réseau de Petri pour modifier les valeurs temporelles associées à celui-ci.

Une solution au problème posé par les objets dont les durées ne sont pas connues a priori a été proposée[63], mais celle-ci envisage que ces durées seront disponibles au moment où la présentation sera effectuée. Cela apporte un peu de flexibilité, mais ne résout pas l'indéterminisme dans le cas général, les durées étant connues une fois l'objet en question arrivé à sa fin.

Dans le cadre de la conception de *MediaWare*[1], un environnement multimédia distribué, un autre modèle basé sur les réseaux de Petri interopérables (*IPN, Interoperable Petri Nets*) a été proposé, ceux-ci étant une extension des réseaux de Petri de haut niveau (*HLPN, High Level Petri Nets*). Les HLPN associent aux places, aux transitions et aux jetons une couleur. Le jeton ne peut résider ou transiter que par les places et les transitions ayant la même couleur que la sienne. Les IPN sont utilisés dans *MediaWare* pour formaliser la synchronisation intra-média et inter-média ainsi que les agents entrant dans le processus de synchronisation des objets multimédia distribués.

Pour conclure, nous pouvons dire que l'utilisation des réseaux de Petri n'est pas aisée et produit des spécifications peu claires. De plus, les interactions utilisateur, lorsqu'elles sont traitées, complexifient encore les modèles. Aucune solution praticable n'est apportée à l'indéterminisme pour ces modèles.

Relations

À partir d'une analyse systématique de l'approche par instants et par intervalles, Wahl et Rothermel ont proposé un modèle reposant sur les intervalles[114]. L'idée est de supprimer l'ambiguïté inhérente à certaines des relations d'Allen. Par exemple, la relation before ne spécifie pas de délai permettant de positionner les deux intervalles l'un par rapport à l'autre. Pour cela, dix nouvelles relations ayant en paramètre un ou plusieurs délais positifs ou nuls sont définies : before(δ_1), cobegin(δ_1), coend(δ_1), beforeendof(δ_1),

while(δ_1, δ_2), cross(δ_1, δ_2), delayed(δ_1, δ_2), startin(δ_1, δ_2), endin(δ_1, δ_2) et overlaps$(\delta_1, \delta_2, \delta_3)$. Quelques exemples sont présentés en figure 3.15.

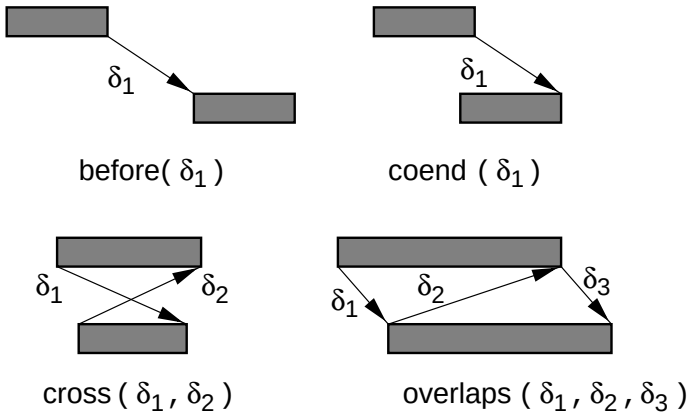


Figure 3.15: Exemples de relations de Wahl et Rothermel

Les auteurs prévoient de spécifier des délais indéterminés en leur attribuant la valeur $*$ s'ils sont positifs ou nuls, ou $+$ s'ils sont strictement positifs. On peut ainsi retrouver toutes les relations d'Allen. Par exemple before = before$(+)$ ou meets = before(0). Le travail exposé dans cet article n'aborde que le modèle temporel et ne propose pas de solutions algorithmique et pratique nécessaires à son utilisation.

Isis[57] est un système multimédia similaire à *Firefly* dans la résolution des contraintes temporelles, mais qui diffère en deux points. D'une part, il utilise une approche par intervalles, en fournissant quatre relations de spécification temporelle : co-start$(Obj_1, \ldots, Obj_m)$, co-end$(Obj_1, \ldots, Obj_m)$, meet$(Obj_1, \ldots, Obj_m)$ et co-occur(Obj_1, Obj_2) (cette dernière relation est en fait équivalente à co-start(Obj_1, Obj_2) et co-end(Obj_1, Obj_2)). D'autre part, il ne se borne pas à offrir une solution optimale au système de contraintes, mais propose à l'auteur un ensemble de solutions possibles pour lui permettre de raffiner son scénario, en choisissant lui-même la durée totale de présentation par exemple.

Madeus[52, 51, 61, 59, 60] est un système d'édition et de présentation de documents multimédia qui a été spécifiquement développé dans le but de fournir un support à l'édition incrémentale de documents multimédia cohérents. Ce système manipule des documents temporels structurés associés à des jeux de contraintes temporelles. Ces contraintes temporelles sont spécifiées sous la forme de relations sur les intervalles. Les relations utilisées sont les sept relations d'Allen plus trois relations causales illustrées sur la figure 3.16 : *parmin*,

parmax et *parmaster*. *Parmaster* définit un des arguments comme maître, ce dernier contrôlant alors la relation, celle-ci étant équivalente à *parmin* si le maître se termine en premier, à *parmax* s'il se termine en dernier.

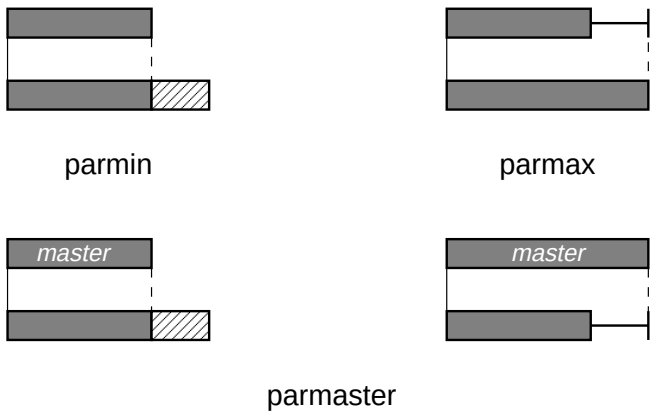


Figure 3.16: Relations causales dans *Madeus*

La spécification temporelle est traduite dans une représentation interne basée sur les instants. Le système de contraintes associé à un document évolue au fur et à mesure du processus d'édition : de nouvelles contraintes sont ajoutées, d'autre supprimées ou modifiées. Sa cohérence est également maintenue, ce qui assure qu'une solution au système permettant la présentation du document existe. Ajouté à cela, un système de présentation prend en charge le traitement dynamique du document nécessaire à sa restitution dans des conditions optimales.

Une contribution originale a été apportée afin d'intégrer les présentations multimédia aux bases de données[75]. Ce travail est basé sur un système de gestion de bases de données à objets. Les relations temporelles utilisées sont des variantes des relations séquentiel et parallèle sur les intervalles. Les concepts d'ombre temporelle et d'ombre comportementale sont utilisés. La première définit le domaine de validité temporelle d'un objet. La seconde spécifie le comportement d'un objet au moyen de triplets événement, intervalle de validité, liste d'actions. L'intervalle de validité représente la période durant laquelle l'événement est valide. Si celui-ci se produit alors la liste d'actions est exécutée. Ainsi des présentations interactives peuvent être spécifiées. Un SGBD multimédia a été conçu pour prendre en compte, outre les médias temporels de base, les scénarios définis à partir de ceux-ci.

Opérateurs

Algebraic Video[116] ou vidéo algébrique est un modèle qui aborde le problème de la composition temporelle sous un aspect algébrique formel. Les objets de base sont des segments vidéo que l'on compose de façon incrémentale en utilisant des opérateurs. L'avantage majeur de cette approche est que par opposition aux systèmes précédents avec lesquels la structure de composition temporelle est un graphe dont la cohérence n'est pas garantie, ici on construit un arbre dont les nœuds sont des opérateurs et dont la cohérence est assurée par le principe de construction lui-même. Toutefois certaines configurations de scénarios ne sont pas réalisables.

En outre, ce modèle offre des opérateurs non temporels permettant une manipulation complexe des séquences vidéo ainsi que l'adjonction d'information utilisée dans le cadre de l'accès par le contenu aux objets multimédia complexes ainsi construits.

Menkalinan[25, 56, 55] généralise cette approche aux présentations multimédia et prend également en compte les aspects indéterministes et de choix proposé à l'utilisateur. Ce modèle s'appuie tout particulièrement sur la causalité dans les présentations multimédia. Les opérateurs ne décrivent pas des états, comme par exemple A et B sont en séquence, mais plutôt les causes de cet état : quand A se termine B démarre ou quand B démarre il termine A. La notion d'activation externe apparaît donc directement dans certains opérateurs. La figure 3.17 montre huit opérateurs temporels. S'ajoutent à ceux-ci, deux autres opérateurs, *alternative* et *loop*. Le premier associe des couples d'intervalles (a_i, b_i) où le premier des a_i se terminant, a_k, provoque le départ de b_k et termine tous les $a_i (i \neq k)$. Le second s'exécute en boucle.

3.3.4 Modèles hybrides

Le système *Xavier*, au-dessus duquel une application d'édition multimédia appelée *Mbuild*[36] a été construite, utilise le concept de *temporal glue* qui est une adaptation du concept de *glue* utilisé dans TEX[58]. Les auteurs utilisent une approche objet avec laquelle la définition d'une présentation se fait par composition successive d'objets. Chaque objet possède des propriétés de temporal glue, contrôlant son extension (*stretchability*) et sa réduction (*shrinkability*). Un objet glue existe également, ne contenant aucun média et dont le seul but est d'ajouter de la souplesse dans le scénario. La compo-

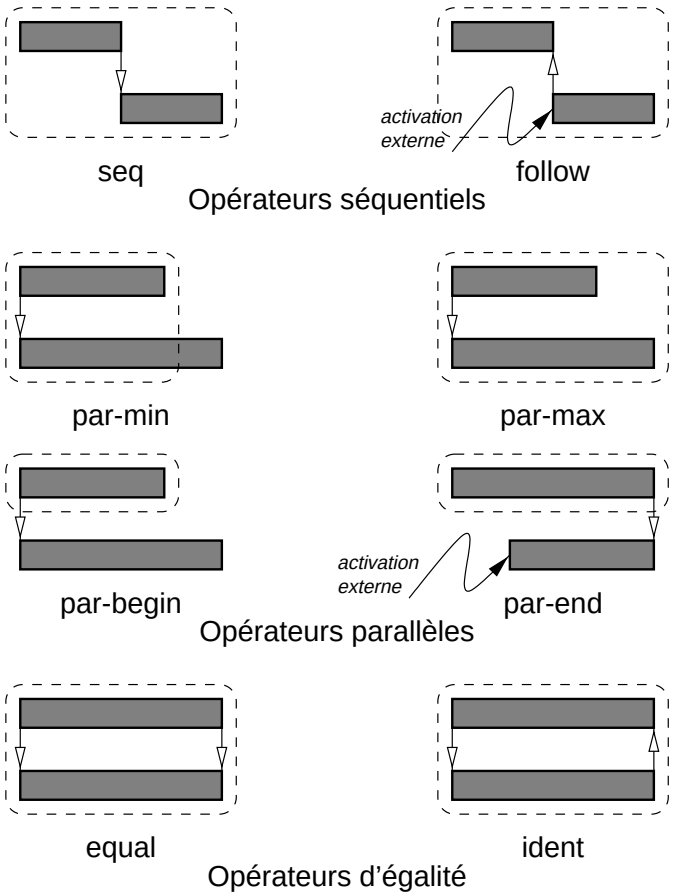


Figure 3.17: Opérateurs de Menkalinan

sition des objets de base se fait en utilisant des opérateurs de composition d'objets. Deux opérateurs permettent de définir les arrangements séquentiel et parallèle : SEBox($Obj_1, Obj_2, \ldots, Obj_n$) pour la séquentialité (*Start-End*) et Overlay($Obj_1, Obj_2, \ldots, Obj_n$) pour le parallélisme entre les objets. Il est également possible de spécifier des contraintes entre des points de référence avec Constraint(*condition*), qui sont du style Constraint($Obj_1.start = Obj_2.mark$). Enfin Position($Obj, StartTime, EndTime$) sert à positionner de façon absolue un objet. Ce modèle mélange donc respectivement les caractéristiques des modèles à intervalles, à instants et time-line. Un formateur temporel statique se charge de déterminer les durées à associer aux objets en fonction des paramètres de glue afin que la présentation soit cohérente. Les durées devant être connues à l'avance, ce modèle ne supporte pas l'indéterminisme.

CMIFed[109] est, à l'instar de *Madeus*, un outil d'édition et de présentation multimédia basé sur la structure. Il utilise le format CMIF[14], CWI Multimedia Interchange Format. La composition temporelle est réalisée en utilisant deux moyens. D'une part, en spécifiant des relations n-aires sur les intervalles représentés par les médias, qui sont *séquentiel* et *parallèle*. Elles correspondent respectivement au *before* et au *during* d'Allen. D'autre part, en utilisant des arcs de synchronisation, *synchronization arcs* : ceux-ci spécifient une relation d'ordre binaire entre leur origine et leur destination, qui sont des instants caractéristiques des deux objets mis en relation.

Une présentation est une structure arborescente dont les feuilles sont les objets multimédia de base et les autres nœuds des objets composites offrant le choix entre une présentation séquentielle ou parallèle des nœuds fils.

L'éditeur offre deux angles de vue différents sur cette structure. Le premier est la vue hiérarchique qui expose cette arborescence du point de vue d'un nœud. Il est possible de la parcourir en descendant dans les fils ou en remontant vers le père. Le second est la vue par canal. À chaque média est associé un canal sur lequel les données qui lui correspondent seront présentées, une portion d'écran, un canal audio, etc. Ces canaux sont logiques et ne correspondent pas nécessairement à un canal physique de la machine, ce dernier pouvant rassembler plusieurs canaux logiques. Tous les canaux ne sont pas nécessairement présentés, par exemple dans le cas où un choix est offert entre plusieurs commentaires dans des langues différentes. Cette vue par canal dispose les objets sur la ligne temporelle correspondant au canal auquel il est associé. Leur position et leurs relations temporelles sont déduites de la structure hiérarchique. Une distinction est faite entre les objets pour lesquels une durée est définie et ceux pour lesquels elle est déduite de la structure. Des arcs de synchronisation, relations sur les instants, peuvent être spécifiés entre les objets des différents canaux pour préciser les relations de la vue hiérarchique.

Finalement, afin d'être présentée, cette représentation est traduite en un graphe de dépendances temporelles sur les instants. Ce graphe est d'abord utilisé pour résoudre les incohérences de la spécification et les conflits de ressources qui peuvent survenir entre les canaux. La présentation est réalisée en parcourant ce graphe. Ce modèle hybride perd l'intérêt majeur de la construction hiérarchique qui garantit la cohérence, puisque comme nous l'avons dit précédemment, l'utilisation d'arcs de synchronisation permet la spécification de scénarios incohérents.

3.4 Standards ou apparentés

Plusieurs efforts de standardisation ont eu lieu dans le domaine du multimédia sans réel succès jusqu'à aujourd'hui puisqu'aucun outil de création multimédia diffusé à grande échelle ne les utilise. La tendance semble s'inverser avec l'apparition de SMIL qui n'est pas un standard à proprement parler mais une recommandation du consortium World Wide Web. Cela pourrait s'expliquer parce qu'il a été développé récemment par des personnes travaillant sur le multimédia depuis déjà quelques années et rapidement mis en œuvre, contrairement aux standards dont les spécifications mettent des années à être finalisées.

3.4.1 HyTime

HyTime[48, 79], Hypermedia/Time-based Structuring Language, est un standard ISO de représentation des documents hypermédia structurés. C'est une application de SGML, Standardized General Markup Language. SGML est un autre standard ISO destiné à l'échange de documents structurés dont le principe consiste à spécifier la structure d'un document à l'aide de balises dont la définition et l'interprétation sont fournies par une DTD, *Document Type Definition*.

HyTime permet de définir des DTD dans le but de décrire des documents hypermédia structurés en spécifiant les relations entre les éléments qui les composent. Il ne concerne pas la description de format et de codage. HyTime définit des éléments multidimensionnels dont les systèmes de coordonnées sont définis par l'utilisateur. Des dépendances spatiales et temporelles peuvent ainsi être spécifiées entre les objets multimédia. Il est également possible d'adresser des parties d'éléments et d'établir des liens entre des parties d'éléments.

La création de documents hypermédia en HyTime se fait en définissant une DTD HyTime dont les éléments sont associés à une forme architecturale, *architectural form*. Celles-ci sont regroupées au sein des six modules suivants.

- Le *module de base* spécifie la forme architecturale comprenant le document.
- Le *module des dimensions* permet de spécifier tout ce qui a trait aux

mesures et ainsi de définir des systèmes de coordonnées dans lesquels seront placés les éléments du document.

- Le *module de localisation* offre des méthodes d'adressage au sein du document suivant des méthodes plus complexes que celle fournie par SGML. Ainsi, il est possible de repérer des éléments par nom, par coordonnées ou d'une manière sémantique permettant ainsi l'accès par le contenu.

- Le *module d'ordonnancement* place les objets multimédia dans un système de coordonnées (FCS, Finite Coordinate Space) constitué de plusieurs axes définis par l'application. Des événements attachés aux objets sont placés sur ces axes de façon absolue ou relativement à d'autres événements.

- Le *module de liaison hypermédia* permet de définir des hyperliens entre les objets en utilisant la localisation.

- Le *module de présentation* spécifie la transformation à appliquer pour passer d'un FCS à un autre. Il permet de passer des FCS du document aux dimensions réelles.

Afin de présenter un document HyTime, un *HyTime engine* doit transformer sa spécification en identifiant les formes architecturales et en effectuant les tâches correspondantes décrites plus haut, qui sont indépendantes de l'application et passer le résultat de ce travail à l'application pour la présentation. Un tel système couplé à une base de donnée appelé *HyOctane*[12] a été développé.

HyTime veut couvrir de façon presque exhaustive l'édition de documents hypermédia. Il en résulte un standard très lourd qu'il est pratiquement impossible de mettre en œuvre. La création d'une présentation résulte alors en sa spécification et dans la réalisation d'une application capable de la restituer. Cela représente un travail énorme absolument pas du ressort d'un auteur. De plus la plus petite spécification temporelle devient excessivement verbeuse et difficilement compréhensible.

Une utilisation de HyTime plus rationnelle a été proposée[97], supprimant l'aspect présentation mais non édition des documents. Il s'agit d'utiliser HyTime pour le stockage de documents hypermédia sachant qu'il spécifie des documents indépendants de toute forme de présentation. Il est ensuite possible en utilisant le standard ISO DSSSL (Document Style Semantics and

Specification Language) de définir une transformation d'une classe de documents HyTime définis par une DTD donnée dans un autre format. Il est par exemple possible de générer des documents SMIL qui peuvent être alors présentés par une application conforme à cette recommandation.

3.4.2 MHEG

MHEG[87, 27], Multimedia and Hypermedia Information Coding Expert Group, est un autre standard défini par l'ISO. C'est le compagnon de deux autres standards bien connus, JPEG et MPEG qui concernent le codage des données, respectivement les images et l'audio et la vidéo. MHEG s'intéresse à l'échange des données multimédia et hypermédia et est basé sur une approche orientée objet. Ce standard est surtout destiné à être utilisé dans la réalisation de programmes multimédia interactifs pour la télévision et dans cette optique intègre dans sa conception le fait que les systèmes de présentation pourront être très frustes.

MHEG définit une hiérarchie de classes de base. Celles-ci implémentent plusieurs concepts nécessaires à la définition de présentations. Du contenu peut être référencé ou inclus, texte, audio, vidéo, etc. Un système de coordonnées à quatre dimensions, espace plus temps, est défini pour positionner le contenu. Des comportements permettent de spécifier la présentation du contenu par des actions qui agissent sur l'état des objets, comme démarre et arrête, ainsi que la coordination des divers objets au sein de la présentation par l'utilisation de liens définissant des dépendances spatiales, temporelles et conditionnelles. Les interactions utilisateur sont de deux sortes, soit sélection permettant d'interagir sur le flot de présentation par l'utilisation de boutons par exemple, soit modification permettant de saisir des données. Enfin, les objets de ces trois catégories peuvent être structurés en objets composites définissant des présentations.

La spécification de la synchronisation dans MHEG est de type axe temporel et repose sur les objets action. Des listes d'actions sont définies qui s'exécutent, soit en parallèle, soit en séquence, avec la possibilité d'ajouter des délais. De plus, l'utilisation de liens conditionnels permet d'exécuter des actions conditionnées à des événements.

Afin d'exécuter des présentations MHEG, il faut disposer d'un *MHEG engine* réalisant les fonctionnalités définies par la hiérarchie de classes et interfaçant celles-ci avec les systèmes nécessaires à la présentation du contenu

comme les décodeurs audio et vidéo.

3.4.3 PREMO

PREMO[42, 43], Presentation Environments for Multimedia Objects, est également un standard développé par l'ISO, mais qui se démarque des deux précédents parce qu'il s'intéresse aux techniques de présentation multimédia, un point ignoré par HyTime et MHEG. Une autre différence vient du fait que PREMO considère les applications multimédia dans leur ensemble et ne privilégie pas l'approche document. Cela ouvre alors le champ à de nombreuses applications, notamment celles dont les données sont générées en temps réel comme l'animation, la réalité virtuelle et la visualisation scientifique.

PREMO utilise un modèle objet basé sur les objets actifs. Les opérations réalisées par ces objets sont de trois types : synchrones, asynchrones ou échantillonnées. Ce dernier cas est asynchrone, mais, si la précédente requête n'a pas été traitée lorsqu'une nouvelle arrive, cette dernière remplace la précédente.

L'interactivité et la synchronisation[26] dans PREMO reposent sur les événements et les gestionnaires d'événements (`Event Handler`). Un événement représente une action qui se produit à un instant déterminé. Il est typé, créé par une source et consommé par un client. Les clients potentiels se signalent auprès d'un gestionnaire d'événements en spécifiant des contraintes sur les événements qu'ils désirent recevoir. Les sources envoient leurs événements aux gestionnaires qui les répartissent aux clients intéressés.

Chaque objet progresse de façon autonome dans le temps. Des points de référence peuvent être définis sur ces objets et des événements leur être associés. Lorsqu'un point de référence est rencontré, les événements qui lui sont associés sont envoyés. Pour les objets intrinsèquement temporels comme l'audio et la vidéo, PREMO offre, par héritage multiple, une synchronisation directe pilotée par une horloge et un facteur vitesse.

L'approche choisie dans la spécification de PREMO est pratique et de bas niveau. Des réponses concrètes sont apportées à l'implémentation d'application multimédia, mais ce standard n'aborde pas le problème de la spécification de présentations à un haut niveau. Il s'agit purement d'un environnement d'exécution et non d'édition.

3.4.4 SMIL

SMIL[113], Synchronized Multimedia Integration Language, prononcé "smile" à l'anglaise, est un langage de description de documents multimédia basé sur XML. XML, Extensible Markup Language, est un format de données destiné à l'échange de documents structurés sur le Web. C'est pour ainsi dire une version allégée de SGML.

La spécification d'un document SMIL est une description textuelle structurée à l'aide de balises de marquage ou *tags*, similaire à HTML, définissant la disposition spatiale et temporelle d'éléments multimédia classiques, texte, audio, vidéo, etc. Ces éléments sont inclus par référence à l'aide d'URI, Uniform Resource Identifiers. Les URI font actuellement l'objet de travaux et sont amenés à remplacer les actuels URL, Uniform Resource Locators[6, 31].

Un document SMIL est constitué de deux parties, une en-tête et un corps. L'en-tête est délimitée par les tags `<head> </head>` et contient, entre autres, la définition de la disposition spatiale. Celle-ci se fait en spécifiant un *layout* constitué de zones graphiques rectangulaires en utilisant un langage de définition. Plusieurs layouts peuvent être définis dans des langages différents en utilisant le tag `<switch>`. SMIL fournit un langage appelé SMIL Basic Layout Language, mais par exemple il est possible d'utiliser CCS, Cascading Style Sheets[111]. La figure 3.18 montre un exemple de spécification de layout, le premier utilise CSS et le second le langage de base fourni par SMIL.

La seconde partie d'un document SMIL est le corps délimité par `<body> </body>` et contenant la déclaration des objets ainsi que leur composition temporelle. Un identificateur peut être associé à chaque élément avec l'attribut `id`. Les éléments de synchronisation constituent la catégorie d'éléments la plus importante. Elle est composée des éléments média : `ref`, `animation`, `audio`, `img`, `video`, `text` et `textstream`. `ref` représente un élément générique, la nature des autres étant évidente en regard de leur nom. Il existe deux autres éléments de synchronisation permettant de spécifier la composition temporelle, `par` et `seq`. Ce sont les deux opérateurs parallèle et séquentiel vus en 3.2.3. L'élément parallèle possède un attribut `end-sync` afin de lever l'ambiguïté sur sa terminaison. Il peut prendre trois valeurs `last`, `first` ou un identificateur d'élément, correspondant ainsi au *parmax*, *parmin* et *parmaster*[3] vus en 3.16.

[3]Le maître étant l'élément référencé.

```
<smil>
  <head>
    <switch>
      <layout type="text/css">
        [region="r"]  top: 20px; left: 20px
      </layout>
      <layout>
        <region id="r" top="20" left="20" />
      </layout>
    </switch>
  </head>
  <body>
    <seq>
      <img region="r" src="http://www.w3.org/test" dur="10s" />
    </seq>
  </body>
</smil>
```

Figure 3.18: Layout dans SMIL

Chaque élément de synchronisation peut se voir doté d'attributs `begin`, `end` et `dur` définissant respectivement son début, sa fin et sa durée. Ces attributs permettent de définir des valeurs explicites. S'ils sont absents, des valeurs implicites sont déterminées à partir de la hiérarchie `par/seq`. Les valeurs explicites sont spécifiées à partir d'événements, `begin`, `end` ou une valeur de temps, et d'éléments de synchronisation référencés par leur identificateur. Par exemple, `begin="id(v)(end)"` signifiant que le début se produira en même temps que la fin de `v` et `end="id(a)(20s)"` signifiant que la fin aura lieu 20 secondes après le début de `a`.

`switch` est un élément permettant de spécifier des alternatives en fonction d'attributs de test comme la bande passante ou la langue de l'utilisateur. Enfin, les attributs `a` et `anchor` assurent la création de liens. `a` fonctionne pratiquement comme dans HTML. `anchor` est plus tourné vers l'hypermédia en permettant de référencer des portions spatiales ou temporelles.

La figure 3.19 montre quelques exemples de constructions SMIL présentées ci-dessus.

Le modèle temporel utilisé dans SMIL est un modèle hybride semblable à celui utilisé dans CMIF vu en 3.3.4, mélangeant les abstractions par in-

```
<smil>
  <head> ... </head>
  <body>
    <par>
      <seq>
        <switch>
          <audio src="speech_fr.wav" system-language="fr"/>
          <audio src="speech_en.wav"/>
        </switch>
        <audio id="bkgd" begin="5s" src="music.mid"/>
      <seq>
      <video src="intro.mpg" end="id(bkgd)(20s)" region="reg_intro"/>
        <anchor href="suite" begin="0s" end="10s"/>
      </video>
      <video src="clip.mpg" region="reg_clip">
        <anchor id="milieu" begin="45s">
      </video>
    </par>
  </body>
</smil>
```

Figure 3.19: Exemples de constructions SMIL

tervalles et instants. La structure de document et la structure temporelle sont confondues et représentées par la hiérarchie d'opérateurs séquentiel et parallèle. Deux nouveaux inconvénients viennent s'ajouter à ceux cités précédemment. D'abord, la nature du couplage entre les éléments de l'opérateur parallèle n'est pas définie par SMIL et laissée à l'implémentation ce qui enlève le contrôle sur la nature de la synchronisation souhaitée. Enfin, l'utilisation de l'élément `switch` ne fait qu'alourdir des documents multimédia dont on peut supposer qu'ils seront assez volumineux puisqu'utilisant un grand nombre et une grande variété d'objets. Par exemple, il parait impensable, qu'à chaque fois que du texte ou de la voix est inclus dans un document, de spécifier toutes les variantes possibles, ne serai-ce que dans les langages les plus usités sur le Web, ou proposer chaque vidéo en six débits différents pour s'adapter aux largeurs de bande les plus courantes.

3.5 Conclusion

Nous avons essayé de présenter, aussi fidèlement que possible dans ce chapitre, les activités de recherche dans le domaine des présentations multimédia. Nous n'avons pas pu ni voulu être exhaustifs sur les travaux en cours ou réalisés par le passé tant ils sont nombreux, mais nous avons plutôt essayé de présenter une vision globale assez conforme à la réalité. Il ressort de cette analyse que la modélisation de la synchronisation, et plus précisément la spécification de la composition temporelle, constituent les centres de recherche principaux.

Cependant, malgré toutes les propositions dont nous avons fait état, nous devons constater que leur impact sur le domaine applicatif reste limité en dehors des quelques prototypes évoqués. Les outils de création multimédia les plus largement utilisés aujourd'hui restent principalement basés sur une approche *timeline* ou impérative (programmation par *scripts*), et la modélisation multimédia n'a pas réussi à se trouver réellement une application phare.

Lorsque ce travail de thèse a débuté, le Web était déjà un des vecteurs de publication majeurs pour le contenu numérique. Pour cette raison, nous avons choisi le World Wide Web comme support aux propositions que nous avons faites et qui vont être exposées par la suite. Le succès que semble rencontrer SMIL est dû en partie à ce même choix. Les éditeurs de logiciels d'édition et de présentation multimédia s'orientent aujourd'hui vers cette

recommandation du consortium W3, bien qu'elle n'en soit encore qu'à ses débuts.

Nous pouvons faire deux observations supplémentaires qui sont également importantes dans les choix que nous avons faits. Nous avons déjà fait remarquer plus haut que la spécification de la composition temporelle est le point sur lequel porte la majorité des travaux. À l'inverse, la problématique liée au couplage inter-média est généralement laissée de côté. Lorsqu'elle est abordée, c'est principalement du point de vue de l'exécution de présentations, elle est presque complètement ignorée pour ce qui est de la spécification.

La seconde observation concerne les modèles temporels utilisés. Ceux utilisés pour la spécification de présentations dans les travaux qui ont été présentés sont très variés, cependant, on constate qu'ils sont, dans de nombreux cas, ramenés au modèle à graphe d'instants, dès lors que des traitements de vérification ou de présentation doivent être appliqués.

Chapitre 4

Documents multimédia pour le Web

4.1 Contexte

4.1.1 Multimédia sur le Web

Comme cela a déjà été dit en 2.1.2, le World Wide Web constitue certainement aujourd'hui le plus grand vecteur de publication, et ce pour deux raisons principales :

- il est désormais pratiquement accessible à tout le monde, même si cela reste subordonné à l'achat ou à l'accès à du matériel informatique et à une connexion, le tout restant assez coûteux ;
- il est facile d'appréhension, le concept de lien hypertexte et son fonctionnement sont assimilables rapidement par tout un chacun et la création de pages HTML présente peu de difficultés pour les personnes réellement intéressées.

Il est donc naturel de tenter d'enrichir le contenu de base du Web que sont le texte et les images par des médias un peu plus riches comme les animations, l'audio, la vidéo, les scènes en trois dimensions et la réalité virtuelle par exemple. Depuis quelque temps déjà, il est facile d'intégrer dans des pages

Web de petites animations sous la forme d'images GIF89a. Il s'agit d'un format maintenant reconnu par tous les browsers permettant de regrouper plusieurs images formant une séquence animée dans un seul fichier contenant également des informations sur la vitesse d'affichage et d'autres paramètres, constituant ainsi une animation.

Cette tendance a continué avec l'apparition de plug-ins dans les browsers. Le principe est le suivant : ces derniers exportent une interface de programmation donnant accès à certaines de leurs fonctionnalités, comme l'accès à des données distantes et l'accès à l'affichage graphique. Cela permet à des éditeurs de logiciels de développer des modules traitant toutes sortes de formats de données. Une multitude de plug-ins permet aujourd'hui de traiter des formats propriétaires ou publics comme des photos panoramiques, dans lesquelles il est possible de naviguer, du son spatial, de la vidéo, des scènes en trois dimensions, décrites dans le langage VRML[1], etc.

Comme on peut le voir le Web est réellement "multi média"[2] dans le sens où il est le vecteur de présentation de nombreux médias de types variés. Toutefois, comme nous l'avons déjà dit, il manque un paramètre pour le transformer en système multimédia, c'est l'*intégration*. Il n'y a pour l'instant pas d'intégration possible de tous ces médias, c'est-à-dire qu'il n'y a pas de moyen pour les faire coopérer, les organiser en une présentation cohérente suivant un scénario défini. Chaque média a sa propre conscience du temps, mais celle-ci n'est pas partagée avec les autres. Il faut donc fournir la notion de temps à l'environnement afin que tous ces médias partagent la même et qu'ils puissent être synchronisés.

Le Web est un excellent support pour la diffusion du multimédia car il possède déjà de nombreuses caractéristiques qu'il serait profitable d'utiliser.

- Il est distribué : l'utilisation de l'hypertexte permet de créer des documents dont certaines des composantes comme les images ou les données utilisées par les plug-ins sont éparpillées sur des serveurs distants.
- Il est efficace, dans le sens où l'hypertexte permet une réutilisation maximale du contenu grâce à la désignation par référence, cette caractéristique découlant directement de la précédente.
- Il est éprouvé : bien que le Web soit utilisé de manière intensive uniquement depuis environ cinq ans, la technologie sous-jacente au Web

[1] Virtual Reality Modeling Language
[2] l'espace entre les deux mots est volontaire

est largement éprouvée.

4.1.2 Des technologies prometteuses

Les récentes avancées de la recherche en ce qui concerne le Web laissent entrevoir de nouvelles possibilités qu'il sera cependant possible d'exploiter pleinement uniquement lorsqu'elles auront été adoptées par l'ensemble des *browsers*, tout au moins les deux principaux, *Netscape Navigator* et *Microsoft Internet Explorer*.

Alors qu'un document hypertexte traditionnel, défini à l'aide de HTML, HyperText Markup Language, est mis en page et présenté par le browser, la présentation de documents multimédia requiert de nombreuses interactions et de nombreux traitements. Il faut les moyens de les réaliser, c'est-à-dire pouvoir intervenir dynamiquement dans le traitement du document par le browser.

De nouvelles technologies déjà existantes ou actuellement en cours de spécification ou de développement vont permettre des interactions, plus faciles et plus importantes, avec les documents. Nous allons les présenter en distinguant d'abord, celles qui ont trait à la description et la structuration du document, offrant ainsi la possibilité de manipuler le document à plusieurs niveaux et ensuite, celles permettant d'effectuer les manipulations en question. De nombreuses activités sont en cours dans ces deux domaines, principalement au W3C.

Description et structuration

D'énormes progrès ont été faits sur la description et la structuration des documents. La séparation entre la structure et le style[3] permet maintenant de réellement structurer les documents indépendamment de toute considération de style et inversement. Cela repose sur deux langages, HTML toujours, dont la nouvelle recommandation HTML 4.0[112] a été acceptée en début d'année, et CSS[111], Cascading Style Sheets. Le principe est d'épurer HTML de tout ce qui concerne le style et de le transférer dans CSS. CSS permet de définir

[3] On appelle style tout ce qui concerne la présentation, cela peut être le positionnement, les propriétés graphiques comme la couleur et la police de caractères, ou le volume pour l'audio.

```
H1,H2 { color: red; font-family: helvetica; }
P { text-indent: 1em; text-align: justify; }
```

Figure 4.1: Exemples de styles avec CSS

le style dans un langage adapté très intuitif intégrant la notion de classe d'éléments. Les exemples de la figure 4.1 signifient respectivement que les titres repérés par les tags `H1` et `H2` seront de couleur rouge et dans la police de caractères helvetica et que le texte contenu dans les paragraphes `P` sera indenté en fonction de la taille de police et justifié.

Aujourd'hui, un mouvement important se créé autour de XML, Extensible Markup Language, déjà présenté succinctement en 3.4.4, dans deux directions. D'une part, différents langages utilisant XML ont été développés ou sont en cours de développement, SMIL pour le multimédia, MathML pour la description de la présentation et du contenu mathématique et scientifique, un langage de description de graphiques vectoriels, pour ne citer que quelques exemples. D'autre part, de nombreux autres travaux visent à fournir un support aux langages et applications de plus haut niveau. On peut citer, par exemple, les activités concernant les méta-données et la description de ressources dont RDF, Resource Description Framework, est le produit, et d'autres activités directement liés à XML comme XPointer, XML Pointer Language, pour référencer la structure interne de documents XML, XLink, XML Linking Language, pour créer des liens de type HTML ou plus évolués entre des références de documents XML, ou XML Namespaces, pour qualifier les noms XML et ainsi pouvoir utiliser plusieurs vocabulaires dans un même document[4].

Interfaçage et programmation

Un second point est aussi important que la représentation de l'information dont nous avons parlé ci-dessus, c'est la possibilité d'y accéder et de la traiter. À cette fin, le W3C a défini une autre activité, DOM pour Document Object Model. Il s'agit d'une interface indépendante de tout langage et de toute plate-forme offrant accès à la structure, au contenu et au style des documents

[4]Un simple exemple est l'utilisation de constructions MathML dans un document HTML : le document est interprété comme HTML mais les élément MathML sont identifiés par un préfixe signifiant à l'application qu'ils doivent être traités comme MathML et non HTML.

et permettant de les modifier dynamiquement, et ce par l'intermédiaire de langages de programmation et de script. DOM est défini pour le moment pour HTML et XML. Les technologies antérieures, *Dynamic HTML* de *Netscape* et *DHTML* de *Microsoft*, sont un embryon d'implémentation de DOM, dans le sens où elles proposent des interfaces dépendantes de langages donnés et offrent un faible contrôle sur le document.

Les langages prenant une importance toute particulière dans le cadre du Web sont ECMAScript et Java. ECMAScript est un langage de script standardisé par ECMA[5] sous la référence ECMA-262 et unifiant les deux langages qui en sont à l'origine, *JavaScript* de *Netscape* et *JScript* de *Microsoft*. Ce langage peut être directement intégré aux pages HTML classiques pour y ajouter un aspect dynamique. Cela peut être le traitement d'événements survenant dans la page ou la vérification de paramètres passés dans un formulaire. Java, présente un intérêt plus grand dans notre optique multimédia sur le Web puisqu'il s'agit d'un langage orienté objet doté de nombreuses bibliothèques de base permettant de développer de véritables applications, destinées ou non au Web. Ce langage introduit par Sun en mai 1995 présente la particularité d'avoir été conçu dans l'optique d'une utilisation à travers le réseau et offre des garanties de sécurité. Il est donc possible d'exécuter des applications embarquées dans des pages HTML, appelées *applets*, sans mettre en péril la sécurité et l'intégrité de la machine hôte. Ce langage allié à DOM permet d'envisager la réalisation de traitements complexes de haut niveau sur des documents HTML et XML, par exemple afin de réaliser des présentations multimédia.

4.2 Objectif

Nous avons vu dans la section précédente le contexte dans lequel nous avons choisi d'évoluer. Toutes ces technologies autour du Web, existantes ou en cours de définition, nous permettent de penser que l'ajout de fonctionnalités multimédia est possible. Cela doit se faire à quatre niveaux.

- *Description* : une composante temporelle et le moyen de spécifier la synchronisation sont nécessaires. Nous avons pris le parti d'une approche déclarative en étendant le langage HTML avec de nouveaux

[5]ECMA - European association for standardizing information and communication systems, anciennement European Computer Manufacturers Association

tags et attributs temporels. Cet aspect sera développé dans la suite de ce chapitre.

- *Client* : les clients Web, que l'on appelle browsers, doivent être agrémentés d'un environnement de présentation pour de tels documents multimédia et notre objectif a été de réaliser un prototype capable de fonctionner dans un browser grand public comme *Netscape Navigator*. Une architecture d'exécution a été conçue et prototypée en Java, comme nous le verrons dans le chapitre suivant.

- *Transport* : ce niveau établit le contact client-serveur. Nous avons abordé ce problème du point de vue du client en étudiant comment il est possible de lui donner un certain contrôle sur le transport par des interactions avec les serveurs. Cela sera également exposé dans le chapitre suivant. Beaucoup de travail reste à faire dans ce domaine. Des protocoles existent mais ils ne suffisent pas. Notre objectif est de proposer un environnement multimédia adaptatif capable de prendre des décisions de compromis en fonction de ses conditions d'évolution nécessitant un support de négociation entre le client et les divers serveurs mis en jeu lors de présentations.

- *Serveur* : une alternative est possible ici, soit les serveurs Web sont étendus avec de nouvelles fonctionnalités leur permettant de supporter les types de données multimédia qui posent des problèmes particuliers quant à leur taille et leur sensibilité au temps, soit des serveurs multimédia spécifiques sont utilisés.

Le travail qui va être exposé dans la suite de ce rapport a fait l'objet de deux publications[95, 93] ainsi que de la présentation d'un poster[94].

Afin d'illustrer notre objectif voici un simple exemple de ce que l'on veut pouvoir produire comme résultat. La figure 4.2 présente deux vues d'une même page HTML. Il s'agit d'une page faisant partie d'une présentation multimédia. Lorsqu'elle est chargée, image en haut à gauche, un commentaire audio commence, puis une fois terminé, il est remplacé par une musique de fond. Lorsque l'utilisateur actionne un des liens dans le texte, une vidéo arrive en glissant par le coin en haut à droite pour s'arrêter sur le texte et continuer jusqu'à ce qu'elle se termine. C'est ce que montre l'image en bas à droite.

On peut, à partir de cet exemple basique, imaginer des scénarios beaucoup plus complexes où les médias et interactions sont plus nombreux. Ce simple

Figure 4.2: Exemple de document multimédia simple

exemple permet d'identifier des caractéristiques nécessaires à la création multimédia. Évidemment en premier lieu l'inclusion d'objets multimédia divers et la spécification de la synchronisation. Puis l'interaction avec l'utilisateur, pas simplement au sens contrôle du déroulement linéaire de la présentation qui est d'un faible intérêt créatif, mais en offrant réellement des capacités hypermédia permettant, par exemple, d'enrichir un document à l'aide de séquences multimédia lancées à volonté par l'utilisateur à l'aide de liens. Finalement, un élément important est la capacité à réaliser des effets visuels et sonores afin de marquer les transitions ou d'attirer l'attention sur les principaux points. Ce dernier aspect est important car les documents multimédia présentent simultanément une importante quantité de données. À cette fin, les effets graphiques de mouvement sont intéressants et soulèvent de nouvelles questions. On ne peut pas simplement superposer le temporel à la disposition spatiale initiale du HTML, il faut une intégration réelle permettant des interactions fortes.

La spécification de SMIL a eu lieu pendant ce travail thèse, et comme ce nouveau langage aborde un problème semblable, il nous sera souvent donné d'y revenir afin de faire des comparaisons. Avec SMIL, le choix d'un nouveau langage a été fait, ce qui limite l'intégration du multimédia au Web. On retrouve les inconvénients posés par les plug-ins, à savoir le manque d'interaction entre document HTML et document SMIL. Deux choses sont

possibles, embarquer une présentation SMIL dans un document HTML, ou inversement inclure des pages HTML dans une présentation SMIL. À l'inverse, notre démarche a été d'essayer d'intégrer le multimédia au Web en étendant HTML. La même approche est proposée dans une récente soumission au W3C datant de septembre 98, Timed Interactive Multimedia Extensions for HTML (HTML+TIME) Extending SMIL into the Web Browser, et soutenue principalement par *Microsoft*.

Nous pouvons faire remarquer ici qu'une grande partie du travail relatif à l'Internet en général et plus particulièrement au Web, puisqu'il s'agit d'un support de publication, est publié sur le Web sous la forme de documents HTML. L'évolution dans le domaine est extrêmement rapide et il est parfois difficile de reconstituer la chronologie des événements. Une première particularité est la compétition que se livrent les deux grands acteurs du secteur, *Netscape* et *Microsoft*, qui produit le schéma classique suivant : chacun développe des extensions au standard HTML existant en ajoutant de nouveaux tags ainsi que leur traitement dans son browser. Cette divergence est ensuite éventuellement réduite par la normalisation d'une des solutions ou plus vraisemblablement d'un compromis entre les deux. Une seconde particularité est la publication de documents de travail ou *working drafts* sur lesquels se base ensuite le processus de normalisation. Il en résulte une multitude de propositions dont des aspects seront éventuellement repris et ainsi diffusés largement, les autres restant en général très peu diffusés et visibles.

4.3 Proposition

Dans les sections suivantes, nous allons formuler précisément la proposition d'extension de HTML que nous avons choisi de réaliser. Commençons toutefois par préciser les choix généraux que nous avons fait a priori, afin que cette proposition s'inscrive naturellement dans l'environnement existant du Web défini par HTML, sans en bouleverser l'esprit.

- Les extensions proposées sont de nature textuelle. Une présentation multimédia est donc éditable sans autre outil qu'un simple éditeur de texte.

- Nous avons trouvé un compromis entre un niveau d'abstraction suffisamment bas pour ne pas induire un traitement lourd sur le client avant

que celui-ci puisse réaliser la présentation d'un document, et un second niveau suffisamment élevé pour qu'un document reste compréhensible à la lecture.

- Nous proposons un format de description et dans ce sens nous avons privilégié son expressivité. Celle-ci garantit que tout type de scénario pourra être exprimé à l'aide des extensions proposées. Comme pour le HTML standard, aucune cohérence n'est imposée. Un document valide doit être présenté correctement, par contre s'il est mal formé ou incohérent on ne peut pas prédire le résultat obtenu[6]. La vérification de la cohérence est du ressort d'un éventuel éditeur multimédia et du moteur d'exécution servant à réaliser la présentation.

Nous allons commencer par présenter dans les sections qui suivent les concepts de base en ce qui concerne la synchronisation, puis la représentation des objets et enfin l'intégration du domaine spatial et du domaine temporel. Finalement, les nouveaux tags que nous proposons afin de concrétiser ces concepts dans HTML seront introduits.

4.4 Modèle de synchronisation

Le modèle temporel que nous proposons d'utiliser est dérivé des graphes d'instants. Celui-ci repose sur deux concepts centraux que nous allons maintenant préciser : le *lien hypertemps* et la *base temporelle*.

4.4.1 Composition temporelle

Liens hypertemps

Le *lien hypertemps*, *hypertime link*, est un paradigme dérivé des graphes d'instants et constitue une analogie directe avec le lien hypertexte que l'on connaît déjà dans HTML. Nous utiliserons dans la suite indifféremment les termes lien hypertemps et lien temporel.

[6] Cela n'exclut pas le fait qu'un outil de présentation puisse signaler les erreurs ou essayer de réaliser une présentation la meilleure possible même en cas d'incohérence.

Le lien hypertexte est un concept puissant et intuitif qui a fait en partie le succès du langage HTML. Il possède une sémantique opératoire précise : lorsque l'on clique sur l'origine il nous emmène vers la destination qu'il référence. De plus, il n'est pas typé et peut représenter des relations diverses : note, renvoi, référence, suite, etc. L'hypertexte constitue un moyen efficace pour structurer un nombre important de documents de manière transverse, indépendamment de leur situation dans un système de fichiers ou dans l'espace distribué de l'Internet.

Nous proposons d'étendre ce concept au domaine temporel pour structurer un ensemble d'objets temporels constituant une présentation multimédia. La sémantique du lien hypertemps est définie intuitivement par analogie avec celle du lien hypertexte, ce qui donne explicitement :

- un lien hypertemps relie un instant d'origine à un instant de destination ;
- un lien hypertemps est déclenché automatiquement lorsque l'instant d'origine devient l'instant courant, sans intervention de l'utilisateur ;
- un lien hypertemps fait coïncider l'instant de destination avec l'instant courant.

La figure 4.3 illustre le lien temporel. Les objets sont représentés par leur intervalle temporel dont les extrémités représentent le début et la fin. L'arrêt de la musique provoque le départ de la vidéo. À un instant ultérieur, le commentaire démarre et lorsque celui-ci se termine, il déclenche l'affichage d'une image. Celle-ci est retirée de l'écran quand la vidéo se termine.

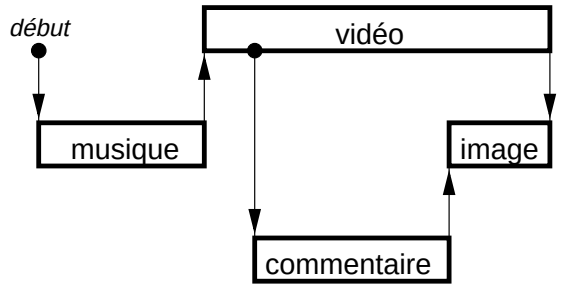


Figure 4.3: Liens hypertemps

Liens hypermédia

Un lien hypermédia est un lien dont l'origine est à la fois spatiale et temporelle. Cela permet de définir sur un média une zone spatio-temporelle qui peut être activée par l'utilisateur. Par exemple, une séquence d'images dans une vidéo qui représente une zone temporelle définie par le début et la fin de cette séquence et une zone spatiale définie par la zone graphique d'affichage.

Nous étendons la notion d'origine d'un lien hypertemps à un segment temporel défini par deux instants. Cela permet de référencer une portion d'un objet multimédia et ainsi de définir la composante temporelle d'un lien hypermédia.

Points temporels

Afin d'ancrer les liens hypertemps, la référence à des instants précis sur des objets est nécessaire. Cela se fait par la définition de *points temporels*, *time points*, qui peuvent ensuite être référencés comme origine ou destination de liens temporels. Un instant peut être de deux types :

- soit c'est un événement lié à l'objet, il sera alors défini par une valeur nominale : le point est lié au média et représentera toujours la même position dans les données, le début et la fin étant des cas particuliers ;
- soit c'est une date absolue sur l'objet : le point est indépendant du média, par exemple trente secondes après le début de l'objet.

Ci-dessus, le terme "absolu" signifie que la valeur de la date n'est pas liée au déroulement de l'objet mais que celle-ci est définie de manière absolue par rapport au début de l'objet. Un lien temporel reste toutefois indépendant de toute valeur de temps absolue et de ce fait le modèle tolère l'indéterminisme. Dans l'exemple cité ci-dessus et illustré sur la figure 4.3, il est possible de faire démarrer la vidéo même si l'on ne connaît pas a priori la durée de la musique.

Causalité

Le modèle temporel résultant des liens hypertemps et des points temporels est équivalent au modèle par graphes d'instants vu en 3.2.3. Il permet d'exprimer tous les types de scénario possibles mais ne garantit pas la cohérence de ces derniers. Plus précisément, le lien temporel est équivalent au déclencheur, *enabler*, défini dans *FLIPS*[98].

La grande force du lien temporel est la *causalité*. Cela signifie qu'il provoque un effet en rapport avec une cause et ne décrit pas simplement un état de fait. Ainsi, un lien peut provoquer différentes actions suivant son instant de destination :

- si la destination est la fin d'un objet, le lien provoque son arrêt ;
- s'il s'agit de tout autre instant dans l'objet, il provoque sa présentation à l'instant donné, donc son démarrage s'il n'était pas actif, le saut à cet instant dans le cas contraire.

Il est important de bien distinguer le lien temporel tel que nous le définissons de la relation de simultanéité. Lors de la présentation, les instants d'origine et de destination seront en théorie[7] simultanés, mais il s'agit de la résultante de l'action définie par le lien — démarrer, aller à, arrêter — et non pas d'un état de simultanéité entre ces instants. Cette approche se distingue donc des modèles comme *Firefly*[9] définissant des relations d'état entre les instants. Ceux-ci utilisent une notion de "temps élastique" pour rendre les spécifications cohérentes en dilatant ou contractant le temps : des états sont définis entre des instants et ensuite les durées des objets ajustées afin de respecter ces états, à moins qu'aucune solution ne soit possible.

Les relations définissant des états entre objets peuvent générer de nombreuses incohérences car le comportement des objets n'est pas nécessairement connu par avance. Comme nous l'avons déjà vu, des perturbations extérieures ou la nature même des objets peuvent induire un certain indéterminisme. Ce type d'incohérence n'existe pas avec des relations causales, ce qui est illustré sur la figure 4.4.

[7] Moyennant les temps de déclenchement liés au système.

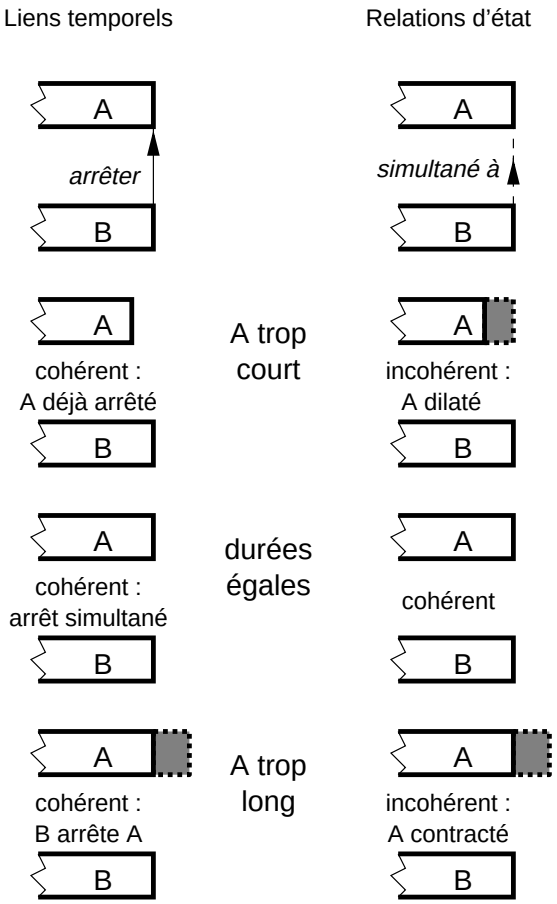


Figure 4.4: Comparaison entre les liens temporels et les relations d'état sur les instants

4.4.2 Couplage

La spécification de la composition temporelle n'est pas suffisante pour définir la synchronisation d'une présentation multimédia. Plus d'information est nécessaire pour préciser quelle sera la nature des relations temporelles entre des objets présentés simultanément, leur couplage (voir 2.2.2). Dans un environnement n'offrant pas de garanties temps réel, plusieurs facteurs peuvent produire une dérive temporelle entre les objets, conduisant ainsi à une désynchronisation, principalement :

- les délais dus au traitement dans un système non temps réel ;

- la surcharge du système induisant un délai important entre le moment où une action aurait dû être effectuée et sa date réelle ;
- les perturbations extérieures venant des accès aux données sur les périphériques, notamment les communications sur le réseau.

Dans certains cas, l'effet sur la présentation de la désynchronisation entre des objets est négligeable, alors que dans d'autres situations, le résultat peut être très gênant, voire catastrophique. En règle générale, le décalage entre une musique de fond et une vidéo est imperceptible, alors qu'à l'inverse les paroles d'un personnage doivent être maintenue synchronisées avec son image de façon très stricte. Il est donc important pour un auteur de pouvoir spécifier la nature du couplage entre les objets ainsi que le type d'action corrective devant être exécutée en cas de problème. Il faut spécifier trois choses :

- quels sont les objets en relation ;
- lesquels sont maîtres du temps et quels autres doivent subir (qui mène la danse si l'on veut) ;
- quel est leur degré de couplage ou granularité de synchronisation.

On peut constater que dans les modèles présentés dans le chapitre précédent la spécification du couplage entre les éléments composant une présentation multimédia est presque ignorée. Soit le couplage est implicitement défini comme étant de nature *lip-sync*, soit il est complètement laissé à la discrétion de la machine de présentation, comme dans SMIL. La question est donc traitée uniquement dans la partie logicielle effectuant la présentation.

Bases temporelles

À cette fin, nous définissons la notion de *base temporelle*, *timebase*. Une base temporelle est un espace temporel virtuel dans lequel évoluent des objets multimédia. Elle définit un système de coordonnées commun à tous les objets qu'elle regroupe. On peut imaginer une base temporelle comme un espace temporel parfait où les phénomènes réels de dérive n'existent pas, où les objets se comportent donc de manière parfaite. Il ne s'agit effectivement que d'une abstraction qui peut cependant être approchée de près sur un système temps réel. Ce n'est pas le cas des systèmes qui nous intéressent, il faut donc

indiquer comment maintenir une qualité de présentation acceptable et quelle est la nature de la synchronisation désirée.

Nous définissons la nature de la synchronisation entre des objets multimédia évoluant au sein d'une même base temporelle en utilisant les notions de *maître* et *esclave*. Classiquement une relation maître-esclave entre deux objets signifie que l'un d'eux, le maître, contrôle le second, l'esclave, selon ses besoins (figure 4.5a). Nous généralisons cette notion à de multiples maîtres et esclaves en transformant la relation initiale en une relation indirecte par l'intermédiaire d'une base temporelle commune (figure 4.5b).

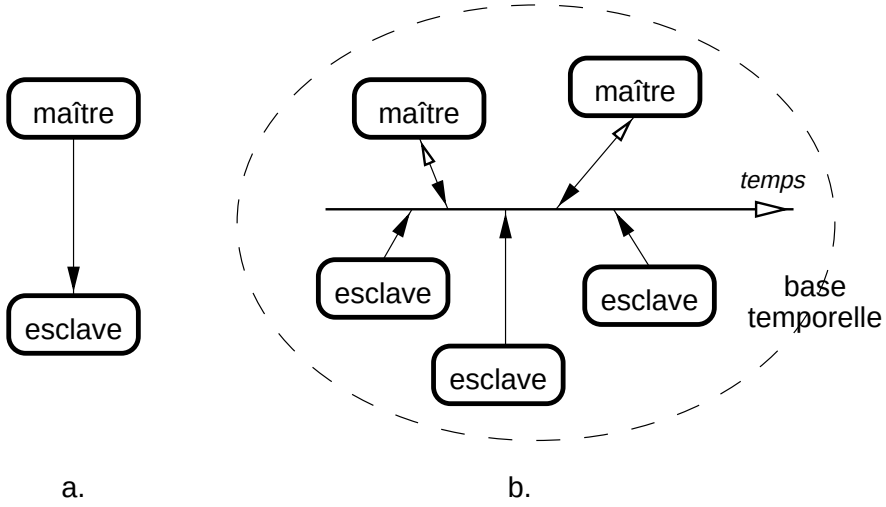


Figure 4.5: Relation maître-esclave

Dans ce cas, la relation maître-esclave est caractérisée ainsi. Chaque maître possède la maîtrise du temps, c'est-à-dire qu'il a le moyen de réajuster le temps à sa convenance si nécessaire, provoquant également un réajustement chez tous les autres objets de la même base temporelle, esclaves et maîtres confondus, afin que la synchronisation soit conservée. À l'inverse, les esclaves subissent le temps et en cas de problème, ils doivent se maintenir en synchronisation par leur propres moyens, en annulant une partie de leur présentation afin de rejoindre la position temporelle courante. La conséquence directe de ce mécanisme est que les maîtres seront toujours présentés dans leur intégralité alors que les esclaves pourront avoir à supprimer une partie de leur présentation.

Un exemple illustrant ce propos est une vidéo présentée avec un commentaire et des sous-titres. Ces deux derniers objets sont alors définis comme maîtres car ils représentent des données importantes, et la vidéo est esclave. Si

un problème survient pendant la diffusion du commentaire, les deux autres objets seront arrêtés le temps que celui-ci reprenne son cours normal. Par contre, si la vidéo ne peut pas suivre le rythme de la présentation, elle ne perturbera pas la déroulement des autres médias. Elle devra par exemple passer des images afin de se réaligner sur la position courante.

Un tel comportement n'est pas une fin en-soi, mais permet de se conformer à la sémantique de synchronisation définie par l'auteur. Il serait préférable, dans l'exemple cité ci-dessus, que l'objet vidéo prenne une décision permettant une présentation correcte avant qu'il soit nécessaire de passer des images. Cela ne dépend pas du document, il s'agit purement d'un problème d'implémentation de la partie logicielle responsable de la vidéo. Nous verrons plus loin qu'une négociation avec le serveur envoyant la vidéo permet par exemple de sélectionner un format correspondant au débit disponible sur le réseau et d'éviter ainsi l'introduction de délais entraînant un traitement plus radical par la suite.

Points de synchronisation

Il est parfois souhaitable de pouvoir raffiner la spécification de la synchronisation en spécifiant la granularité à laquelle elle doit être effectuée. La relation définie ci-dessus précise uniquement la méthode à appliquer pour resynchroniser des objets en fonction de leur rôle, maître ou esclave. Les *points de synchronisation*, *synchronization points*, permettent de définir quand les resynchronisations doivent avoir lieu. Notons que le terme utilisé est le même que dans *MODE*[7], cependant la sémantique diffère.

Les points de synchronisation sont spécifiés par l'auteur de la présentation et identifient, sur les objets, les instants particuliers auxquels une resynchronisation peut être effectuée. Chaque point est défini sur un objet donné mais concerne tous les objets se trouvant dans la même base temporelle. Il est également possible de définir toute une séquence de points à partir d'une période. Finalement, dans le cas où aucun point de synchronisation n'est déclaré dans une base temporelle, le couplage devra être réalisé à la granularité la plus fine, c'est-à-dire que la restitution de tous les LDU — image vidéo, buffer de données audio, etc. — sera considérée comme un point de synchronisation. D'après le comportement des maîtres et esclaves définis plus tôt, la resynchronisation s'effectue de la façon suivante :

- si un maître est en retard, les autres objets l'attendront au point de synchronisation ;
- lorsque tous les maîtres ont atteint le point de synchronisation, ce dernier est libéré et l'exécution continue ;
- si un esclave est en retard, il devra rejoindre le point de synchronisation instantanément.

Ce mécanisme permet de relâcher les contraintes de synchronisation lorsqu'une synchronisation fine telle que le *lip-sync* n'est pas nécessaire mais qu'une cohérence doit être maintenue dans le document en certains points. Cela permet de définir une synchronisation plus spécifique, dépendante du contenu des médias, chose que seul l'auteur peut spécifier puisqu'il en connaît le contenu. La qualité de synchronisation nécessaire entre deux objets dépend des relations de contenu qui existent : plus celles-ci sont importantes, plus la synchronisation doit être fine.

Prenons l'exemple suivant, la présentation d'un voyage autour du monde composée d'une vidéo, d'un commentaire et d'une musique de fond. La vidéo est une suite de séquences présentant chaque pays traversé. Le commentaire est divisé de la même façon et décrit succinctement l'itinéraire. La musique de fond recrée une ambiance en accord avec le pays dont il est question. Dans ce cas, une synchronisation fine de type *lip-sync* n'est pas nécessaire, il suffit de s'assurer que les transitions se feront simultanément sur les trois objets pour ne pas qu'en cas de décalage, on se retrouve dans une situation où le commentaire sur les États-Unis commence pendant que la vidéo sur le Japon est à l'écran sur fond de musique indienne qui n'a pas encore eu le temps de se terminer.

Dans ce cas, la synchronisation peut être spécifiée de deux façons, suivant que l'on considère qu'il est important que les séquences vidéo sur chaque pays soient présentées en intégralité ou au contraire que la fluidité du commentaire est primordiale. La figure 4.6 illustre ce propos. Dans les deux cas, le commentaire sera maître, la musique esclave et des points de synchronisation seront définis entre les séquences ce qui assure que les pauses éventuellement effectuée pour resynchronisation se feront à un instant causant le plus faible désagrément possible : transition dans la vidéo, écran noir, titre, etc. ou transition dans le commentaire, assurant qu'aucune phrase ne sera coupée. Dans le premier cas, la vidéo sera maître, ce qui permettra à la vidéo d'être présentée en intégralité, le commentaire réalisant une pause

aux inter-séquences en cas de désynchronisation. Dans le second cas, la vidéo sera esclave, assurant ainsi la continuité du commentaire, et en cas de retard de la vidéo ou de la musique celles-ci seront recalées sur la bonne séquence suivante.

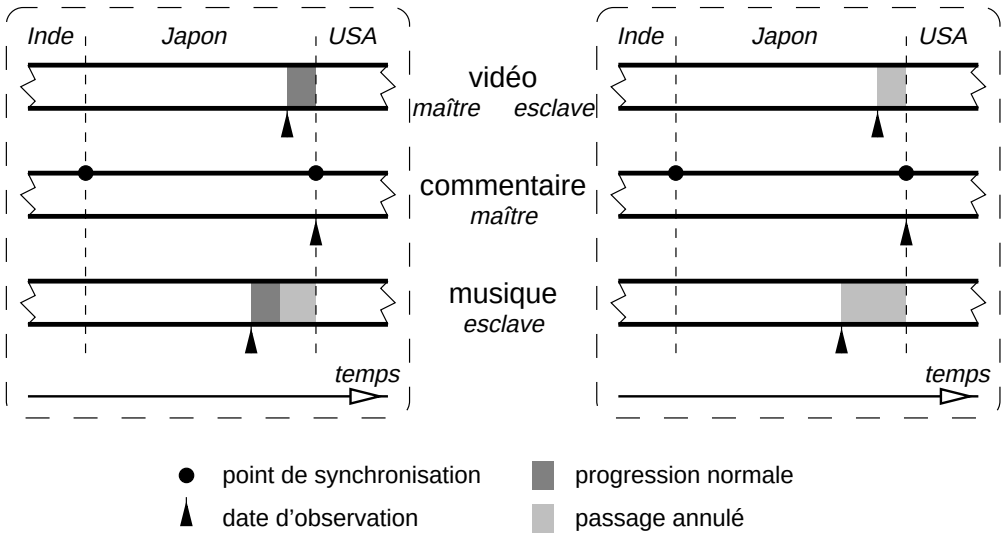


Figure 4.6: Exemple d'utilisation des points de synchronisation

4.5 Objets multimédia

Le modèle de synchronisation étant exposé, nous allons maintenant définir quelle est la nature des objets multimédia. Afin de garantir un modèle multimédia général, nous considérons la définition la plus large possible. Sur ces objets sont définis des points temporels et des points de synchronisation afin de pouvoir spécifier la synchronisation.

4.5.1 Objets de base

Les objets de base appartiennent à l'une des catégories suivantes, déjà évoquées plus tôt dans le rapport.

- Médias statiques : leur durée sera définie explicitement par association d'une valeur temporelle, ou implicitement comme conséquence de la

composition temporelle. Un objet statique aura toujours un début. Sa fin sera définie explicitement grâce à une durée, implicitement comme destination d'un lien, ou n'aura jamais lieu — dans ce dernier cas un lien temporel ayant pour origine la fin de l'objet ne sera jamais traversé. La figure 4.7 représente ces trois cas, dans le dernier la vidéo ne sera jamais démarrée.

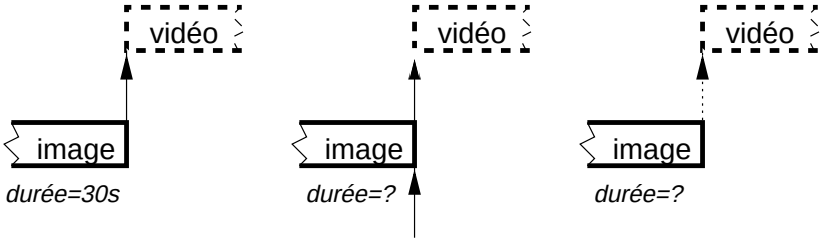


Figure 4.7: Média statique

- Médias continus : ils ont un comportement temporel intrinsèque, déterministe ou indéterministe et leur durée pourra être connue ou inconnue. Entrent dans cette catégorie les médias de base traditionnels comme l'audio et la vidéo, mais aussi les animations en temps réel, les scènes de réalité virtuelle, etc.

- Programmes : cette catégorie regroupe les objets dont le contenu est du code exécutable. Ils peuvent avoir ou non une représentation graphique et sont utilisés dans plusieurs buts : interaction avec l'utilisateur, contrôle sur la présentation. Ce peut être par exemple des éléments graphiques interactifs comme des boutons (voir figure 4.8) ou des opérateurs implémentant des fonctions temporelles diverses — nous y reviendront dans le chapitre suivant.

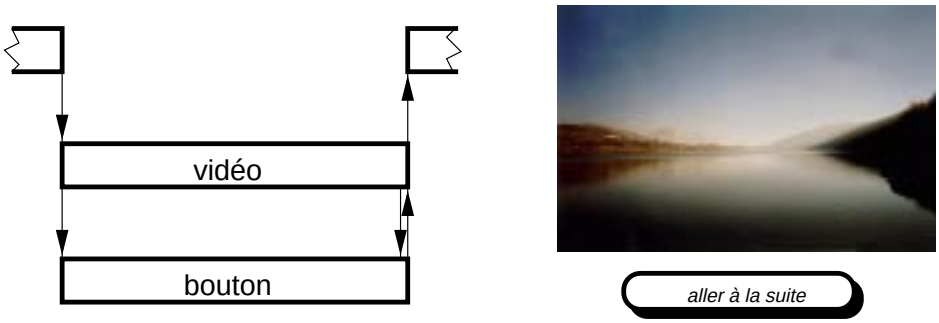


Figure 4.8: Objet interactif

Comme c'est déjà le cas pour les objets non textuels de HTML, les objets multimédia référencent leur contenu par l'intermédiaire d'un URL[6]. Cela est indispensable pour garder la spécification au format texte et offre l'avantage de permettre la réalisation de présentations dont les composants sont distribués sur le réseau.

4.5.2 Objets composites

Afin de faciliter la création de documents complexes, il est nécessaire de disposer d'un moyen permettant de regrouper les objets en modules qui seront eux-mêmes vus de l'extérieur comme des objets. Cela permet de clarifier l'organisation des objets et de simplifier la composition temporelle. Il est ainsi possible de référencer des instants définis, soit sur l'objet composite, soit sur ses composants internes.

Le début d'un objet composite est défini explicitement par un ou plusieurs liens temporels démarrant un ou plusieurs composants. Sa fin correspond à la fin du dernier composant actif. De plus, tout objet de base doit être associé à une base temporelle. Celle-ci définit la référence de temps utilisée pour la progression de l'objet. Un objet composite n'est associé à aucune base temporelle, ce sont uniquement ses composants de base qui le sont. La figure 4.9 illustre ces propos par quelques exemples.

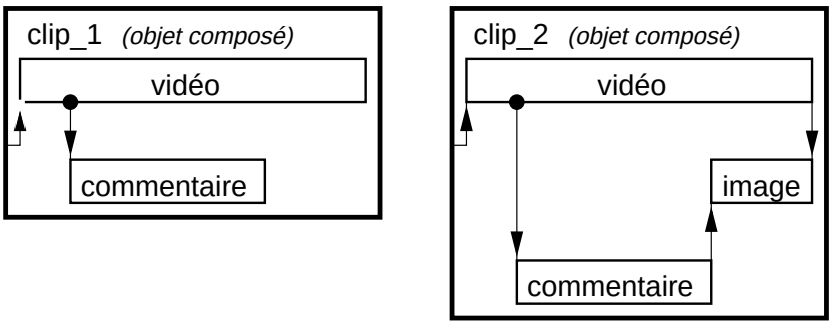


Figure 4.9: Objets composés

4.6 Intégration spatial-temporel

4.6.1 Disposition spatiale

Finalement, nous exposons les solutions proposées pour intégrer réellement la dimension spatiale et la dimension temporelle des documents multimédia. Par intégration, nous entendons interaction forte, c'est-à-dire la possibilité de définir des dépendances temporelles sur la représentation spatiale des objets et non pas seulement sur les objets. Un simple exemple permet d'exprimer cette distinction. Supposons que l'on veuille synchroniser le mouvement d'une vidéo traversant l'écran avec de la musique : la vidéo doit partir du coin en haut à gauche de l'écran lorsque la musique démarre, vers le coin en bas à droite et arrêter son mouvement lorsque la musique se termine. Ce que nous appelons "la vidéo" est en fait sa représentation, c'est-à-dire la zone graphique dans laquelle elle est affichée. La vidéo elle même est un objet distinct de cette zone spatiale et peut être synchronisée avec d'autres objets.

Cette approche dissociant un objet de sa représentation, c'est-à-dire sa disposition spatiale lors de sa présentation, permet de réaliser des effets spatio-temporels intéressant dans le cadre de l'édition multimédia et de rendre dynamique la présentation de contenu statique comme des portions de HTML "pur" présentées en synchronisation avec des médias temporels.

Les exemples cités ci-dessus sont du domaine graphique mais cette distinction est également valable dans le domaine sonore. Dans le cadre du son spatialisé[68, 83] les facteurs de positionnement sont généralement azimut, élévation et distance ou volume. En même temps que la vidéo de l'exemple précédent traverse l'écran, le son peut être localisé dans l'espace pour donner l'impression qu'il la suit. Cependant, étant peu familiers du domaine, nous n'avons pas encore traité ce sujet, qu'il faudra aborder dans le futur. Pour le moment, l'utilisation du son spatialisé est très peu répandue et demande un équipement acoustique spécifique. Toutefois des solutions simplifiées basées uniquement sur la stéréo existent.

4.6.2 Layouts et frames

Ainsi, il est possible de rendre dynamique la disposition des éléments graphiques que nous appelons *dynamic layout*. Pour cela, il est nécessaire de

faire la distinction entre un objet multimédia et sa représentation graphique que nous proposons de représenter par deux objets distincts. Chacun des deux peut alors prendre part dans la composition temporelle et les relations de couplage.

Nous introduisons deux nouveaux objets de type particulier, le *layout* et la *frame*[8]. Un layout représente un espace de coordonnées graphiques dans lequel sont disposées des frames. Une frame définit une région graphique à l'intérieur d'un layout et représente la zone d'affichage allouée à un objet multimédia de base ou un layout qui lui est associé. Ce dernier cas permet de définir des layouts imbriqués. La figure 4.10 présente un exemple de disposition spatiale évoluant dans le temps. Lorsque l'audio se termine un nouveau layout lay3 est utilisé pour afficher les objets.

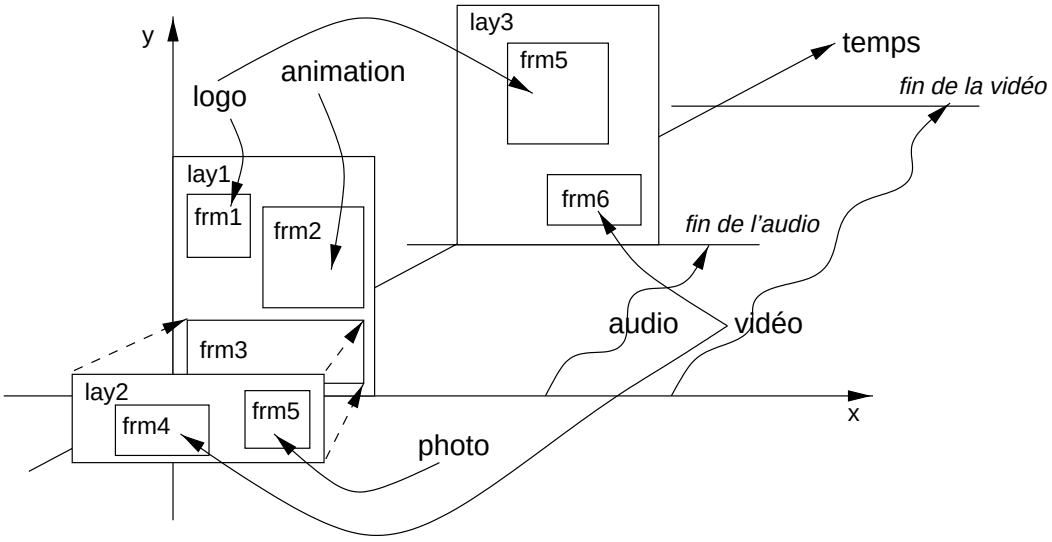


Figure 4.10: Exemple de disposition spatiale

Il paraît aujourd'hui logique d'envisager de traiter ce problème de positionnement dynamique avec CSS, Cascading Style Sheets, qui est le langage de description de style associé à HTML. Cependant, cela n'est pas aussi évident qu'il y paraît : il faut en effet pouvoir spécifier des relations temporelles entre les zones graphiques d'affichage et les objets multimédia classiques, ce qui représente une difficulté lorsqu'ils ne sont pas définis de façon homogène en utilisant un même langage. Pour des raisons de simplicité et de clarté, nous avons défini le dynamic layout comme une extension à HTML.

[8]N'ayant pas de traduction française appropriée, nous allons utiliser la terminologie anglaise

Cela n'empêche pas dans le futur d'étudier son intégration à CSS. La première étape et la plus évidente est de conserver la définition des layouts et frames en HTML et de spécifier leur positionnement avec CSS.

La composition temporelle associée aux layouts permet de définir aisément une disposition spatiale qui évolue dans le temps de façon discontinue, comme dans l'exemple de la figure 4.10 où une première disposition est remplacée par une alternative à une date donnée. Cependant nous voulons également pouvoir définir un comportement temporel continu comme dans l'exemple de la vidéo traversant l'écran, et cela requiert la spécification du mouvement. Nous n'avons pas encore complété cette partie du langage qui demande une étude précise des méthodes existantes de description du mouvement, puis une adaptation de celles-ci ou la définition d'une nouvelle. Nous avons pour le moment seulement envisagé le mouvement sous la forme d'une trajectoire et d'une vitesse de parcours associée, la figure 4.11 en montre un exemple. Peut-être serait-il utile de définir des types de mouvement générique contrôlable par quelques paramètres, par exemple *slide*, *swirl*, *gravity*, etc.

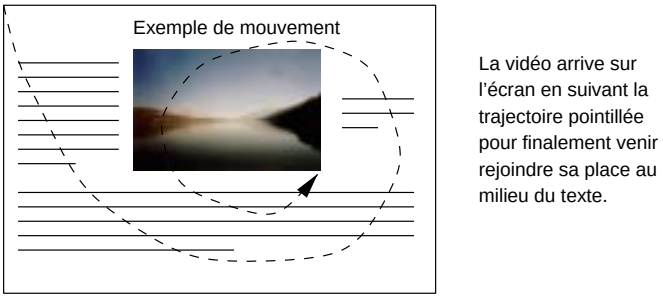


Figure 4.11: Mouvement des objets

4.7 Extensions à HTML

Nous allons présenter dans cette section les tags et les attributs correspondant aux divers éléments que nous avons proposé ci-dessus afin d'étendre HTML pour la description de documents multimédia. Pour des raisons de lisibilité, nous avons décrit les tags par leur syntaxe, ce qui est beaucoup plus facile à lire qu'une DTD. De plus, ce travail n'en est pas à un stade de formalisation suffisant pour justifier une DTD. Nous introduisons des idées qui sont encore étudiées et ne sont pas définitives. Pour cette raison, nous ne présentons pas de manière exhaustive les attributs standard de HTML, ceux

qui sont définis pour tous, ou un grand nombre d'éléments comme les *core attributes*, *i18n* pour l'internationalisation et *events* définissant les événements reconnus par les objets.

Dans la description qui suit, nous utilisons la convention typographique suivante : les symboles de type `normal` sont terminaux, ***incliné*** non terminaux et *italique* spécifient le contenu possible.

4.7.1 Hypertime link

Un lien hypertemps relie un instant ou un intervalle d'origine à un instant destination. Les instants sont définis à partir de l'identificateur d'un objet et de l'identificateur d'un point temporel défini sur cet objet. Lorsque l'origine est omise celle-ci est implicitement définie comme le début du document.

```
< htlink orig   = " time-point-id |
                  [ time-point-id , time-point-id ] "
         target = " time-point-id | time-point-id " >
```

Exemples :

- `<htlink orig="video.audio-trigger" target="audio.beg">` démarre un clip audio quand le point temporel `audio-trigger` défini sur l'objet `vidéo` est atteint ;
- `<htlink orig="clip.audio.end" target="video.end">` arrête la vidéo lorsque l'objet `audio` de l'objet composé `clip` se termine.

4.7.2 Time point

Un point temporel définit un instant d'un objet multimédia qui peut être utilisé comme origine ou destination d'un lien temporel. Le positionnement peut être de deux sortes : nominal ou absolu. Un point nominal est positionné dans le temps en utilisant la fréquence nominale de présentation de l'objet et relativement à celui-ci. Un point absolu repère une date indépendamment du déroulement de la présentation de l'objet.

```
< time-point id    = " name "
             value = " integer value | float value "
             unit  = " frame | sample | timestamp | second "
             type  = " nominal | absolute " >
```

Les points temporels `beg` pour début et `end` pour fin sont définis par défaut pour les objets continus. Seul `beg` est défini pour les objets statiques. Ces deux points peuvent être surchargés afin de redéfinir le début et la fin d'un objet, lui imposant ainsi une nouvelle durée.

Exemples :

- `<time-point id="audio-trigger" value="300" unit="frame" type="nominal">`

  définit un time point sur l'image 300 d'une vidéo par exemple ;

- `<time-point id="wait-20s" value="20" unit="seconds" type="absolute">`

  positionne un time point 20 secondes après le début de l'objet ou du layout contenant.

4.7.3 Media object

Les objets multimédia peuvent être de nombreux types. Nous avons choisi de les représenter en utilisant un tag `object`. Lorsque ce travail a été commencé, la recommandation de HTML ne spécifiait pas de tag `object`, cependant son introduction était déjà discutée dans divers documents de travail.

Les objets sont différenciés soit en spécifiant explicitement un type, soit à partir de l'URL, directement en fonction de l'extension de fichier ou alors par le protocole utilisé s'il transmet un format, ce qui est le cas pour RTP ou RTSP par exemple. Il est possible de définir des objets composites par encapsulation de plusieurs objets de types différents.

À terme, nous envisageons d'étendre les propriétés temporelles aux autres tags de HTML. Les interfaces de programmation définies par DOM permettront, une fois implémentées, de manipuler tous les éléments de HTML, et par exemple de leur conférer un comportement temporel. Toutefois, cela pose de

nouveaux problèmes du point de vue de la disposition spatiale, notamment à cause du texte. Les objets graphiques définissent une zone rectangulaire d'affichage dans laquelle ils s'inscrivent, même s'ils sont de forme quelconque. De plus, ils supportent bien d'être mis en mouvement ou partiellement cachés. Ce n'est pas le cas du texte. Comment par exemple réaliser la mise en page d'un texte dont certains des paragraphes seront montrés ou cachés au cours du temps : le flot de texte doit-il être reformaté à chaque événement, doit-on prévoir les emplacements, les paragraphes doivent-ils se superposer ? De nombreuses stratégies sont possibles et doivent être étudiées afin d'évaluer leur intérêt et le réalisme de leur mise en œuvre.

```
< object id    = " name "
         src   = " url "
         type  = " mime-type "
         role  = " master | slave "
         scale = " object-id | layout-id | float value " >
    objects
    synchronization points
    time points
    hypertime links
< /object >
```

L'attribut `scale` permet de spécifier simplement un contrôle temporel sur la présentation de l'objet. Par exemple :

- `+1.5` : présenter une fois et demi plus vite ;
- `-1` : présenter en arrière ;
- `++1` : présenter en boucle ;
- `--2` : présenter en boucle en arrière et à vitesse double ;
- `+-1` : présenter en avant puis en arrière ;
- `-2+1` : présenter en arrière et à vitesse double puis en avant.

Si la valeur de l'attribut `scale` n'est pas un nombre mais l'identificateur d'un autre objet, alors la présentation de l'objet sera adaptée de manière à ce que sa durée soit égale à celle de l'objet référencé, à condition que cette

dernière soit connue. Dans ce cas, le rôle de l'objet n'a pas de sens puisque sa présentation est liée à celle de la référence. Cela peut être utilisé pour qu'une vidéo adapte sa cadence de présentation afin de terminer en même temps que la bande sonore associée, mais l'utilisation la plus intéressante est la possibilité d'adapter la durée de mouvement d'un layout à la présentation d'un autre objet.

Exemples :

- `<object id="video" src="clip.mpg" role="slave" scale="+2">` `...</object>`

  présente `clip.mpg` en tant qu'esclave à deux fois sa vitesse nominale ;

- `<object id="music" src="http://www.imag.fr/music.wav"` `role="master" scale="++1">...</object>`

  joue le fichier audio localisé à `http://www.imag.fr/music.wav` en boucle et en tant que maître ;

- `<object id="logo" src="logo.gif" scale="video">...</object>` présente une image GIF animée en adaptant sa durée à celle de la vidéo de manière à ce qu'ils se terminent ensemble ;

- `<object id="movie">`
  `<object id="see" src="hello.qt">...</object>`
  `<object id="hear" src="hello.au">...</object>`
  liens hypertemps définissant la composition temporelle
  `</object>`

  objet composite contenant une vidéo QuickTime et un fichier audio.

4.7.4 Layout et frame

Le tag `layout` est un objet à part. Il ne spécifie pas de contenu multimédia, mais encapsule des frames définissant des régions graphiques. Ces deux objets particuliers n'ont pas de rôle associé parce qu'ils ne possèdent pas de contenu à proprement parler, ils définissent seulement des portions d'espace graphique. Ils sont par défaut esclaves.

`< layout id` `= "` *name* `"`

```
        scale = " object-id | layout-id | float-value " >
    frames
    synchronization points
    time points
    hypertime links
< /layout >

< frame id    = " name "
        src   = " object-id | layout-id | url "
        layer = " integer value "
        shape = " shape "
        mask  = " mask " >
```

L'attribut `layer` définit la priorité de recouvrement des frames, il est équivalent au `z-index` de CSS2. L'attribut `shape` est utilisé pour décrire la zone graphique couverte par la frame et `mask` définit un masque appliqué à la représentation graphique de l'objet. Cela permet de réaliser des effets comme jouer une vidéo dans une forme triangulaire ou à travers une forme plus complexe définie par un masque, voir la figure 4.12.

Figure 4.12: Exemple d'effets graphiques obtenus avec les frames

Lors de ce travail nous nous sommes concentrés sur les aspects temporels et nous n'avons pas étudié de façon exhaustive quels seraient les moyens les plus appropriés pour représenter les formes et les masques. Des activités spécifiques au graphisme sont actuellement en cours et les recommandations qui en seront issues prochainement fourniront peut-être des solutions adéquates et standard. Une proposition d'amélioration de HTML afin d'y ajouter la possibilité de décrire des effets d'animation a été faite[108] ; cependant, celle-ci repose sur une approche à objets dont les interactions se font par passage de messages qui se révèle assez lourde et complexe. Celle-ci tient également plus d'une approche programmatique que déclarative. Pour les besoins du prototypage nous n'avons retenu que l'attribut `shape` définissant un rectangle sur l'écran spécifié par une liste de deux ou quatre entiers définissant respectivement :

- sa position (x, y), sa taille étant déduite de l'objet qu'il contient ;
- sa position et sa taille (x, y, w, h).

Exemples :

- ```
  <layout id="root" scale="video">
     <frame id="frame1" src="video">
     <frame id="frame2" src="logo">
     <frame id="frame3" src="embeded">
  </layout>
  <layout id="embeded">
     <frame id="frame4" src="movie">
     <frame id="frame5" src="picture">
  </layout>
  ```

  spécifie des layouts imbriqués.

4.7.5 Timebase

Le tag `timebase` sert à grouper les objets et layouts ayant une relation de synchronisation de couplage lors de leur présentation. Le rôle des objets, maître ou esclave, est spécifié dans les objets eux-mêmes.

```
< timebase >
    layouts
    objects
< /timebase >
```

4.7.6 Synchronization point

Les points de synchronisation sont utilisés pour spécifier un couplage de granularité quelconque au sein d'une base temporelle. La syntaxe est similaire à celle des points temporels avec en plus un attribut `period` qui permet de définir en une déclaration une séquence de points de synchronisation périodiques sur un objet.
```

```
< sync-point id     = "  name "
             value  = "  integer value | float value "
             period = "  integer value | float value "
             unit   = "  frame | sample | timestamp | second"
             type   = "  nominal | absolute " >
```

Exemples :

- `<sync-point value="300" unit="frame" type="nominal">`
  tous les objets de cette base temporelle seront synchronisés sur la trois centième image de l'objet dans lequel ce point est défini ;
- `<sync-point period="5" unit="seconds" type="absolute">`
  dans ce cas les objets seront synchronisés toutes les 5 secondes.

4.7.7 Structure d'un document

Finalement, la structure globale d'un document se présente ainsi. Des bases temporelles sont déclarées. Celles-ci contiennent des déclarations d'objets et de layouts. La déclaration des liens hypertemps spécifiant la composition temporelle peut se faire à tout niveau dans le document. Cependant il est préférable de les organiser de la façon suivante :

- les liens spécifiant la composition temporelle interne d'objets composés sont regroupés à l'intérieur de ces derniers ;
- la composition temporelle globale du document est placée après toutes les autres déclarations.

Le début du document est identifié par les liens temporels ne possédant pas d'origine explicitement déclarée. L'origine de la présentation leur est alors associée et ce sont eux qui sont activés pour démarrer la présentation du document multimédia. Voici un exemple de définition de document.

```
<timebase>
   <object id="title" src="title.txt" role="master"
           font="sansserif-bold-30"
```

```
            bgcolor="yellow" fgcolor="red">
         <time-point id="end" value="5" unit="second" type="absolute">
      </object>
      <object id="space" src="audio/spacemusic.au" role="master"
            scale="++1">
      </object>
      <object id="gong" src="audio/gong.au" role="master">
      </object>
      <object id="apple" src="images/apple.gif" role="master">
      </object>
      <object id="aspen1" src="video/aspen1p.mpg" role="master">
      </object>
      <object id="cuckoo" src="audio/cuckoo.au" role="master">
      </object>
      <object id="computer" src="audio/computer.au" role="master"
            scale="++1">
      </object>
      <object id="madcow" src="images/madcow.gif" role="master">
         <time-point id="end" value="5" unit="second" type="absolute">
      </object>
      <object id="blues" src="audio/blues_8.au" role="master">
      </object>
</timebase>

<timebase>
      <!-- vidéo synchronisée avec des sous-titres -->
      <object id="aspen2" src="video/aspen2.mpg" role="master">
      </object>
      <!-- sous-titres en anglais -->
      <object id="english" src="aspen2.en.sbt" role="slave"
            bgcolor="green" fgcolor="blue">
      </object>
      <!-- sous-titres en français -->
      <object id="francais" src="aspen2.fr.sbt" role="slave"
            bgcolor="blue" fgcolor="green">
      </object>
</timebase>

<timebase>
      <layout id="second">
         <frame id="aspen2_f" src="aspen2" shape="0,20">
         <frame id="english_f" src="english" shape="0,0,160,20">
         <frame id="francais_f" src="francais" shape="0,140,160,20">
```

```
   </layout>

   <layout id="first">
      <frame id="apple_f" src="apple" shape="100,10">
      <frame id="madcow_f" src="madcow" shape="50,100">
      <frame id="aspen1_f" src="aspen1" shape="20,160">
      <frame id="embed" src="second" shape="200,140,160,160">
   </layout>

   <layout id="default">
      <frame id="title_f" src="title" shape="50,50">
      <frame id="embed" src="first" shape="50,50,400,300">
   </layout>
</timebase>

<htlink target="title.beg">

<htlink orig="title.beg" target="space.beg">
<htlink orig="title.beg" target="gong.beg">

<htlink orig="title.end" target="apple.beg">
<htlink orig="title.end" target="gong.end">
<htlink orig="title.end" target="space.end">

<htlink orig="apple.beg" target="aspen1.beg">
<htlink orig="apple.beg" target="cuckoo.beg">
<htlink orig="apple.beg" target="computer.beg">

<htlink orig="aspen1.end" target="aspen2.beg">
<htlink orig="aspen1.end" target="madcow.beg">
<htlink orig="aspen1.end" target="cuckoo.end">
<htlink orig="aspen1.end" target="computer.end">

<htlink orig="aspen2.beg" target="english.beg">
<htlink orig="aspen2.beg" target="francais.beg">
<htlink orig="aspen2.beg" target="blues.beg">

<htlink orig="aspen2.end" target="blues.end">

<htlink orig="madcow.end" target="apple.end">
<htlink orig="madcow.end" target="gong.beg">
```

4.7.8 Commentaires

L'étude et la spécification de ces extensions à HTML s'est faite en même temps que la définition et les premiers tests d'implémentation de l'architecture d'exécution qui est présentée dans le chapitre suivant. Cela a eu plusieurs incidences sur l'ensemble du travail.

- Lors de la définition des extensions, la priorité a été de fait donnée aux aspects qu'il a été possible de tester par l'implémentation dans le prototype, qui constituaient également notre centre d'intérêt principal. Il s'agit des fonctionnalités de base de la composition temporelle et de synchronisation, c'est-à-dire les liens temporels et les bases temporelles.

- Par conséquent, la formalisation des extensions à HTML n'est pas achevée, d'abord parce que le prototype reste pour le moment limité et doit encore être étendu pour permettre de valider les nouvelles idées, ensuite parce que les activités de spécification des nouveaux standards du Web sur lesquels ce travail s'appuie sont toujours en cours.

- Toutefois, il résulte de la méthode employée un langage réellement utilisable et facile à mettre en œuvre en utilisant des technologies existantes ou déjà bien avancées dans leur développement. De plus une conception simple et ouverte permet des améliorations tant au niveau du langage qu'au niveau de l'architecture d'exécution et prototype.

4.8 Conclusion et problèmes ouverts

Nous avons présenté les extensions que nous proposons d'ajouter à HTML afin de lui procurer des capacités multimédia. Ce travail ne vise absolument pas à être exhaustif dans le domaine de l'édition et de la représentation de documents multimédia. L'objectif était d'étudier comment le multimédia pouvait être introduit simplement dans l'environnement du World Wide Web, à une époque où aucune approche intégrée du multimédia au niveau du langage n'était disponible. Entre-temps, le Consortium W3 a initié et mené jusqu'à la publication de la recommandation SMIL[113] un groupe de travail sur le multimédia synchronisé.

Notre proposition de HTML temporel et SMIL sont très proches dans le

but qu'ils se fixent, toutefois il existe de nombreux points de divergences — SMIL a été décrit en 3.4.4.

- D'une part, le modèle de synchronisation employé n'est pas le même. Nous avons opté pour un modèle simple et expressif en utilisant les liens temporels. Dans SMIL le modèle choisi est hybride. Il mélange une approche hiérarchique parallèle/séquentiel et une approche par graphe d'instants permettant de surmonter le manque d'expressivité de la première. Il en résulte un modèle peu homogène avec lequel les documents peuvent être représentés par de nombreuses spécifications, notamment celle utilisant uniquement l'approche par instants.

- D'autre part, SMIL sépare complètement la disposition spatiale des interactions temporelles, ce qui empêche la réalisation de *layout effects*, des effets de mouvement associés à la représentation des objets afin de rendre les présentations attractives. Nous n'apportons pas encore de solution descriptive à la spécification du mouvement, mais en considérant le layout comme un objet pouvant prendre part à la synchronisation, il va être possible d'ajouter simplement cette caractéristique une fois qu'un formalisme particulier aura été choisi. Une première approche a été de décrire le mouvement comme un ensemble de lignes droites entre des points. Cependant, cela est trop basique et il faudrait pouvoir prendre en compte des figures plus complexes comme des mouvements circulaires ou en spirale, des facteurs d'accélération, etc.

- Le langage SMIL dépasse largement le cadre de la description de documents puisque, par l'introduction du tag `switch` il propose de résoudre un problème d'une toute autre nature. Ce tag permet de spécifier des alternatives sur des critères liés au système client présentant le document, comme par exemple le débit réseau disponible, la langue ou les capacités d'affichage. Nous n'avons pas considéré ces problèmes comme étant du ressort de la description de documents multimédia mais de celui des systèmes et des communications, au travers de négociations. Ils seront abordés dans le chapitre suivant.

La garantie de la cohérence n'a pas été retenue comme caractéristique du langage. D'abord parce que cela aurait été trop restrictif et aurait produit un langage plus limité dans ses possibilités. Ensuite, il s'agit d'un problème lié à l'édition et à la présentation mais pas à la description des documents. Enfin, ce n'était déjà pas la philosophie de HTML, avec lequel il est possible

de créer des documents parfaitement incohérents tant du point de vue de la structure que de la syntaxe. Toutefois, le formalisme choisi est similaire à celui utilisé par de nombreux systèmes assurant la cohérence, il est donc possible d'appliquer la théorie et les algorithmes qui leur sont liés pour résoudre ce problème dans ce cas particulier.

Reste le problème de la complexité atteinte par les documents multimédia volumineux. Celui-ci n'est pas lié particulièrement au format que nous avons choisi mais concerne tous les types de description. Il est certain que l'édition de documents multimédia doit se faire à l'aide d'un éditeur adapté fournissant à l'auteur des abstractions de plus haut niveau que celles fournies par un format de représentation, ainsi qu'une aide à l'édition, par exemple par des outils de vérification de la cohérence.

Pour conclure, voici les deux aspects du langage sur lesquels nous travaillons et qui doivent encore évoluer. L'utilisation du nouveau langage XML devrait permettre de proposer des solutions satisfaisantes, cependant cela implique que HTML devienne une application de XML, ce qui devrait se produire assez prochainement.

- La modularité de la représentation doit être augmentée en étudiant deux approches :
  - l'encapsulation au niveau description des contraintes de couplage représentées par les bases temporelles et de la disposition spatiale ;
  - la réutilisation par référence de portions de présentations existantes, en utilisant par exemple XPointer et XLink actuellement spécifiés par le W3C.
- La spécification précise des méthodes à employer pour décrire et synchroniser le mouvement est nécessaire. Cette partie requiert une étude approfondie des méthodes développées dans d'autres domaines comme la réalité virtuelle, en particulier VRML, qui est également un langage descriptif.

Chapitre 5

Architecture d'exécution

5.1 Support de présentation

La démarche suivie lors de ce travail de thèse a été d'étudier et de proposer des extensions nécessaires à l'introduction du multimédia dans l'environnement du Web. Comme il a déjà été expliqué plus tôt dans ce document, le travail sur les niveaux description et présentation s'est effectué conjointement. L'objectif à atteindre était, d'une part, de proposer des extensions à HTML qui soient expressives dans le cadre de la description de documents multimédia mais limitées en nombre pour ne pas remettre en cause le langage, d'autre part, de fournir un environnement d'exécution à ces extensions afin de les valider.

Nous avons particulièrement veillé à ne pas définir un système fermé. Les extensions de spécification de la composition temporelle que nous avons définies sont basées sur un modèle temporel et un modèle de synchronisation courants dans le domaine de la modélisation multimédia. Ces modèles ont été étendus pour répondre à nos besoins en matière de spécification du couplage entre objets. Ces extensions peuvent donc facilement être utilisées dans un cadre différent de celui défini par l'architecture d'exécution de présentations que nous allons décrire maintenant. De façon symétrique, des techniques développées dans un contexte différent peuvent être réutilisées, par exemple en ce qui concerne le contrôle et la garantie de la cohérence.

De même, les concepts de base de l'architecture d'exécution sont directement issus des techniques de programmation objet et des modèles à

événements. Cette architecture n'est donc pas spécifiquement liée au langage pour lequel elle a été conçue au départ. On peut noter que celle-ci est destinée au contrôle et à la synchronisation d'objets multimédia et que son utilisation dans le cadre de la présentation de documents multimédia ne constitue qu'une application particulière.

5.1.1 Principes

La conception de l'architecture d'exécution devant servir de support aux documents multimédia sur le Web a été conçue avec les caractéristiques suivantes dans l'optique de répondre aux contraintes imposées par l'environnement du Web.

- L'architecture est *extensible* afin de pouvoir profiter à faible coût de l'évolution rapide du Web. Cela doit-être pris en compte lors de la conception, mais également lors de l'implémentation, par l'utilisation d'un langage et de techniques de programmation adaptés. Le langage choisi est naturellement Java puisqu'il a été conçu pour le Web et offre donc énormément d'avantages, dont la portabilité n'est pas le moindre.

- L'architecture est également *légère*. Nous entendons par là qu'elle doit fournir le support nécessaire et suffisant à l'exécution de présentations en réutilisant au maximum les composants déjà existants comme les browsers Web et les décodeurs de médias. Dans cette optique, son rôle principal est la gestion et le contrôle de la synchronisation. Cela réduit d'autant la taille de l'implémentation et permet de pallier un inconvénient dont Java, le langage d'implémentation, est la source : les faibles performances. Toutefois, ce langage étant très récent, il nous est également paru logique de compter sur une amélioration rapide de celles-ci.

Les fonctionnalités devant être apportées par l'architecture d'exécution se résument donc au contrôle du temps et de la synchronisation afin de gérer un ensemble d'objets multimédia hétérogènes. Son rôle est de maintenir le déroulement des présentations dans un état de cohérence avec leur spécification temporelle. Les trois aspects de la synchronisation multimédia ont été décrits en 2.2.2.

- La *synchronisation intra-média* consiste à présenter au moment attendu la séquence d'unités de présentation constituant un objet multimédia. Elle est définie implicitement par le taux d'échantillonnage pour les médias continus comme la vidéo et l'audio, ou explicitement par rapport à un média auxiliaire, comme c'est le cas pour les sous-titres d'une vidéo.

- Le *couplage inter-média* assure que les relations temporelles définies entre des objets distincts sont bien maintenues tout au long de leur présentation. Il maintient la vitesse de présentation relative entre les médias, notamment lorsque le comportement de l'un est défini explicitement par rapport à un autre.

- L'*ordonnancement* organise le contrôle des objets afin d'assurer que les événements attendus, démarrage et arrêt, se produisent en accord avec la composition temporelle spécifiée dans le document.

L'approche initialement choisie était d'utiliser des objets actifs pour représenter les différents objets multimédia en interaction, comme c'est le cas dans *PREMO*[42, 43] et dans *ATOM*[84]. Celle-ci repose sur un modèle à événements permettant aux objets d'interagir les uns avec les autres. Cependant, notre but étant uniquement la gestion de la synchronisation, un modèle plus spécialisé et plus léger que ces derniers était suffisant et surtout nécessaire afin de ne pas trop alourdir l'implémentation.

La réutilisation des modèles existants posait plusieurs difficultés. D'abord, comme nous l'avons vu leur trop grande généralité est gênante pour notre cas particulier. Afin de les utiliser il aurait fallu en définir des sous-ensembles en éliminant les aspects inutiles. Ensuite, soit ils sont toujours à l'état de spécification, soit leur implémentation ne correspond pas aux critères nécessaires à leur utilisation dans l'environnement du Web. Cela n'épargnait donc pas un travail d'implémentation. Nous avons donc choisi de concevoir une architecture d'exécution en parallèle avec la spécification du langage, chacun profitant de l'avancement de l'autre, et en s'appuyant sur les technologies Web en développement qui sont déjà ou qui apparaîtront bientôt dans tous les browsers.

5.1.2 Composants

Les extensions à HTML que nous avons proposé reposent sur des abstractions de niveau suffisamment bas pour que l'on puisse retrouver presque directement leur contrepartie dans l'architecture d'exécution. La figure 5.1 présente une vue structurelle simplifiée de l'architecture d'exécution. Les principaux éléments en sont les objets synchronisables, les gestionnaires de synchronisation et les événements de synchronisation.

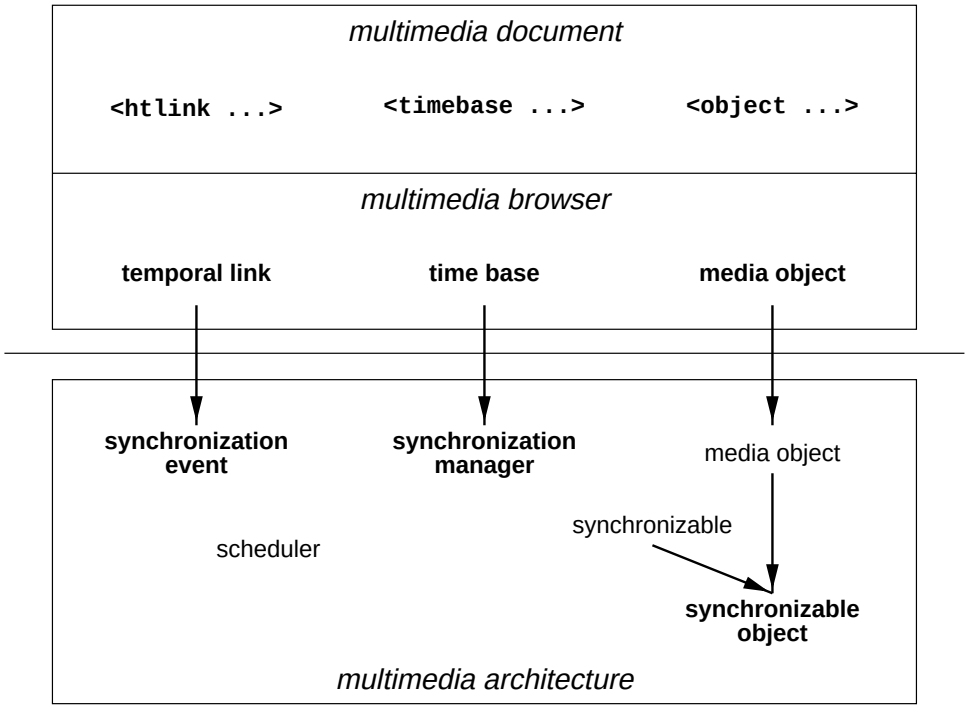


Figure 5.1: Vue structurelle de l'architecture d'exécution

Dans le cas qui nous intéresse, les objets multimédia ne sont pas autonomes comme les objets actifs au sens strict définis dans les modèles cités auparavant peuvent l'être. Premièrement, leur comportement est soumis au déroulement du temps, ce qui se traduit par la génération d'événements à des instants bien définis. À cette fin, un *scheduler* prend en charge la gestion globale du temps et des événements.

Deuxièmement, la spécification temporelle d'un document multimédia fait apparaître des relations de couplage entre des objets dont les rôles peuvent être maître ou esclave. Les esclaves sont dépendants des maîtres, et les maîtres eux-mêmes peuvent être soumis au contrôle d'un autre maître (voir en 4.4.2).

Aucun objet n'est autonome, tant vis-à-vis du temps que des autres objets. Une entité tierce est donc nécessaire afin de contrôler globalement les objets définis dans une même base temporelle : le gestionnaire de synchronisation ou *synchronisation manager*.

Nous allons maintenant présenter en détail chacun des composants de l'architecture de synchronisation dont on peut voir les relations fonctionnelles illustrées sur la figure 5.3.

Synchronizable objects

Les objets synchronisables sont les objets multimédia de base manipulés par l'architecture de synchronisation. Ils encapsulent tout le traitement nécessaire à la présentation synchronisée des médias et à leur contrôle. Une implémentation modulaire est nécessaire pour disposer d'un environnement flexible et ouvert. Pour permettre la réutilisation et l'intégration aisée de composants externes, comme des décodeurs pour certains types de médias, les fonctionnalités de transport et de traitement des données numériques doivent être clairement séparées de la synchronisation. Du point de vue de l'architecture de synchronisation, un objet synchronisable est composé de deux modules : un premier encapsulant tous les traitements spécifiques au média traité et un second fournissant une interface de synchronisation par laquelle le comportement temporel de l'objet peut être contrôlé.

Les *media objects* sont des entités autonomes utilisées comme composants de base dans la construction de documents multimédia. Ils regroupent les fonctions de transport, traitement et affichage des données dont ils ont la charge. Afin de proposer des présentations de qualité optimale, ils doivent également contrôler la qualité de service tout au long de ce processus. Nous ne nous intéressons pas à ces fonctionnalités dans le cadre de la synchronisation, toutefois nous reviendrons plus loin sur les problèmes posés par la gestion de multiples objets sur le Web et le contrôle global qu'il est nécessaire d'exercer afin de garantir des présentations optimales dans des conditions variées.

Les media objects doivent offrir un moyen de contrôle sur leur déroulement temporel interne par l'intermédiaire d'une interface que nous appelons *synchronizable*. Dans la terminologie Java, une interface est un ensemble de méthodes sans implémentation, que tout objet peut implémenter pour devenir du type de cette interface. Un *synchronizable object*, quel qu'il soit, peut être contrôlé et synchronisé par l'architecture de synchronisation, pour

peu qu'il implémente correctement les points d'entrée nécessaires : il doit générer des événements de synchronisation et permettre son contrôle par les gestionnaires de synchronisation.

Nous avons envisagé l'utilisation d'objets se présentant sous la forme de code exécutable et Java se prête bien à ce rôle. Ce langage étant portable et sécurisé, il y a alors moyen d'exécuter des programmes sans courir de risque supplémentaire par rapport aux médias traditionnels. Une utilisation originale de cette caractéristique est la création d'opérateurs simples. La figure 5.2 illustre un opérateur aléatoire déclenchant au hasard un seul des objets qui lui sont consécutifs.

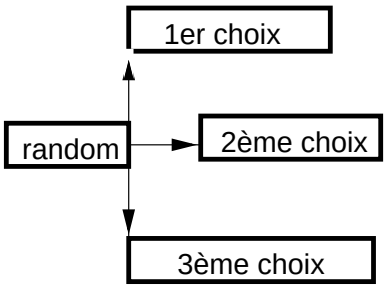


Figure 5.2: Un opérateur aléatoire

Synchronization events

Les événements de synchronisation transmettent l'information temporelle entre les différentes entités concernées, c'est-à-dire le scheduler, les gestionnaires de synchronisation et les objets synchronisables. Un événement de synchronisation est défini par les éléments suivants :

- un émetteur ou source : l'objet ayant généré l'événement en question ;
- un destinataire : l'objet auquel cet événement est destiné, qui peut être le même que la source ;
- une action, devant se produire sur la destination lorsque cet événement est activé ;
- une date, spécifiant quand l'événement doit être activé et ainsi son action associée déclenchée.

Lorsque la date de l'événement est atteinte, celui-ci est activé par le scheduler et envoyé au gestionnaire de synchronisation associé à l'objet destinataire afin qu'il exécute l'action associée. Les événements de synchronisation sont utilisés à la fois pour la synchronisation intra-média et inter-média. Un lien temporel peut être directement représenté par un événement de synchronisation, ils possèdent les mêmes éléments de définition : une origine, une destination, une action, définie implicitement dans le cas du lien, et une date d'activation. La clé du problème est de pouvoir représenter cette date de façon relative et non absolue, ce qui est rendu possible par la génération dynamique des événements expliquée en 5.1.4.

Synchronization managers

Les gestionnaires de synchronisation matérialisent les bases temporelles. Chacune d'elles est représentée au niveau exécution par un *synchronization manager* dont le rôle est de gérer les événements de synchronisation qui lui sont transmis par les objets et de mettre en application la politique de synchronisation définie par les rôles maître ou esclave. Les managers n'ont pas directement accès au temps mais transmettent les événements à un scheduler auquel ils demandent l'exécution des opérations temporelles requises.

Chaque synchronization manager centralise la gestion des événements des objets appartenant à une base temporelle donnée. Les événements étant contrôlés par une seule entité, cela permet de gérer leur positionnement relatif de manière optimale afin d'assurer le maintien de la cohérence temporelle relative entre les objets d'une même base temporelle. Cette gestion est alors plus rapide que par l'utilisation d'objets actifs devant s'échanger perpétuellement des informations afin que chacun se positionne relativement par rapport aux autres.

Scheduler

Le tâche du *scheduler* est de centraliser la gestion du temps. Celui-ci reçoit les événements depuis les managers, les place en attente et signale le manager concerné lorsqu'un événement a atteint sa date d'activation, *deadline*. La centralisation de la gestion du temps est nécessaire, car la seule notion de temps disponible au niveau système est le temps machine fourni par l'horloge interne et celui-ci est évidemment absolu. Afin d'ordonner les événements de

manière relative, il faut centraliser l'information temporelle dans un endroit pour pouvoir transformer les valeurs relatives en valeurs absolues et inversement.

5.1.3 Principe de fonctionnement

Le principe de fonctionnement a déjà été un peu entrevu grâce à la description du comportement des principaux éléments de l'architecture de synchronisation, dont la figure 5.3 présente une vue fonctionnelle. Nous allons décrire maintenant comment sont effectués la synchronisation intra-média et inter-média — couplage et ordonnancement.

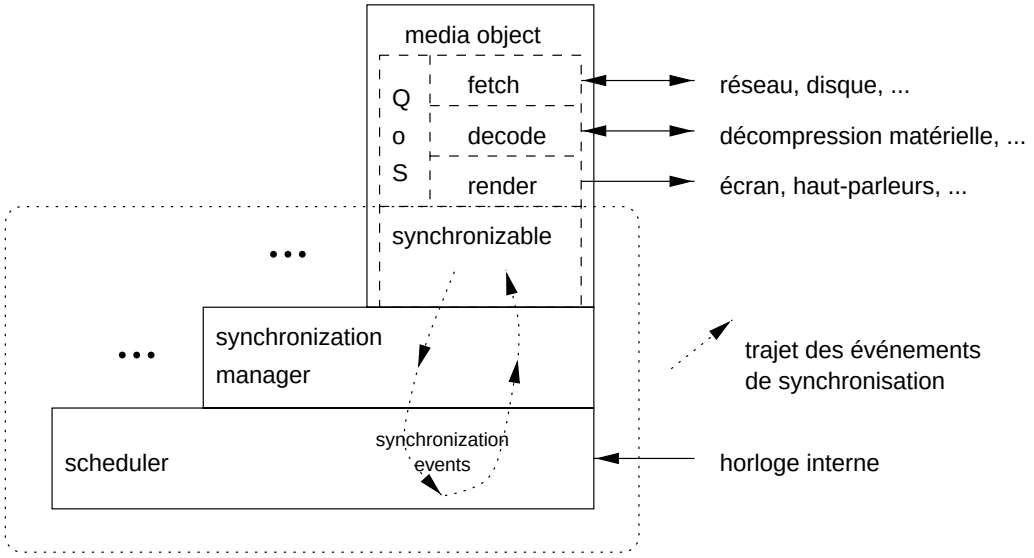


Figure 5.3: Architecture de synchronisation

Ordonnancement

L'ordonnancement des objets multimédia est réalisé simplement avec les événements de synchronisation. La composition temporelle spécifiée à l'aide de liens hypertemps définit l'ordonnancement. Il a été vu plus haut que les liens hypertemps sont directement représentés par des événements de synchronisation lors de la présentation. Un objet peut associer à un événement le concernant un autre événement destiné à un second objet et provoquant

l'exécution de l'action désirée. Vu à un niveau élevé, l'ordonnancement apparaît comme une propagation en chaîne d'événements.

Synchronisation intra-média

La synchronisation intra-média est réalisée par l'utilisation de séquences d'événements. Chaque objet synchronisable peut générer des événements de synchronisation comme bon lui semble pour ses propres besoins. Ces événements lui sont simplement destinés, et lui reviendront une fois la date d'activation atteinte. Le chemin emprunté par les événements est décrit sur la figure 5.3.

De cette manière tout objet peut simuler sa propre horloge et l'utiliser pour sa synchronisation interne. Notre travail s'arrête à ce niveau, c'est-à-dire fournir un support temporel aux objets multimédia. Les méthodes de synchronisation internes aux objets sont en général dépendantes des types de médias, de codage et de transport utilisés et doivent donc être implémentés en leur sein : réservation de buffers, anticipation du décodage, gestion de délais de garde, etc.

Par exemple, une vidéo jouant 25 images par seconde générera une séquence d'événements espacés de 40 millisecondes. L'objet vidéo sera alors signalé à chaque fois qu'il devra afficher une nouvelle image.

Couplage inter-média

D'après le modèle de composition temporelle, le couplage inter-média est réalisé au niveau des bases temporelles en utilisant les points de synchronisation. Dans l'architecture de synchronisation, le rôle des bases temporelles est joué par les gestionnaires de synchronisation. Tous les objets définis dans une base temporelle commune sont couplés à la granularité définie par les points de synchronisation.

Les événements de synchronisation peuvent être marqués comme étant des points de synchronisation. Il est en effet possible de resynchroniser des objets uniquement lorsqu'une action doit être effectuée. Lorsqu'aucun point de synchronisation n'est défini, tous les événements sont marqués, la synchronisation est alors réalisée au grain le plus fin possible. Pour garantir le couplage inter-média les gestionnaires de synchronisation contrôlent et modifient

la position relative des événements en fonction du rôle de leur destinataire. Ce mécanisme est décrit en détail dans la section suivante.

5.1.4 Génération dynamique des événements

Le fonctionnement de l'architecture de synchronisation repose sur la génération dynamique des événements de synchronisation. Le principe de l'algorithme est le suivant : un objet génère chaque événement lorsqu'il est nécessaire et relativement à l'événement précédent. Cette approche comporte plusieurs avantages.

- Il résulte de ce principe de génération des événements qu'il y a au maximum un événement par objet dans le système. Les événements étant traités de manière centralisée au sein des gestionnaires de synchronisation et du scheduler, cela réduit la taille des files d'éléments à manipuler et à trier.
- La gestion de la synchronisation et de l'indéterminisme, qui consiste à contrôler et ajuster la position relative des événements, se fait de façon transparente. Toute modification temporelle apportée à un événement est automatiquement répercutée sur les suivants puisqu'ils seront générés après. Aucune propagation de correction ou réévaluation de contraintes n'est donc nécessaire au sein des objets.
- La synchronisation intra-média, l'ordonnancement et le couplage inter-média sont réalisés au moyen d'un seul mécanisme à événements apportant ainsi une grande cohérence dans la gestion du temps et des multiples objets composant un document.
- La gestion centralisée des événements permet de diminuer le coût causé par les opérations qui leur sont appliquées. Cela évite les processus de communication complexes et coûteux entre les objets.

La gestion de la synchronisation consiste à manipuler des événements afin de garantir une position relative entre ceux-ci et leur activation au bon moment. De ce point de vue, l'unique paramètre caractérisant un événement est sa date d'activation ou *deadline*. Nous définissons celle-ci à partir d'une date de création et d'un délai, on a alors : $e_i.deadline = e_i.creationTime + e_i.delay$.

Les événements d'un objet sont générés dynamiquement par rapport au précédent événement de cet objet. Une séquence d'événements est définie ainsi : lorsque e_i est activé alors e_{i+1} est généré et $e_{i+1}.creationTime = e_i.deadline$. Un objet continu de fréquence fixe comme une vidéo jouant 25 images par seconde génère une séquence d'événements de délai fixe égal à 40 millisecondes. Cette technique présente deux avantages majeurs.

- D'abord, les petites variations de temps dans l'activation des événements n'ont pas d'effet sur une séquence d'événements puisque chacun d'eux est généré en fonction de la date théorique de déclenchement $e_i.deadline$ de son prédécesseur et non par rapport à sa date réelle. Cela permet d'éviter les phénomènes de dérive, ou *drift*, rencontrés sur les systèmes classiques non temps réel et causés par le délai variable et inconnu qui existe entre le moment où la date d'activation d'un événement est atteinte et son activation réelle. L'accumulation de tels délais cause la dérive.

- Enfin, si un événement est modifié pour les besoins du maintien de la synchronisation, cette modification est répercutée de façon transparente à tous ses successeurs. De même pour le changement de cadence de restitution d'un média. Il suffit, pour ralentir la cadence d'une vidéo jouant 25 images par seconde de moitié, de changer le délai initial de 40 millisecondes en 80 millisecondes.

Nous allons maintenant décrire le principe utilisé afin de maintenir le couplage entre des objets maîtres et esclaves au sein d'une même base temporelle. La figure 5.4 illustre ce propos. Les événements sont représentés sur la ligne du temps en fonction de leur date d'activation. La figure 5.4a représente des événements activés sans aucun problème. Il y a toujours un délai entre l'instant où un événement est activé et l'instant où l'action correspondante est exécutée. Quand aucune garantie temps réel n'est assurée, ce délai n'est pas borné. Dans ce cas, les objets en charge de la présentation de médias sensibles comme l'audio doivent utiliser des techniques de *buffering* pour limiter l'impact de ces petites variations.

Supposons maintenant que pour une raison quelconque — décodeur trop lent, problème réseau, etc. — l'objet destinataire de l'événement ne puisse pas exécuter l'action prévue. Il y a alors deux solutions, correspondant aux deux rôles possibles pour un objet. Si le destinataire est un esclave, il ne peut pas influencer le déroulement du temps, il doit donc continuer à progresser dans le

temps sans attendre que l'action soit exécutée. Les événements suivants vont être annulés jusqu'à ce que l'objet reprenne son déroulement normal, c'est-à-dire que les actions puissent être exécutées à leur instant correct, figure 5.4b. Le cas d'un objet maître est présenté sur la figure 5.4c. Si l'action prévue ne peut être exécutée immédiatement, l'objet peut attendre son exécution ultérieure. L'événement est donc observé après un délai Δ par rapport à sa date d'activation initiale. L'événement suivant sera alors généré en lui ajoutant ce délai supplémentaire pour le "recaler".

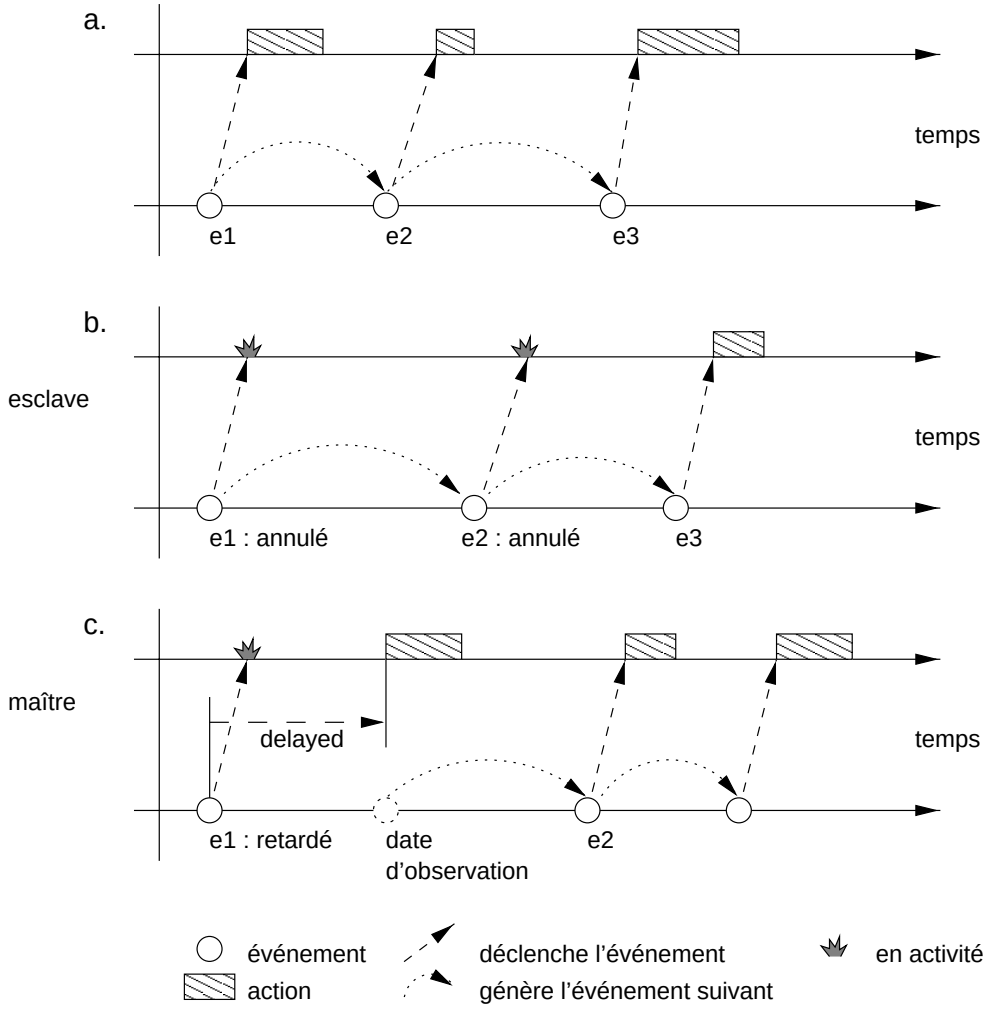


Figure 5.4: Ordonnancement des événements de synchronisation

Ces traitements ne suffisent pas parce que nous avons uniquement considéré un objet isolé. Lorsque plusieurs objets sont couplés, d'autres traitements doivent être réalisés. Dans le cas où l'objet est un esclave, des actions

sont annulées mais le déroulement temporel est inchangé. Par contre, dans le cas d'un maître, le problème est plus complexe (voir la figure 5.5) : il faut conserver la relation temporelle entre tous les objets sous la responsabilité du gestionnaire de synchronisation. Le délai Δ doit être ajouté à tous les événements. Dans ce cas, un processus complexe est nécessaire car plusieurs cas sont possibles pour chaque événement : il peut avoir juste été généré ou activé, être en attente dans le gestionnaire ou dans le scheduler. Les gestionnaires de synchronisation sont en charge de cette tâche car ils voient passer tous les événements et ce sont les seuls à avoir une vue globale comme le montre la figure 5.3.

La figure 5.5 est un peu complexe mais illustre la chronologie des actions effectuées pour maintenir la cohérence entre un maître et un esclave liés par une relation de couplage.

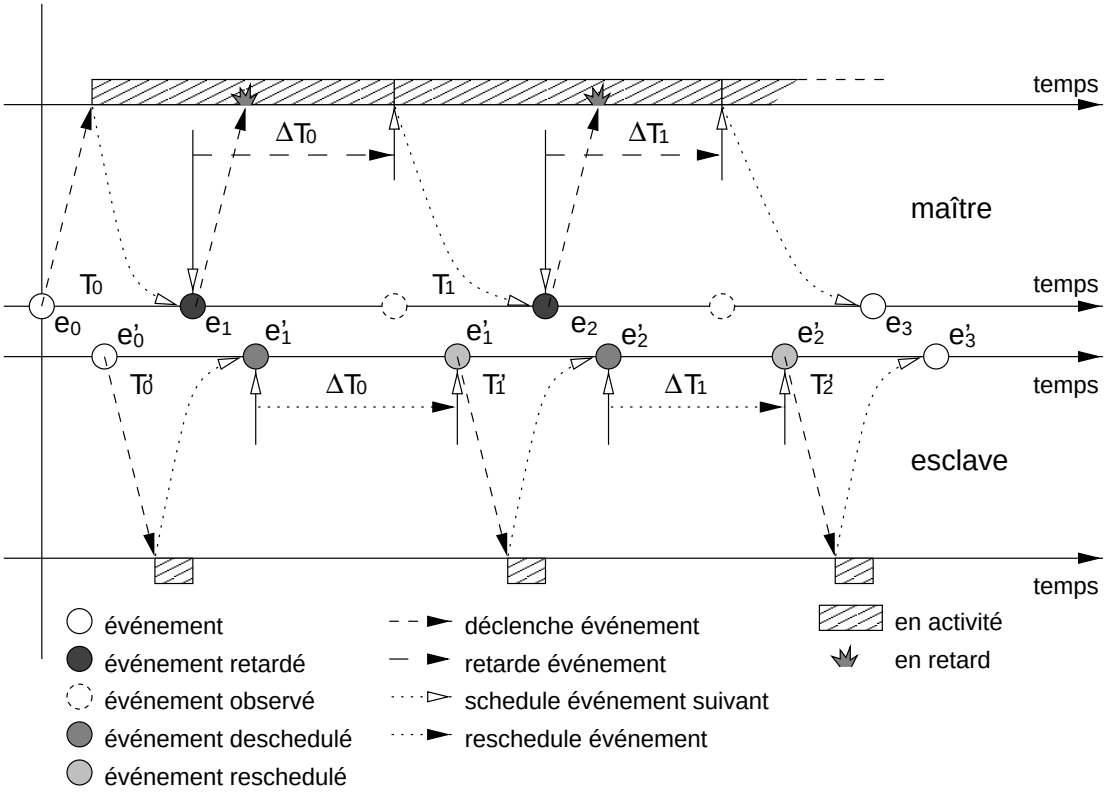


Figure 5.5: Maintien de la cohérence entre événements

5.1.5 Conclusion

L'architecture de synchronisation proposée ci-dessus permet de contrôler la synchronisation des objets constituant un document multimédia en accord avec sa spécification. Le principe de génération dynamique des événements et l'algorithme de contrôle de ceux-ci fournit la souplesse nécessaire pour gérer l'indéterminisme qui est inhérent à l'environnement du Web. La possibilité de dilater le temps garantit que la position relative des objets d'une même base temporelle est maintenue en toute circonstance, ainsi l'état d'un document est toujours conforme à sa spécification et lui permet de supporter des perturbations passagères sans dommages : la dégradation de la présentation se fait de manière cohérente. Bien sûr, ce mécanisme n'est qu'un dernier recours et ne constitue pas un moyen de contrôle souhaitable. L'implémentation des différents objets et des protocoles de transport des données multimédia doit comporter des mécanismes de contrôle de qualité de service adaptatifs afin de prévenir l'apparition de tels problèmes et d'éviter l'introduction de délais qui dégradent la qualité de présentation globale.

Des techniques de synchronisation par événements similaires dans le principe ont déjà été développées, cependant, elles ne sont pas adaptées au modèle de synchronisation défini par le langage que nous avons proposé. Un système utilisant des événements globaux[70] permet de synchroniser plusieurs flots, chacun de ceux-ci contenant les mêmes événements qui doivent être mis en correspondance lors de la restitution afin d'assurer la synchronisation. Flinn a proposé un mécanisme de contrôle d'effets sonores[32] dans lequel les événements sont gérés d'une manière similaire à celle de notre proposition. Un scheduler distribue des événements ou des séquences d'événements à des applications en ayant fait la demande et intègre un mécanisme permettant une dégradation de qualité élégante. Cependant, les événements sont gérés de manière statique sous la forme de collections et les solutions proposés sont orientées vers la synchronisation de phase et de motifs rythmiques. Il existe également d'autres travaux propres au domaine musical[81], mais qui sont trop spécifiques. De nombreuses autres techniques de synchronisation[28] ont été proposées et certaines essaient de fournir des méthodes assez formelles de calcul prenant en compte les délais introduits par les réseaux ou la charge des machines[89, 105].

La structure modulaire de l'architecture de présentation nous permet de prévoir son extension future. Il est possible de l'étendre de nombreuses manières. D'abord, en créant de nouvelles classes d'objets synchronisables à partir de types de médias dont le support existe déjà. Nous envisageons

également l'ajout de fonctionnalités de contrôle de qualité de service destinées aux objets multimédia ainsi que la création d'objets particuliers de type programme ne produisant aucun média mais implémentant des opérateurs.

5.2 Le prototype MADCOW

MADCOW, pour *Multimedia Architecture for Distribution of Content Over the Web*, est un prototype de l'architecture proposée. Un module de traitement des extensions à HTML a également été réalisé. L'implémentation a été effectuée dans le langage Java.

5.2.1 Le langage Java

Le langage Java a été créé par *Sun Microsystems* : il est apparu en 1995. Ce langage possède plusieurs particularités qui en font un langage de choix dans le cadre du développement d'applications pour le Web car il a été conçu précisément dans cette optique.

Java est un langage orienté objet compilé dans un format de *bytecode* et ensuite interprété sur une machine virtuelle. Le langage est syntaxiquement proche de C et C++ mais beaucoup plus épuré que ce dernier du point de vue de l'approche objet. Java est très complet et a été conçu pour être robuste, ce qui se retrouve à plusieurs niveaux.

- Il ne supporte ni l'héritage multiple ni la surcharge des opérateurs.
- Les fonctions n'existent pas, seules des méthodes appartenant à des objets peuvent être définies.
- Les pointeurs n'existent pas, seules les références sur les objets.
- Un mécanisme de gestion des exceptions est fourni au niveau du langage.
- La gestion de la mémoire est automatique et assurée par un *garbage collector* dans la machine virtuelle.

- Le parallélisme est assuré au niveau langage par la gestion des processus légers, ou *threads*, et la mise à disposition de mécanismes d'exclusion et de synchronisation.

- Les classes sont regroupées en *packages* ce qui garantit une organisation et une identification claire des différentes bibliothèques. Par exemple, le package `java.lang` regroupe les primitives du langage et `java.net` ce qui concerne le réseau.

- Enfin les interfaces de base, API, sont très complètes et fournissent en standard les fonctions d'entrée/sorties, graphique et réseau, entre autres.

De plus, Java est un langage sécurisé. L'interpréteur effectue des contrôles statiques sur le bytecode chargé pour vérifier que les opérations sur la pile, les types et les branchements sont bien valides. Lors de l'exécution, de nouveaux contrôles sont réalisés sur les classes chargées et sur l'accès aux ressources.

Nous utilisons deux caractéristiques du langage Java permettant de définir proprement une architecture comme celle que nous proposons en la rendant extensible : ce sont les interfaces et les classes abstraites. Une classe abstraite est une classe dont certaines méthodes sont uniquement déclarées sous la forme de leur prototype et ne sont pas implémentées. Une interface est semblable et n'est constituée que de prototypes de méthodes sans implémentation. Cela permet, dans le cas de notre architecture, de mettre à disposition une interface `Synchronizable` qu'il est possible d'implémenter dans toute autre classe. Ainsi, des classes existantes peuvent être adaptées à notre proposition d'architecture et utilisées sans réimplémentation, seulement en les sous-classant en une nouvelle version implémentant l'interface en question.

Une autre caractéristique nous concernant directement est l'apparition du concept d'*applet*. Une applet est une application spécialement développée pour s'exécuter dans un environnement embarqué dans un browser Web. Celle-ci est incluse par référence dans un fichier HTML à l'aide d'un nouveau tag `applet` créé pour l'occasion. Lorsqu'une page Web contenant une applet est visualisée dans un browser, celui-ci charge son bytecode et l'exécute localement sur son interpréteur. L'accès aux ressources locales et réseau est contraint par l'environnement d'exécution afin qu'une applet ne se transforme pas en virus. C'est cette forme d'application que nous allons utiliser pour présenter des documents multimédia sur le Web.

Comme nous l'avons déjà dit, les API de base de Java sont très complètes, ce qui est un énorme avantage pour réaliser rapidement un prototype.

- Les threads permettent de rapidement mettre en œuvre des tâches concurrentes au sein d'une même application, ce qui est appréciable pour les applications multimédia où elles cohabitent généralement en grand nombre.

- La gestion graphique de base facilite le développement des interfaces graphiques et la manipulation des objets tels que les images et le texte. Cela se fait de manière transparente vis-à-vis du type et des capacités de la plate-forme.

- Les protocoles réseau de base sont présents et leur utilisation est aisée. Notamment, l'accès à des ressources sur le Web se fait simplement à partir de leur URL.

Toutefois, lorsque nous avons commencé ce travail, il n'existait pas de support pour le traitement des données multimédia, vidéo et son en particulier. Aujourd'hui, certaines fonctionnalités sont disponibles via le *Java Media Framework*, cependant ce dernier n'offre pas le contrôle suffisant qui nous est nécessaire, ni le code source qui permettrait de contourner cette limitation.

Pour conclure, nous devons aborder les limites de Java. D'abord, il s'agit d'un langage jeune dont les interfaces de programmation de base sont pour la plupart stables mais pas toujours bien définies et dont l'implémentation est parfois peu fiable. Le code source de nombreuses bibliothèques est disponible, ce qui à notamment permis de corriger plusieurs *bugs* dans la partie graphique qui étaient la cause d'une qualité de présentation médiocre (mauvais positionnement, clignotement). Ensuite, c'est un langage interprété ce qui implique des performances assez faibles dans le cas de traitements intensifs comme ceux nécessaires au décodage de certains médias par exemple. Deux solutions sont possibles, soit le recours à du code natif interfacé avec Java qui provoque alors la perte de la portabilité, soit l'utilisation d'un *JIT-compiler*, compilateur juste à temps, qui compile le bytecode en code natif avant exécution préservant ainsi la portabilité. Ces techniques permettent d'obtenir des résultats juste satisfaisants pour de petites présentations.

5.2.2 Conception du prototype

Deux versions du prototype ont été conçues mais elles reposent toutes les deux sur le même principe. Les extensions multimédia à HTML que nous avons proposées ne sont évidemment pas présentes dans les browsers Web. Il a donc fallu trouver une méthode afin de les expérimenter. À l'époque où ce travail a été débuté, aucun des éditeurs des principaux browsers susceptibles de nous intéresser, *Netscape Communicator*, *Microsoft Internet Explorer* et *Sun HotJava*, ne fournissait le code source de son logiciel qui nous aurait permis de le modifier. Même si cela avait été le cas, il aurait fallu un travail considérable pour analyser le code volumineux, déterminer tous les endroits où apporter des modifications et finalement les réaliser.

Dans un premier temps, nous avons finalement choisi la méthode suivante. Puisque nous sommes intéressés par les extensions multimédia à HTML, nous avons décidé de ne considérer que celles-ci et de réaliser un premier prototype capable de transformer une spécification de document en une applet s'exécutant au dessus de l'architecture de synchronisation définie plus tôt. Un document multimédia peut alors être présenté dans un browser supportant Java, cependant, il s'agit d'une méthode un peu artificielle puisque le document est en fait constitué uniquement d'une applet. La figure 5.6 illustre cette première version.

Dans un second temps, une fois ce premier prototype réalisé nous l'avons modifié afin de produire à partir d'un document multimédia un second document en HTML standard dans lequel des applets sont utilisées, voir figure 5.7. Celles-ci implémentent certains objets multimédia, interfacent des éléments HTML par l'intermédiaire du modèle objet de document et gèrent la synchronisation toujours au-dessus de l'architecture de synchronisation.

5.2.3 Architecture de synchronisation

L'architecture de synchronisation a été implémentée sous la forme d'un package `osf.sync` en accord avec la définition qui en a été donnée plus haut dans ce chapitre. La figure 5.9 en montre la hiérarchie de classes. Il y a un objet `Scheduler` par présentation. Celui-ci est exécuté dans un thread de priorité maximale et est le seul dans ce cas. Il est ainsi garanti de pouvoir traiter au plus tôt les événements lorsque ceux-ci doivent être activés. La figure 5.8 illustre le principe de fonctionnement du scheduler. Elle montre

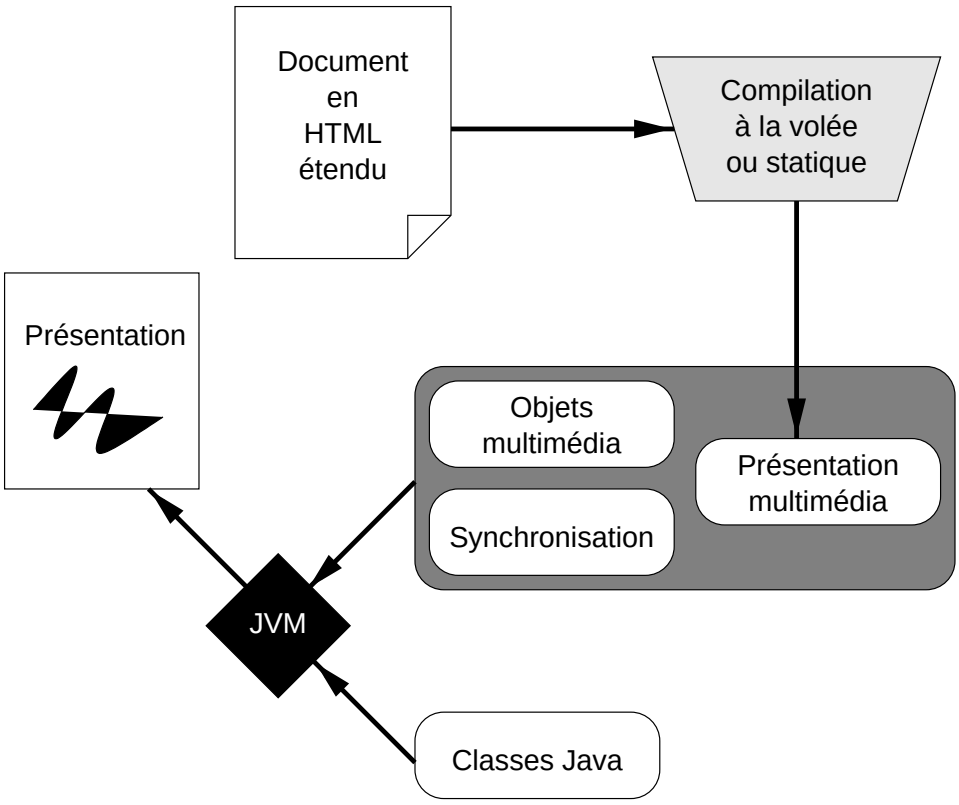


Figure 5.6: Conception du premier prototype

la progression de l'état du scheduler dans le temps au fur et à mesure que des événements sont schedulés — ajoutés au scheduler. Les événements sont placés dans une file, dans l'ordre de leur date d'activation. Le délai séparant l'instant courant de la date d'activation du premier événement est calculé et le scheduler s'endort pour cette période. Il est réveillé, soit par la fin de ce délai, soit par l'ajout d'un événement venant s'insérer devant le premier de la file.

`SyncManager` implémente les gestionnaires de synchronisation. Ceux-ci sont de priorité immédiatement inférieure au scheduler et tous les objets leur sont de priorité inférieure ce qui permet aux managers de les préempter si nécessaire. Nous pouvons faire remarquer ici que la spécification de la machine virtuelle Java, JVM, est très laxiste sur la question de la politique de gestion des threads et des priorités. Une JVM dans laquelle tous les threads seraient de même priorité est parfaitement conforme, mais il va de soi que notre système, et beaucoup d'autres, ne fonctionneraient pas correctement.

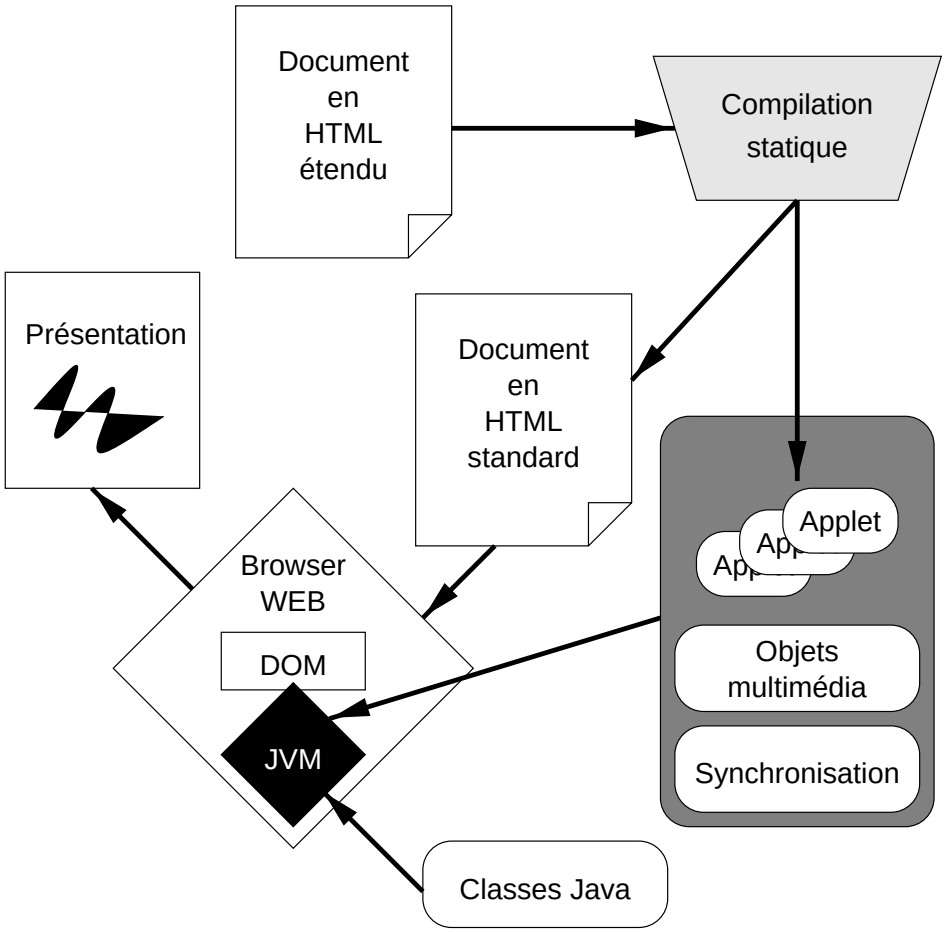


Figure 5.7: Conception du second prototype

Une classe abstraite `SyncEvent` est définie. Elle est sous-classée en `IntraEvent` et en `InterEvent`. Les événements de type intra sont générés par les objets pour les besoins de leur synchronisation interne. Les événements de type inter servent à l'ordonnancement. Parmi ceux-ci deux sous-classes particulières sont définies, `StartEvent` et `StopEvent` qui servent respectivement à démarrer et arrêter l'objet destinataire d'événements de ce type.

Les objets multimédia sont divisés en deux parties distinctes :

- la partie commune à tous les objets, celle qui est indépendante du type de média manipulé : celle-ci est spécifique à l'architecture de synchronisation et regroupe les variables et méthodes nécessaires au traitement général de tous les objets dans la classe `SyncObject` ;

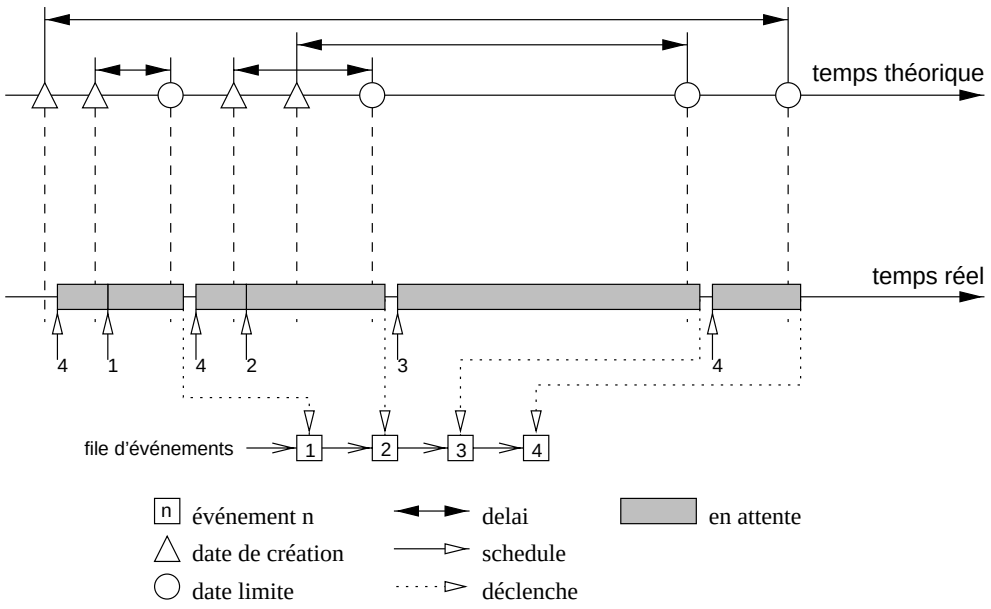


Figure 5.8: Fonctionnement du scheduler

- la partie spécifique à chaque objet, son type, le décodage, etc. : celle-ci ne fait pas partie de l'architecture mais en utilise le support temporel et y est intégrée en utilisant l'interface `Synchronizable`.

L'interface `Synchronizable` définit les méthodes devant être implémentées par un objet quelconque pour le rendre synchronisable par cette architecture. L'objet abandonne en quelque sorte la gestion du temps à un tiers. Celui-ci est alors enveloppé dans un `SyncObject` et ainsi le support de synchronisation peut manipuler des objets divers de façon transparente.

Comme prévu, le tableau de la figure 5.10 montre que l'implémentation de l'architecture de synchronisation est très légère et efficace. Le temps nécessaire à la gestion des événements est totalement négligeable par rapport à la lourdeur des traitements réalisés sur les données multimédia que nous allons aborder dans la section suivante. Les données correspondent à une présentation composée de texte, de cinq pistes audio, deux vidéos et deux séquences de sous-titres associées à une vidéo.

La durée totale de présentation est égale à la durée de vie du scheduler. La durée d'activité du scheduler représente le temps passé par celui-ci à gérer et déclencher les événements, et on peut constater qu'il passe une grande partie de son temps à attendre qu'un événement arrive à sa date limite. Pour

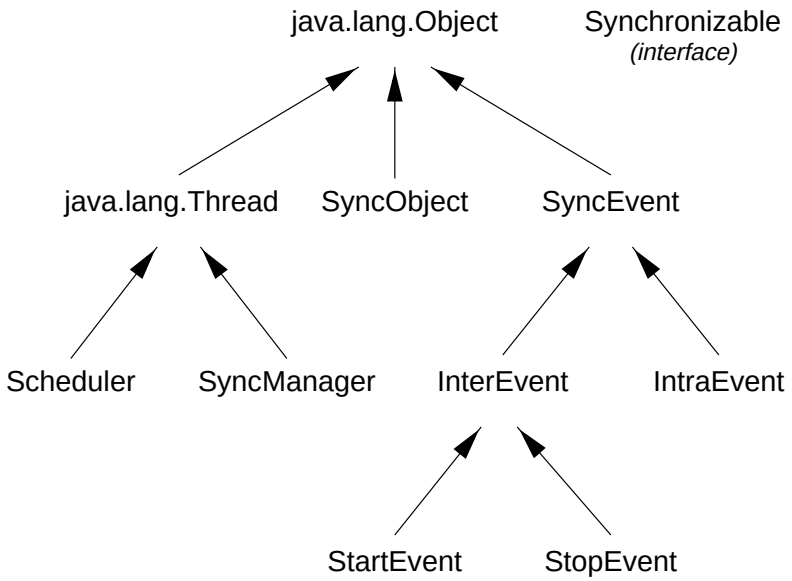


Figure 5.9: Hiérarchie de classes Java implémentant l'architecture de synchronisation, package `osf.sync`

	Durée d'activité	Durée d'attente	Durée de vie	Pourcentage de l'activité totale
Présentation			370 725 ms	
Synchronisation				2,598 %
scheduler	56 ms	370 669 ms	370 725 ms	0,015 %
managers	9 577 ms			2,583 %
Objets	361 092 ms			97,402 %

Figure 5.10: Performances de la synchronisation

simplifier les calculs nous avons imputé aux gestionnaires de synchronisation (managers) tout le temps qui n'est pas occupé par le scheduler et les objets. Malgré cette approximation, manifestement au désavantage de la synchronisation, le temps passé dans celle-ci est environ 2,6 % du temps de traitement total.

5.2.4 Objets multimédia

Afin de produire réellement des présentations multimédia, des objets plus évolués que les images et les graphiques sont nécessaires. Il y a la possibilité

dans les applets d'utiliser de l'audio, mais cela est limité à un seul format et codage et aucun contrôle n'est disponible afin de réaliser la synchronisation. Nous avons donc développé des objets multimédia permettant de présenter de la vidéo MPEG-1 et 2, des sous-titres associés à une vidéo et de l'audio, regroupés dans l'arborescence de packages `osf.media.*`.

Un décodeur MPEG-2 a été implémenté en Java à partir du code source en C de `mpeg2decode` version 1.2 du 19 juillet 1996 délivré par le *MPEG Software Simulation Group*[1]. La version initiale a été encapsulée et légèrement modifiée pour permettre le contrôle de synchronisation. Le principal problème est la vitesse de décodage et d'affichage, plus faible que pour la version originale, cependant l'amélioration des compilateurs JIT n'a cessé d'améliorer les performances pour arriver aujourd'hui presque à 25 images par secondes sur des vidéos de petit format. L'utilisation de compilateurs statiques comme *TurboJ* développé à l'*Open Group Research Institute* permet d'accroître encore les performances mais au détriment de la portabilité. Un type d'objet particulier permettant d'afficher conjointement des sous-titres avec une vidéo a également été développé.

De nombreuses classes ont aussi été développées pour permettre l'utilisation d'audio dans des formats et des codages variés sur station *Sun Sparc Solaris*. Il est également possible de les mixer entre eux afin de jouer plusieurs sources audio simultanément. Cependant, la restitution sonore demande un accès direct au périphérique ce qui impose le recours à des portions de code natif et pose des problèmes de portabilité et de sécurité dans le cas des applets.

Il est possible aussi d'utiliser des objets représentant des fichiers HTML au moyen des classes fournies par *Sun* dans le nouvel environnement graphique *Swing*. Nous venons également de terminer l'interfaçage du décodeur et du codeur H.261 de *vic*[74] avec Java que nous comptons utiliser pour des expérimentations sur le transport des données, dont nous parlerons dans la section 5.3. Le décodeur permettra d'inclure des flots vidéo en direct dans les documents multimédia.

[1] `http://www.mpeg.org/MSSG/`

5.2.5 Analyse des extensions à HTML

L'analyse d'un document composé d'extensions multimédia à HTML est réalisée à l'aide d'un analyseur lexical et syntaxique généré à partir d'une grammaire de ces extensions et du générateur automatique SableCC[33] développé à l'université de McGill de Montréal. SableCC est un générateur de compilateurs orienté objet implémenté en Java et reposant sur deux concepts fondamentaux. Premièrement, il utilise des techniques orientées objet pour construire un arbre syntaxique abstrait, AST, *abstract syntax tree*, strictement typé. Deuxièmement, il génère des classes de parcours de l'arbre en utilisant une version étendue du principe des visiteurs. Cet outil permet la réalisation aisée de compilateurs en se reposant sur l'héritage : seule les productions spécifiques sont à implémenter, le reste est présent par défaut et peut être modifié en sous-classant si nécessaire.

Ainsi, l'analyse d'un document multimédia produit un AST fortement typé représentant sa structure. Deux visiteurs ont été écrits, un pour vérifier la cohérence des identificateurs utilisés dans le document et le second pour générer la présentation. Ce dernier collecte les informations nécessaires dans l'AST et appelle une classe tierce pour effectuer la génération proprement dite de la présentation multimédia. Cette conception modulaire a permis de réutiliser cet analyseur tel quel dans les différentes variantes du prototype. La figure 5.11 illustre ces principes.

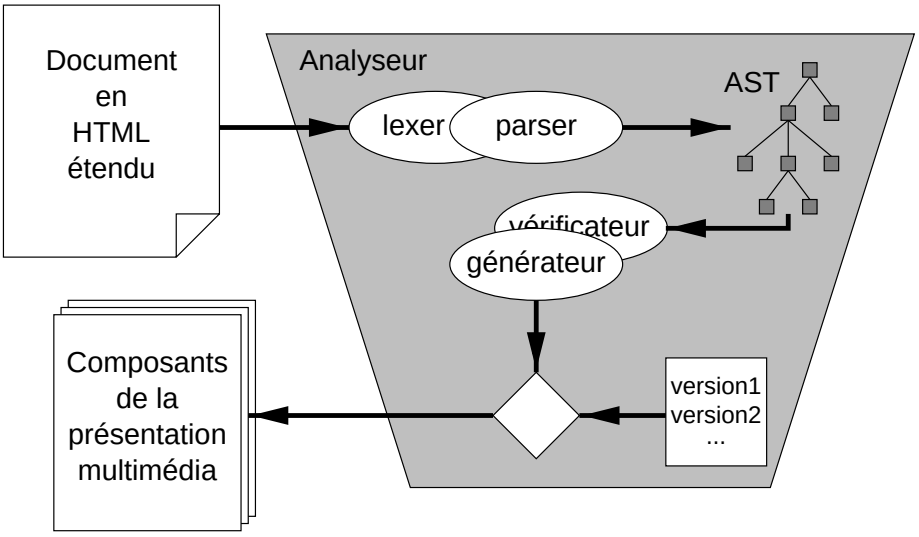


Figure 5.11: Structure du compilateur de documents HTML étendu

Les classes développées pour réaliser ce compilateur de présentations ainsi que les classes utilitaires nécessaires ont été regroupés dans la hiérarchie de packages `osf.madcow.*`.

5.2.6 Génération de la présentation

Nous allons maintenant décrire les deux versions du prototype, qui constituent dans sa réalisation deux étapes pour s'orienter vers une intégration plus grande avec le monde du Web. Nous allons décrire dans cette partie uniquement les différents modules de génération utilisés puisque la partie d'analyse est à chaque fois la même.

Première version

La première version constitue l'approche la plus simple pour jouer un document multimédia sur le Web. Il s'agit de générer une applet Java qui exécute cette présentation et de créer une page HTML très simple contenant cette applet. Alors, la visualisation de la présentation se fait en chargeant cette page dans un browser capable d'exécuter des applets Java.

Cette technique est efficace mais fait perdre sa nature HTML au document. De ce point de vue, on ne peut pas réellement parler d'intégration au Web car le document est uniquement constitué de bytecode Java et ne comporte plus de composante HTML, si ce n'est la référence à l'applet.

Deux solutions sont possibles pour la génération de l'applet. Soit elle est effectuée en mode statique, c'est-à-dire que le document initial est compilé en code source Java qui est à son tour compilé en bytecode. Toute référence au document initial est alors perdue. Soit l'applet est générée à la volée. Dans ce cas, une applet générique est utilisée. Une fois sur le client Web, elle charge le document multimédia qui est compilé à la volée. Lors de ce processus les objets sont générés dynamiquement et finalement la présentation démarrée.

DOM

La technologie DOM, évoquée au chapitre précédent en 4.1.2, permet de contrôler les éléments inclus dans une page HTML de façon dynamique

lorsque celle-ci est présentée. Les browsers n'implémentent pas aujourd'hui la recommandation DOM issue du W3C, cependant les concepts principaux sont présents et offrent un contrôle limité mais suffisant pour la réalisation d'un prototype. Il n'est donc pas possible de se conformer exactement à la figure 5.7. Plutôt que de générer un document en HTML standard, nous allons générer un document en HTML dynamique, *Dynamic HTML*, qui est une version spécifique à *Netscape* proposant des capacités de contrôle dynamique. *Microsoft* propose *DHTML* qui est semblable dans le principe, mais dont les interfaces de programmation diffèrent.

La principale extension de *Dynamic HTML* que nous allons utiliser est le tag `layer`. Celui-ci définit un objet conteneur d'objets graphiques dont certains des attributs sont manipulables dynamiquement depuis JavaScript et Java. Parmi ces attributs exposés, on retrouve les attributs graphiques visibilité, taille et position. Ils sont suffisants pour réaliser les opérations que nous avons définies dans le langage.

Bien sûr, le travail réalisé spécifiquement pour l'adaptation à cette version particulière de HTML pourra être récupéré et modifié assez facilement lorsque DOM aura été intégré dans l'architecture des browsers. Cela rendra alors le prototype portable.

Seconde version

L'idée de base dans la réalisation de cette seconde version est la suivante. Prenons un document en HTML temporel. Pour qu'il puisse être présenté dans un browser, il faut que celui-ci comprenne les éléments inclus dans la page. La solution envisagée consiste à analyser le document initial, à le transformer en un document HTML compréhensible par le browser et à remplacer les éléments qui lui sont inconnus, c'est-à-dire les extensions temporelles, par des applets Java.

Tout objet multimédia inconnu de HTML et du browser est remplacé par une applet. Tous les objets existants dans le HTML standard sont transformés ou laissés en HTML. Une applet supplémentaire en charge de l'initialisation des objets de la page Web est générée. Une fois les objets initialisés, elle lance la présentation qui est prise en charge par le support de synchronisation. Cette applet gère également, par l'intermédiaire des *layers*, l'interfaçage entre Java et les objets HTML affichés par le browser. Elle peut ainsi, par exemple, montrer ou cacher une image en fonction de la composition temporelle.

Un document doit donc être compilé statiquement pour produire une nouvelle version HTML et un ensemble d'applets comme cela est illustré sur la figure 5.7. La structure complète de l'architecture est illustrée sur la figure 5.12. Le browser affiche le document *Dynamic HTML*. Le style de celui-ci peut être défini avec CSS, Cascading Style Sheets. L'environnement Java contient tout le traitement relatif à la synchronisation, au traitement et au transport des médias. Les modules de décodage vidéo H.261 et de transport RTP/RTCP ne sont pas encore intégrés au prototype actuel. Nous allons aborder la problématique liée au transport dans la section 5.3 qui suit celle-ci.

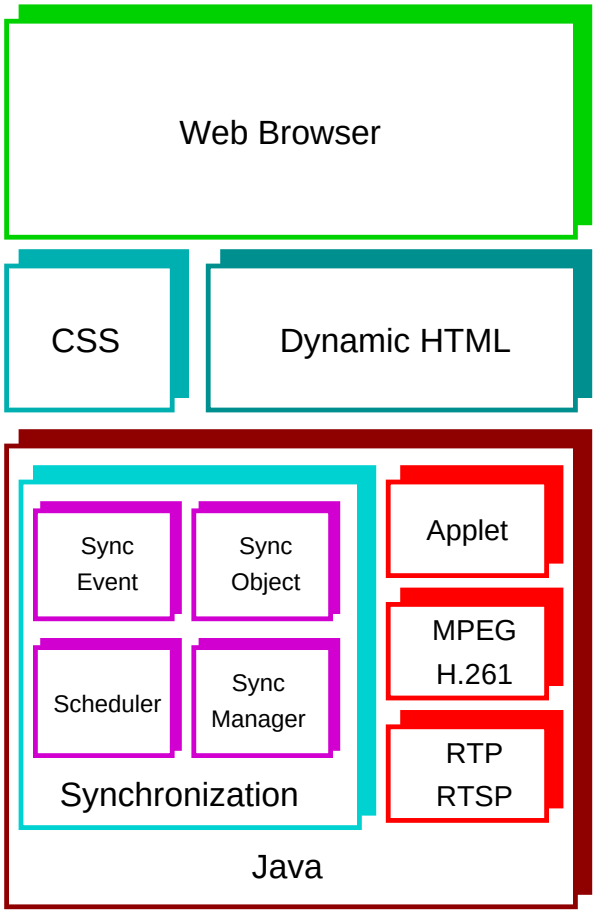


Figure 5.12: Architecture du prototype final

5.2.7 Conclusion

Nous avons réalisé un prototype mettant en œuvre les concepts définis aussi bien au niveau spécification par les extensions au langage HTML qu'au niveau présentation par la définition de l'architecture de synchronisation. Le caractère original de cette implémentation est son intégration presque parfaite dans l'environnement Web déjà existant. Par l'utilisation des technologies les plus récentes, il a en effet été possible d'étendre de façon assez transparente les capacités du browser *Netscape*. Toutefois, ce travail est dépendant d'une technologie propriétaire particulière, même si elle est largement déployée, et il deviendra vraiment intéressant lorsque des solutions standardisées seront implémentées dans tous les clients Web.

Toutes les fonctionnalités définies dans le langage ne sont pas encore implémentées dans le prototype. La principale activité lors de sa réalisation a été l'étude des technologies disponibles pour sa mise en œuvre et le test de celles-ci pour évaluer leur adéquation réelle aux problèmes à traiter. Une partie non négligeable du développement à également consisté à développer le support pour plusieurs types de médias en Java parce que celui-ci était pratiquement inexistant quand nous avons commencé ce travail. Les quelques implémentations disponibles n'offraient pas le contrôle nécessaire à la réalisation de la synchronisation. Les objets multimédia implémentés constituent également un frein à l'ajout de certaines des fonctionnalités manquantes parce qu'ils n'offrent pas les moyens de les réaliser et nécessitent des modifications assez profondes. Leur ajout au prototype est donc inutile si finalement ces fonctionnalités étaient ignorés lors de l'exécution.

Malgré ces limitations, les principaux concepts développés dans ce travail de thèse ont été concrétisés dans ce prototype : les liens hypertemps, les bases temporelles, les objets synchronisables, la gestion dynamique des événements. Bien qu'elles limitent les possibilités d'édition, les fonctionnalités manquantes ne remettent jamais en cause la validation des concepts. Il n'est pas possible, par exemple, de définir des points temporels autres que le début et la fin sur certains objets, à cause de limitations propres à ceux-ci. Le fonctionnement général n'est pas remis en cause, puisque du point de vue de la gestion des événements, c'est le même en ces points internes qu'au début et à la fin.

La seule limitation importante vient du fait que nous n'avons encore pas défini comment spécifier le mouvement des layouts et des frames dans les extensions à HTML. Il ne s'agit là que d'un problème de langage parce que cette fonctionnalité est disponible au niveau implémentation. Il est par exemple

possible en éditant le code source Java généré par le compilateur MADCOW de programmer, de façon *ad hoc*, le mouvement synchronisé d'un objet à l'écran en modifiant dynamiquement sa position à travers *Dynamic HTML* (voir en 5.2.6).

Les performances de ce prototype sont plutôt satisfaisantes lorsqu'il est testé sur un réseau local avec des présentations faisant appel uniquement à des vidéos de petite taille et ne les présentant pas simultanément. Elles dépendent toutefois beaucoup du type de machine virtuelle Java utilisée : l'absence de compilateur JIT limite grandement les capacités de décodage vidéo et dégrade donc beaucoup la qualité obtenue. Il est assez difficile de quantifier exactement les performances d'une présentation multimédia puisque la perception par l'utilisateur est un facteur hautement subjectif. De plus, notre but premier n'était pas de travailler sur la qualité des présentations mais de montrer que l'on pouvait définir un modèle de spécification de présentations multimédia adapté au Web, ainsi qu'une architecture de synchronisation. La qualité de restitution des médias n'est pas de notre ressort et de nombreux travaux ont cours sur ce sujet. On peut donc supposer raisonnablement que dans peu de temps l'amélioration des techniques logicielles et l'augmentation de la puissance des machines permettra d'obtenir une qualité de média quasiment parfaite.

Le facteur de performance qui nous intéresse est en fait paradoxalement le comportement de la présentation lorsque la synchronisation ne peut se faire normalement. La cohérence avec la spécification doit être conservée en toute circonstance. Un test simple consiste à présenter une vidéo définie comme maître, synchronisée avec d'autres médias, et de perturber sa restitution en ralentissant le décodage d'une façon quelconque, en chargeant la machine ou le réseau par exemple. L'expérience la plus simple consiste à désactiver le compilateur JIT de la machine virtuelle Java, faisant alors tomber le décodage de la vidéo à quelques images par seconde seulement. On peut alors constater que le comportement correspond bien à celui décrit plus haut. La vidéo étant ralentie, elle ralentit également tous les objets synchronisés avec elle, conservant ainsi la cohérence avec la spécification. Lorsque le décodage reprend son rythme normal, les autres objets font de même, sans qu'aucun décalage n'ait été introduit. L'architecture de synchronisation garantit donc qu'en cas de perturbation intempestive la cohérence avec la spécification temporelle est conservée.

5.3 Transport et qualité de présentation

Afin de pouvoir garantir qu'une présentation multimédia sur le Web sera restituée avec une qualité optimale, le contrôle de la synchronisation sur le client ne suffit pas. Dans un réseau à grande échelle comme l'Internet, qui est le support du Web, le transport des données multimédia constitue un obstacle de taille. La bande passante en général assez faible et variable, ainsi que les délais variables observés dans ce type de réseau, ne permettent pas le transfert de médias continus dans de bonnes conditions (voir en 2.3).

Nous avons vu plus haut que les tests effectués sur réseau local se passent correctement parce que dans ce cas le transport des données ne constitue généralement pas un problème. Pour conserver ce niveau de qualité de présentation il faut assurer un transport de données aussi transparent que possible.

5.3.1 Flots de données multimédia

Le domaine des communications multimédia est aujourd'hui en plein essor et de nouveaux protocoles ont été spécifiés ou sont en cours de spécification pour traiter les différents aspects liés au transport de flots de données temps réel.

Protocoles

- RSVP, Reservation Protocol[8], permet au destinataire d'un flot de données de spécifier et réserver les ressources qu'il désire pour satisfaire une certaine qualité de service, QoS.

- RTP, Real-time Transfert Protocol[101], est un protocole de transport léger adapté à différents types de médias[99]. RTCP, le protocole de contrôle associé fournit des statistiques sur le transport des données permettant de mettre en œuvre des mécanismes de contrôle de flux.

- RTSP, Real-Time Streaming Protocol[102], offre des fonctionnalités de signalisation et de contrôle de flots média : récupérer des informations sur les médias disponibles sur le serveur, initier, démarrer, arrêter et terminer une session.

- D'autres protocoles comme SIP, Session Invitation Protocol, et SDP, Session Description Protocol[100, 92, 39], sont plus spécifiquement destinés à la vidéoconférence sur Internet ou le MBone.

L'architecture générale du contrôle d'un flot est illustrée sur la figure 5.13. Tous ces protocoles ont été définis dans le cadre du développement de la vidéoconférence sur Internet. Ils ont cependant été conçus de manière à ne pas être spécifiques à ce domaine particulier du multimédia et peuvent être utilisés dans le cas qui nous intéresse, la présentation de documents multimédia.

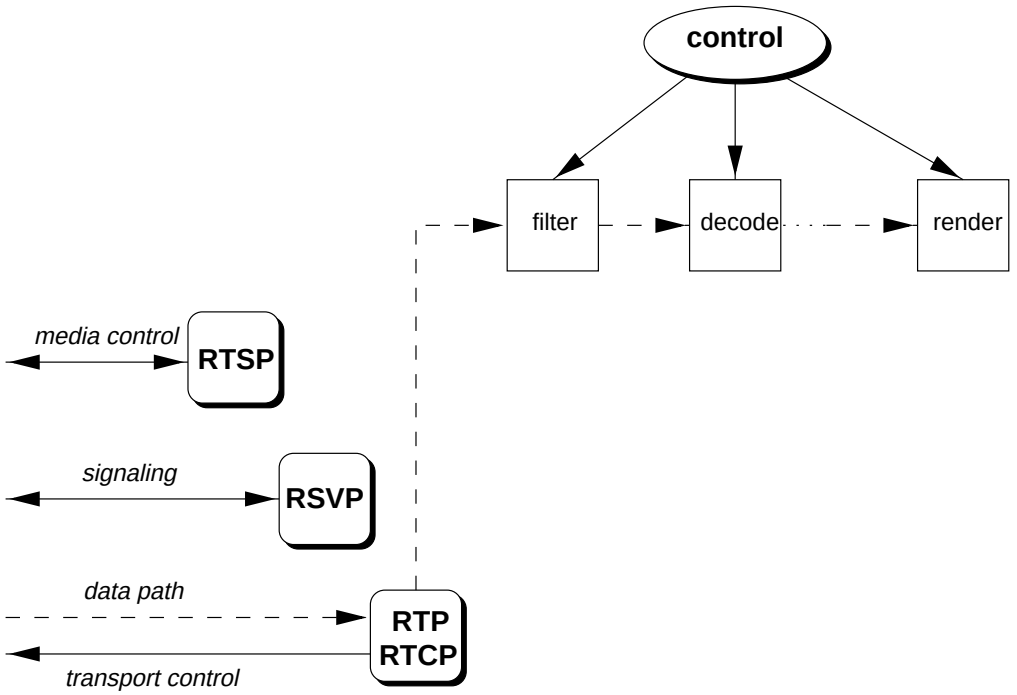


Figure 5.13: Architecture de contrôle d'un flot temps-réel

Gestion de flots

La vidéoconférence multipartie met en jeu un grand nombre de flots. Chaque correspondant envoie ses données à tous les autres et reçoit celles de tous les autres. Le transfert de données se fait en utilisant la multidiffusion de paquets, *multicast*, que nous avons brièvement décrit en 2.3.4. Nous considérons la vidéoconférence point-à-point comme un cas particulier.

Tous les participants reçoivent en général les mêmes données. La source émettrice transmet régulièrement des paquets de contrôle contenant des informations sur son flot. Chaque participant retourne alors des paquets de contrôle à l'émetteur contenant des statistiques sur les conditions de réception. La source émettrice peut alors adapter son débit en fonction du retour fourni par les récepteurs, voir la figure 5.14. La grosse difficulté réside dans le fait que plusieurs destinataires sont concernés, les décisions doivent donc être basées sur un compromis satisfaisant le maximum de participants.

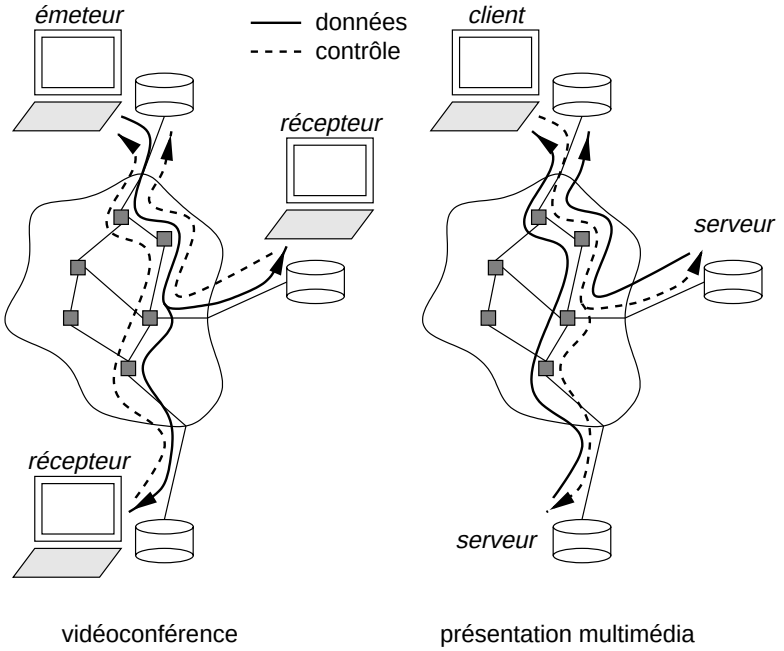


Figure 5.14: Contrôle des communications multimédia

Dans le cas de documents multimédia la problématique n'est pas la même puisque le contenu n'est pas "multicasté", la seule exception étant éventuellement les flots en direct référencés dans le document, auquel cas ce que nous avons dit ci-dessus s'applique à ce flot en particulier. Pour les autres médias constituant le document, l'environnement de présentation va initialiser des flots de données multimédia en point-à-point, *unicast*. Pour les médias continus, ces flots sont de nature similaire à des flots de vidéoconférence et utiliseront les mêmes techniques de transport. Toutefois, la contrainte temporelle est moins forte et autorise une plus grande marge de manœuvre dans sa gestion : il est possible de jouer sur les paramètres de stockage temporaire, *buffering*, d'anticipation, *pre-fetching*, etc., afin d'améliorer la qualité des données récupérées à qualité de transport égale.

Cela autorise une plus grande souplesse dans la gestion des communications puisque la topologie du transport est inverse : un seul client reçoit des données de plusieurs serveurs. Le contrôle est donc également inversé, c'est le client qui a la possibilité de gérer en partie le transport, voir la figure 5.14.

5.3.2 Qualité de présentation

L'expression Quality of Presentations, QoPs, a déjà été utilisée dans d'autres travaux[45, 46], cependant les auteurs basent la qualité uniquement sur un critère de synchronisation. La QoPs est définie par rapport aux anomalies de synchronisation intra et inter-média. Nous entendons par qualité de présentation, QoP, la qualité du résultat perçu par l'utilisateur. Celle-ci est évidemment subjective, mais elle dépend principalement de trois facteurs quantifiables :

- la *qualité de synchronisation*, QoSync ;
- la *qualité des médias*, QoM ;
- la *qualité de service*, QoS.

Contrôle global

Dans le cas qui nous concerne, nous souhaitons offrir un contrôle de qualité de présentation qui soit global au document multimédia et qui prenne les problèmes au plus tôt afin de les traiter au mieux et le plus efficacement possible. L'asynchronisme dans une présentation est en fait le résultat visible de problèmes plus profonds dont l'origine peut être localisée en trois endroits :

- sur le *client* : la charge du client ou sa puissance trop faible pour réaliser une présentation peut provoquer des chutes de performances locales venant du manque de ressources matérielles ;
- sur le *serveur* : le même problème peut se produire sur le serveur, à la différence près que l'utilisateur n'a aucun recours pour le résoudre, sauf bien sûr s'il en est l'administrateur ;

- dans le *réseau* : un trafic important sur le réseau utilisé pour transporter les données multimédia peut créer de fortes baisses de qualité, soit par la perte pure et simple des données, soit par leur retard qui occasionne des discontinuités dans les flots temps-réel et équivaut à la perte, si l'on souhaite continuer la présentation, ou à un arrêt temporaire, si l'on souhaite attendre leur arrivée.

Pour obtenir une qualité de présentation optimale, il faut donc contrôler les trois facteurs cités plus haut en commençant par le plus bas niveau pour remonter au plus haut niveau. Il faut premièrement s'assurer de l'état de la liaison et éventuellement réserver la QoS nécessaire si le réseau le permet. Ensuite, il peut être nécessaire de dégrader la qualité du média transporté. Enfin, en dernier recours, le client devra adapter la synchronisation à la qualité qu'il arrive à obtenir.

Cela nécessite des capacités de négociation entre le client et le serveur. Le client ne peut pas contrôler directement le serveur et doit donc procéder indirectement, en lui fournissant certains paramètres ainsi que des requêtes. Un protocole comme RTSP peut être utilisé pour cette négociation. La procédure se présente ainsi :

- le client demande une certaine qualité de service en fonction du débit dont il dispose, celle-ci peut être maintenue par un protocole comme RSVP s'il est disponible ou il peut s'agir seulement d'une indication de débit maximum au serveur ;

- le client demande également une certaine qualité de média correspondant à ses capacités, pas besoin d'images couleur de grande taille sur un petit écran en niveau de gris par exemple ;

- le serveur réduit la qualité du média envoyé pour satisfaire au mieux ces contraintes ;

- lors de la présentation, une renégociation peut être effectuée en utilisant le même processus pour s'adapter aux conditions changeantes, soit sur le réseau, soit sur le client.

Gestion de flots multiples

Le principe de contrôle décrit ci-dessus correspond bien à la gestion d'un flot par le client. Cependant, il est insuffisant dans le cadre de la présentation de documents multimédia complexes.

Du point de vue du transport, un document multimédia est perçu comme un ensemble de flots de données provenant de serveurs distants répartis dans le réseau (figure 5.14). La spécification du document définit une certaine sémantique sur ces flots, dont certains sont prépondérants par rapport aux autres. La définition d'objets en tant que maîtres leur donne une importance supérieure de fait sur les esclaves. De plus, la nature du média transporté influe également beaucoup sur l'importance du flot : l'audio étant très sensible aux pertes, il faudra s'assurer de son transport dans de bonnes conditions au détriment des médias moins sensibles comme la vidéo. Finalement, on peut imaginer un mécanisme de désignation explicite de l'importance dans le langage de description de documents.

Afin de répartir correctement les ressources entre les différents composants d'un document, en respectant cette notion d'importance entre les objets, il est nécessaire de contrôler la qualité au niveau document. À cette fin, nous proposons d'étendre l'architecture de synchronisation avec de nouveaux modules.

5.3.3 Architecture de contrôle de qualité de présentation

La figure 5.15 illustre la structure de l'architecture. On identifie quatre composants principaux : les gestionnaires de synchronisation, de média, de service et de qualité de présentation.

Le gestionnaire de QoP est le point central du dispositif. Il est notifié par les objets et le gestionnaire de synchronisation des problèmes rencontrés. Sur la base de ces informations, il décide des mesures devant être prises.

- Il est informé de l'objet en difficulté et du type de problème : transport ou décodage, c'est-à-dire réseau ou client.
- Il doit choisir le ou les objets sur lesquels agir, cela en fonction de leur niveau d'importance. L'objet rencontrant des problèmes n'est pas

nécessairement celui sur lequel une action sera entreprise. Si un flot audio provenant d'un serveur subit des pertes trop importantes alors qu'un flot vidéo provient également de ce même serveur, il est par exemple souhaitable de diminuer la cadence d'affichage de la vidéo pour libérer les ressources nécessaires au flot audio.

- Il doit choisir le niveau d'action en fonction des possibilités. Une action au niveau service est d'abord tentée en essayant de réserver une QoS supérieure, par exemple. Si cela n'est pas possible, une action au niveau média peut être prise, soit en réduisant la qualité d'un média de moindre importance pour libérer de la bande passante comme dans l'exemple cité ci-dessus, soit en réduisant la qualité du média en difficulté pour diminuer le débit nécessaire à son transport.

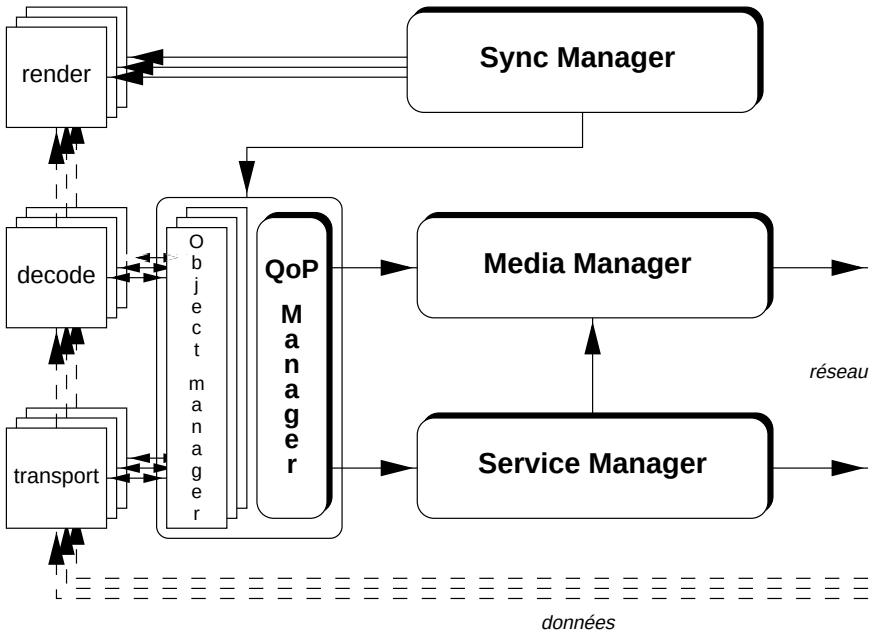


Figure 5.15: Architecture de présentation de documents multimédia intégrant synchronisation et communications

Nous avons présenté la conception de l'architecture de contrôle ainsi que son principe de fonctionnement. Pour le moment, nous n'avons encore étudié les modalités de mise en œuvre. Il faut concevoir le support de négociation entre client et serveur, nous comptons pour cela utiliser le protocole RTSP existant, ainsi que les algorithmes permettant d'effectuer les décisions en fonction de l'analyse des difficultés rencontrées.

Cette partie peut bénéficier de travaux déjà réalisés dans ce domaine. Campbell et Coulson ont défini une architecture de qualité de service, QoS-A, dans un environnement ATM fournissant des garanties de service strictes ainsi qu'un système temps-réel[16, 17], caractéristiques que l'on ne trouve généralement pas sur le Web. D'autres travaux concernent la négociation et le contrôle de qualité de service[3], appliqués aux communications standard[29] ou au cas du multicast[72].

Une architecture de contrôle pour le multimédia sur Internet est présentée par Schulzrinne[100] mais celle-ci est très orientée vers la vidéoconférence multipartie. Toujours dans le domaine de l'Internet, un effort commun est entrepris[73] afin de fournir une architecture commune de *multimedia middleware* basée sur l'approche commune utilisée par *CMT*[96], *vic*[74] et *vat* ainsi que *VuSystem*[64]. D'autre part, de nombreuses techniques de contrôle logiciel au niveau flot ont été proposées[18, 19, 20, 88] et peuvent constituer une base au dessus de laquelle une architecture telle que celle que nous présentons peut être développée.

Chapitre 6

Conclusion

6.1 Bilan du travail réalisé

Le travail réalisé au cours de cette thèse s'inscrit dans le vaste domaine du multimédia. Nous avons essayé de présenter un panorama des systèmes et applications multimédia du domaine informatique en ouverture de ce document pour situer notre activité, les présentations multimédia. La singularité du multimédia vient du fait qu'il conduit aujourd'hui au regroupement d'activités qui n'étaient traditionnellement pas liées. Dans l'informatique, où les différentes communautés des langages, des communications, des bases de données, des systèmes se mettent à collaborer sur le thème commun du multimédia pour essayer d'apporter des solutions globales aux problèmes soulevés. Mais aussi en dehors. Depuis l'avènement du numérique, de nouvelles applications ont vu le jour dans les domaines des télécommunications et de la télévision, qui se rapprochent ainsi un peu plus encore de l'informatique.

Cette richesse du domaine multimédia nous amène à faire des choix très précis sur notre domaine d'activité. Bien que de nombreuses passerelles existent entre les activités citées plus haut, il n'est pas possible d'attaquer le problème de front dans son ensemble. Nous avons choisi de nous focaliser sur les *présentations multimédia synchronisées pour le World Wide Web*. Le Web est aujourd'hui un support fédérateur puisqu'il est né de l'informatique et des réseaux d'ordinateurs, mais il est également considéré comme support d'information pour la télévision interactive ou la téléphonie mobile[115].

L'étude des travaux réalisés sur les présentations multimédia, exposée

dans le troisième chapitre, a permis de mettre en lumière les avantages et les inconvénients des différents modèles et d'éviter ainsi les lacunes de certaines propositions. De plus, notre démarche a volontairement été appliquée dès le départ. Ayant choisi de travailler dans l'environnement du Web, nous avons inscrit notre proposition dans la continuité des travaux réalisés actuellement. Cela a permis de concrétiser le travail théorique sous la forme d'un prototype tirant parti des technologies récentes toujours en développement.

Lorsque ce travail a débuté, seul le mécanisme des plug-ins permettait d'ajouter quelques capacités multimédia aux pages HTML, sans fournir de possibilité d'intégration ni de contrôle. Aujourd'hui, il existe un certain engouement dû à l'apparition du langage SMIL, qui provoque l'intérêt de certains acteurs majeurs du multimédia et du Web comme *Macromedia* et *RealNetworks*. Nous avons exposé dans ce document notre travail sur la description de présentations multimédia et sur leur exécution dans l'environnement du Web. Nous pensons avoir apporté des réponses originales aux problèmes suivants dans le cadre de la spécification de présentations.

- *Modélisation et représentation de la synchronisation*: la double problématique de la synchronisation inter-média, d'une part, l'*ordonnancement* des objets, d'autre part, le *couplage inter-média*, a été identifiée dès le début de ce travail.

- *Composition temporelle*: celle-ci exprime la façon dont les objets devront être ordonnancés lors de la présentation. Un modèle de description de la composition temporelle basé sur une représentation du temps par instants est défini. Il se concrétise à travers des extensions au langage HTML, en étendant le concept de lien hypertexte au domaine temporel sous forme de *lien hypertemps* ou lien temporel.

- *Couplage inter-média*: ce problème n'est en général pas abordé dans les modèles de synchronisation, ou alors très superficiellement, et laissé au support d'exécution. Toutefois, le rôle de ce dernier est de réaliser le couplage et non de le définir. L'introduction des *bases temporelles* et des *rôles* des objets fournit une méthode de spécification du couplage qui est directement applicable au niveau présentation par l'intermédiaire de l'architecture de synchronisation proposée conjointement.

- *Intégration spatial-temporel*: l'ajout de la dimension temporelle aux documents Web traditionnels a dépassé la simple juxtaposition à la description spatiale initiale. La distinction entre le comportement d'un

objet et le comportement de sa représentation, faite par l'introduction de la *disposition dynamique* ou *dynamic layout*, permet une véritable intégration. La représentation spatiale est désormais dynamique et participe au comportement temporel du document au même titre que les objets multimédia.

Conjointement aux solutions apportées dans le cadre de la spécification, nous proposons également des solutions permettant de les exploiter au mieux lors de l'exécution des présentations. Ces travaux ont été menés parallèlement, ce qui a présenté plusieurs avantages majeurs. D'abord, nous avons montré qu'il était possible de concrétiser simplement et efficacement les concepts que nous avons définis, et ainsi évité l'écueil des propositions restant à l'état théorique. Ensuite, cela a permis d'adapter au plus près la synchronisation et l'exécution en décelant les éléments manquants ou superflus d'un côté comme de l'autre. Enfin, la réalisation d'un prototype a nécessité un suivi continuel des nombreuses propositions et recommandations régissant l'évolution du Web. Cela nous a permis de rester en phase avec les avancées les plus récentes, malgré la progression extrêmement rapide de la recherche en informatique dans les domaines des communications, d'Internet et du Web. Nous avons pu ainsi adapter au mieux nos propres propositions aux technologies les plus récentes et réaliser un prototype en tirant parti.

- *Architecture de synchronisation* : elle utilise une approche inspirée des objets actifs qui a été spécialisée pour la gestion de la synchronisation, donnant ainsi un modèle plus léger et mieux adapté. Une technique de gestion dynamique des événements permet de garantir à moindre coût la cohérence de la présentation avec la spécification qui en a été faite par l'auteur. Une bibliothèque de classes Java implémente ces fonctionnalités.

- *Prototype* : il a été développé au-dessus de l'architecture de synchronisation, et un effort particulier a été fait pour tirer parti au maximum des technologies actuellement en développement dans l'environnement du Web.

- *Contrôle de qualité de présentation* : finalement, à partir de l'analyse de la problématique du transport des données multimédia temporelles dans un environnement *best effort* distribué comme l'Internet, une architecture de contrôle global des présentations a été définie. À terme, l'objectif est de fournir une qualité de présentation optimale en traitant

les problèmes au niveau le plus adapté dans la chaîne serveur-transport-client.

Le travail exposé dans cette thèse a donné lieu à deux publications[95, 93] et à la présentation d'un poster[94] dans des conférences internationales. Beaucoup de points restent à préciser et donneront sûrement lieu à des travaux supplémentaires sur la base de ceux que nous venons de présenter.

6.2 Perspectives

L'activité multimédia est aujourd'hui importante dans la recherche et les perspectives d'avenir sont évidemment nombreuses. Nous allons les aborder maintenant en considérant deux points de vue, d'abord celui de notre travail en présentant les évolutions possibles et la direction que nous souhaitons prendre, enfin celui plus général du multimédia dans son ensemble en exposant quelles sont les tendances de son évolution et comment nous pouvons nous inscrire par rapport à celles-ci.

Tout au long de ce document, nous avons signalé les limites de nos propositions et les points nécessitant des précisions. Nous allons y revenir en proposant des directions d'étude possibles.

- L'utilisation de HTML pose des problèmes dans le cas particulier de la spécification des objets composés. Il est en effet théoriquement possible de créer des objets composés dont les composants appartiennent à des bases temporelles différentes. Toutefois, la méthode déclarative des extensions, qui a été calquée sur celle de HTML pour des raisons évidentes d'homogénéité, ne permet pas cela. Il faudrait disposer d'une distinction entre déclaration d'un objet et référence à celui-ci, qui permettrait d'inclure un même objet à l'intérieur de différents éléments. Des solutions standard seront probablement apportées dans le cadre de XML.
- Une méthode de spécification déclarative du mouvement est nécessaire à la description du comportement temporel de la disposition spatiale des objets. Aussi bien le modèle de spécification de présentation que le support logiciel développé permettent l'intégration immédiate de cette amélioration qui a été prise en compte dès le départ lors de leur conception.

- L'extension des propriétés temporelles à tous les éléments de HTML est envisageable afin de créer des documents entièrement dynamiques. Tout élément représentant du contenu, tel un paragraphe `<p>`, peut se voir agrémenté des attributs et éléments internes définis sur les objets. L'apparition du Document Object Model dans les browsers Web permet d'envisager la réalisation de telles fonctionnalités dans un prototype.

- La réalisation de l'architecture de contrôle de la qualité de présentation est un objectif important. Elle constituerait un support d'expérimentation afin de proposer des techniques de négociation entre client et serveurs ainsi que de contrôle de multiples flots de données multimédia.

Pendant le déroulement de ce travail, SMIL, un nouveau langage destiné à la description de documents multimédia pour le Web a été développé à l'initiative du consortium W3, qui gère l'évolution du World Wide Web. Nous avons exposé à plusieurs reprises certaines des imperfections de cette première version du langage. Un prolongement intéressant de ce travail serait de proposer, lorsqu'une évolution de SMIL sera entreprise, quelques unes des solutions que nous avons établies comme alternatives permettant de contourner certaines de ses imperfections.

De façon plus générale, les perspectives liées à notre travail concernent les domaines liés au Web et à l'Internet. Il semble que le Web soit voué à devenir le support de toute une classe de nouvelles applications multimédia. Deux évolutions à venir nous paraissent importantes et sont paradoxalement opposées : d'un côté la télévision digitale, de l'autre la téléphonie mobile et l'informatique nomade.

L'apparition de formats de codage numérique pour les médias continus traditionnels a permis au son et à la vidéo, de faire leur entrée dans le monde de l'informatique et ouvert la voie à la recherche multimédia. Aujourd'hui le phénomène symétrique se produit. Avec l'apparition de la télévision digitale la technologie informatique fait une entrée en force dans le domaine de la diffusion de programmes. Les programmes interactifs, la vidéo à la demande, l'accès au Web par l'intermédiaire de *set-top boxes*, direct ou couplé à des programmes télévisuels, et le commerce électronique sont autant d'applications qui vont bénéficier de la recherche multimédia. Elles pourront également tirer parti de la bande passante volumineuse disponible sur les canaux de communication tels que le câble et le satellite. Simultanément, les communications sans-fil à faible débit prennent de l'importance avec l'envol de la téléphonie mobile et entraînent le développement de l'informatique nomade.

Ces évolutions participent de l'évolution des communications. Bientôt tous les appareils, quels qu'ils soient, seront dotés de capacités à communiquer[78, 23], si ce n'est plus... Cela ira de la montre à la console de jeux, de la télévision au réfrigérateur et du désormais classique ordinateur à l'automobile. Les moyens d'accès à l'information en général et au Web en particulier devront donc évoluer en fonction de ce parc de terminaux et des voies de communication devenant de plus en plus hétérogènes. À nos yeux, l'élaboration de solutions fournissant un accès universel et transparent à l'information distribuée constituera une des principales activités de la recherche à venir et regroupera, entre autres, les activités concernant la description de l'information, l'indexation des données et l'accès par le contenu, le contrôle et la garantie de la qualité de service, l'adaptation contextuelle de l'information et des communications, l'interactivité et la coopération.

Bibliographie

[1] Yahya Y. Al-Salqan and Carl K. Chang. Temporal Relations and Synchronization Agents. *IEEE Multimedia*, 3(2): 30–39, Summer 1996.

[2] J.F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11), November 1983.

[3] Cristina Aurrecoechea, Andrew T. Campbell, and Linda Hauw. A Survey of QoS Architectures. *Multimedia Systems, Special Issue on QoS Architecture*, May 1998.

[4] Brian Bailey and Joseph A. Konstan. Nsync - A Constraint Based Toolkit for Multimedia. In *Proceedings of the Fifth Annual Tcl/Tk Workshop*, Boston, MA, July 14–17, 1997.

[5] Brian Bailey, Joseph A. Konstan, Robert Cooley, and Moses Dejong. Nsync - A Toolkit for Building Interactive Multimedia Presentations. In *Proceedings of ACM Multimedia'98*, pages 257–266, Bristol, UK, September 12–16, 1998.

[6] Tim Berners-Lee, Larry Masinter, and Mark McCahill. Uniform Resource Locators. Request for Comments (Proposed Standard) RFC 1738, Internet Engineering Task Force, December 1994.

[7] Gerold Blakowski, Jens Hübel, Ulrike Langrehr, and Max Mühlhäuser. Tool Support for the Synchronization and Presentation of Distributed Multimedia. *Computer Communications*, 15(10): 611–618, December 1992.

[8] Robert Braden, Lixia Zhang, Steve Berson, Shai Herzog, and Sugih Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. Request for Comments (Proposed Standard) RFC 2205, Internet Engineering Task Force, September 1997.

[9] M. Cecelia Buchanan and Polle T. Zellweger. Automatic Temporal Layout Mechanisms. In *Proceedings of ACM Multimedia'93*, pages 341–350, Anaheim, CA, August 1–6, 1993.

[10] M. Cecelia Buchanan and Polle T. Zellweger. Automatically Generating Consistent Schedules for Multimedia Applications. *Multimedia Systems*, 1(2): 55–67, 1993.

[11] John F. Koegel Buford. *Multimedia Systems.* ACM Press. Addison-Wesley, 1994.

[12] John F. Koegel Buford, L. Rutledge, J. Rutledge, and Keskin C. HyOctane: A HyTime Engine for an MMIS. In *Proceedings of ACM Multimedia'93*, pages 129–136, Anaheim, CA, August 1–6, 1993.

[13] Dick C.A. Bulterman and Robert van Liere. Multimedia Synchronization and UNIX. In *Proceedings of the 2nd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'91)*, Heidelberg, Germany, November 1991.

[14] Dick C.A. Bulterman, Guido van Rossum, and Robert van Liere. A Structure for Transportable, Dynamic Multimedia Documents. In *Proceedings of the Summer USENIX Conference*, pages 137–155, Nashville, TN, 1991.

[15] Vannevar Bush. As We May Think. *The Atlantic Monthly*, 1: 102–108, July 1945.

[16] Andrew T. Campbell, Geoff Coulson, and David Hutchison. A Quality of Service Architecture. *Computer Communication Review (ACM SIGCOMM)*, 24(2), April 1994.

[17] Andrew T. Campbell, Geoff Coulson, and David Hutchison. Supporting Adaptive Flows in Quality of Service Architecture. *Multimedia Systems, Special Issue on QoS Architecture*, May 1998.

[18] Shanwei Cen, Calton Pu, Richard Staehli, Crispin Cowan, and Jonathan Walpole. A Distributed Real-Time MPEG Video Audio Player. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, volume 1018 of *Lecture Notes in Computer Science*, pages 151–162, Durham, NH, April 18–21, 1995. Springer-Verlag.

[19] Shanwei Cen, Calton Pu, Richard Staehli, Crispin Cowan, and Jonathan Walpole. Demonstrating the Effect of Software Feedback on a Distributed Real-Time MPEG Video Audio Player. In *Proceedings of ACM Multimedia'95*, pages 239–240, San Francisco, CA, November 5–9, 1995.

[20] Miguel Correia and Paulo Pinto. Low-Level Multimedia Synchronization Algorithms on Broadband Networks. In *Proceedings of ACM Multimedia'95*, pages 423–434, San Francisco, CA, November 5–9, 1995.

[21] Stephen E. Deering. Host Extensions for IP Multicasting. Request for Comments RFC 1112, Internet Engineering Task Force, August 1989.

[22] Stephen E. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Department of Electrical Engineering, Stanford University, CA, December 1991.

[23] Michael L. Dertouzos. *What Will Be: How the New World of Information Will Change Our Lives*. Harper, 1998.

[24] George D. Drapeau. Synchronization in the MAEstro Multimedia Authoring Environment. In *Proceedings of ACM Multimedia'93*, pages 331–339, Anaheim, CA, August 1–6, 1993.

[25] Andrzej Duda and Chérif Keramane. Structured Temporal Composition of Multimedia Data. In *Proceedings of the IEEE International Workshop on Multimedia Data Base Management Systems*, Blue Mountain Lake, NY, August 1995.

[26] D.J. Duke, D.A. Duce, I. Herman, and G. Faconti. Specifying the PREMO Synchronization Objects. Technical Report ERCIM-02/97-R048, ERCIM, February 1997.

[27] Wolfgang Effelsberg and Thomas Meyer-Boudnik. MHEG Explained. *IEEE Multimedia*, 2(1): 26–38, Spring 1995.

[28] Lynnae Ehley, Borko Furht, and Mohammad Ilyas. Evaluation of Multimedia Synchronization Techniques. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'94)*, pages 514–519, Boston, MA, May 14–19, 1994.

[29] Linda Fédaoui, Aruna Seneviratne, and Éric Horlait. Implementation of a End-to-End Quality of Service Management Scheme. In *Multimedia Transport and Teleservices*, volume 882 of *Lecture Notes in Computer Science*, Vienna, Austria, November 13–15, 1994. Springer-Verlag.

[30] William C. Fenner. Internet Group Management Protocol, Version 2. Request for Comments (Proposed Standard) RFC 2236, Internet Engineering Task Force, November 1997.

[31] R. Fielding. Relative Uniform Resource Locators. Request for Comments (Proposed Standard) RFC 1808, Internet Engineering Task Force, June 1995.

[32] Scott Flinn. Coordinating Heterogeneous Time-Based Media Between Independant Applications. In *Proceedings of ACM Multimedia'95*, pages 435–444, San Francisco, CA, November 5–9, 1995.

[33] Étienne Gagnon. SableCC, an Object-Oriented Compiler Framework. Master's thesis, School of Computer Science, McGill University, Montreal, Quebec, Canada, March 1998.

[34] Marc Gelgon and Patrick Bouthemy. Determining a structured spatio-temporal representation of video content for efficient visualisation and indexing. In *Proceedings of the 5th European Conference on Computer Vision (ECCV'98)*, volume 1, pages 595–609, Fribourg, Germany, June 1998.

[35] Patrick Gros, Roger Mohr, Marc Gelgon, and Patrick Bouthemy. Indexation de vidéos par le contenu. In *Proceedings of CORESA 97*, Issy-les-Moulineaux, France, March 1997.

[36] Rei Hamakawa and Jun Rekimoto. Object composition and playback models for handling multimedia data. *Multimedia Systems*, 2(1): 26–35, February 1994.

[37] C.L. Hamblin. Instants and Intervals. In *Proceedings of the 1st Conference of the International Society for the Study of Time*, pages 324–331, New York, 1972.

[38] Mark Handley. SAP: Session Announcement Protocol. Internet Draft RFC 2327, Internet Engineering Task Force, November 1996. Work in progress.

[39] Mark Handley and Van Jacobson. SDP: Session Description Protocol. Request for Comments (Proposed Standard) RFC 2327, Internet Engineering Task Force, April 1998.

[40] Jonathan Herlocker and Joseph Konstan. Commands as Media: Implementation of a Command Stream. In *Proceedings of ACM Multimedia'95*, San Francisco, CA, November 1995.

[41] Jonathan Herlocker and Joseph Konstan. Tcl Commands as Media in a Distributed Multimedia Toolkit. In *Proceedings of the Third Annual Tcl/Tk Workshop*, Toronto, Canada, July 1995.

[42] Ivan Herman et al. PREMO: An ISO standard for a presentation environment for multimedia objects. In *Proceedings of ACM Multimedia'94*, San Francisco, CA, October 15–20, 1994.

[43] Ivan Herman, Graham J. Reynolds, and James van Loo. PREMO: An emerging standard for multimedia presentation. *IEEE Multimedia*, 3(3 & 4), Autumn & Winter 1996.

[44] Nael Hirzalla, Ben Falchuk, and Ahmed Karmouch. A Temporal Model for Interactive Multimedia Scenarios. *IEEE Multimedia*, 2(3): 24–31, Fall 1995.

[45] Chung-Ming Huang and Ruey-Yang Lee. Quantification of Quality-of-Presentations (QOPs) for Multimedia Synchronization Schemes. *Computer Communication Review (ACM SIGCOMM)*, 26(3): 76–104, July 1996.

[46] Chung-Ming Huang and Ruey-Yang Lee. Quality-of-Presentations (QOP) of the Cyclic Multimedia Synchronization Scheme. *unknown source*, 1997.

[47] Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2: Video. International Standard ISO/IEC 11172-2: 1993, International Organization for Standardization, 1993.

[48] Information technology – Hypermedia Time-based Structuring Language (HyTime). International Standard ISO/IEC 10744: 1997, International Organization for Standardization, 1997.

[49] Video codec for audiovisual services at p×64 kbit/s. Recommendation H.261, Telecommunication Standardization Sector of the International Telecommunication Union, March 1993.

[50] Packet-based multimedia communication systems. Recommendation H.323, Telecommunication Standardization Sector of the International Telecommunication Union, February 1998.

[51] Muriel Jourdan, Nabil Layaïda, and Loay Sabry-Ismail. MADEUS: an authoring environment for multimedia documents. In *Proceedings*

of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'97), Ottawa, Canada, June 3–6, 1997.

[52] Muriel Jourdan, Nabil Layaïda, and Loay Sabry-Ismail. Time Representation and Management in MADEUS: an authoring environment for multimedia documents. In *Multimedia Computing and Networking 1997, IS&T/SPIE Proceedings*, volume 3020, pages 68–79, San Jose, CA, February 8–14, 1997.

[53] Dilip D. Kandlur, Debanjan Saha, and Mark Willebeek-LeMair. Protocol Architecture for Multimedia Applications over ATM Networks. *Computer Communication Review (ACM SIGCOMM)*, 25(3): 33–43, July 1995.

[54] Ahmed Karmouch and James Emery. A Playback Schedule Model for Multimedia Documents. *IEEE Multimedia*, 3(1): 50–61, Spring 1996.

[55] Chérif Keramane. *Spécification de présentations multimédia structurées.* PhD thesis, Institut National Polytechnique de Grenoble, France, November 4, 1997.

[56] Chérif Keramane and Andrzej Duda. Interval Expressions - a Functional Model for Interactive Dynamic Multimedia Presentations. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'96)*, Hiroshima, Japan, June 1996.

[57] Michelle Y. Kim and Junehwa Song. Multimedia Documents with Elastic Time. In *Proceedings of ACM Multimedia'95*, pages 143–154, San Francisco, CA, November 5–9, 1995.

[58] Donald E. Knuth. *The TEXbook.* Computers and Typesetting. Addison-Wesley, 1986.

[59] Nabil Layaïda. Issues in Temporal Representation of Multimedia Documents. In *W3C Workshop on Real Time Multimedia and the Web 96 (RTMW'96)*, Sophia Antipolis, France, October 24–25, 1996.

[60] Nabil Layaïda. *Madeus : système d'édition et de présentation de documents structurés multimédia.* PhD thesis, Université Joseph Fourier, Grenoble, France, June 12, 1997.

[61] Nabil Layaïda and Loay Sabry-Ismail. Maintaining Temporal Consistency of Multimedia Documents using Constraint Networks. In *Multimedia Computing and Networking 1996, IS&T/SPIE Proceedings*, volume 2667, pages 124–135, San Jose, CA, January 29–31, 1996.

[62] Didier Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4): 46–58, April 1991.

[63] Lian Li, Ahmed Karmouch, and Nicolas D. Georganas. Multimedia Teleorchestra with Independant Sources: Part 1 — Modelling of Collaborative Multimedia Scenarios. *IEEE Multimedia*, 1(4): 143–153, Spring 1994.

[64] Christopher J. Lindblad, David J. Wetherall, and David L. Tennenhouse. The VuSystem: A Programming System for Visual Processing of Digital Video. In *Proceedings of ACM Multimedia'94*, pages 307–314, San Francisco, CA, October 15–20, 1994.

[65] Ming Liou. Overview of the p×64 kbits/s Video Coding Standard. *Communications of the ACM*, 34(4): 59–63, April 1991.

[66] Thomas D.C. Little and Arif Ghafoor. Synchronization and Storage Models for Multimedia Objects. *IEEE Journal on Selected Areas in Communications*, 8(3): 413–427, April 1990.

[67] Thomas D.C. Little and Arif Ghafoor. Interval-Based Conceptual Models for Time-Dependent Multimedia Data. *IEEE Transactions on Knowledge and Data Engineering, Special Issue on Multimedia Systems*, 5(4): 551–563, August 1993.

[68] Colin Low and Laurent Babarit. Distributed 3D audio rendering. In *Proceedings of the 7th International World Wide Web Conference (WWW7), Computer Networks and ISDN Systems*, volume 30, pages 407–415, Brisbane, Australia, April 14–18, 1998.

[69] Guojun Lu. *Communication and Computing for Distributed Multimedia Systems*. Artech House, 1996.

[70] Nelson R. Manohar and Atul Prakash. Dealing with Synchronization and Timing Variability in the Playback of Session Recordings. In *Proceedings of ACM Multimedia'95*, pages 45–56, San Francisco, CA, November 5–9, 1995.

[71] Jean Michel Marie-Julie and Hassane Essafi. Image Indexing by Using a Rotation and Scale Invariant Partition. In *Proceedings of the 3rd European Conference on Multimedia Applications, Services and Techniques (ECMAST'98)*, volume 1425 of *Lecture Notes in Computer Science*, pages 163–176, Berlin, Germany, May 26–28, 1998. Springer-Verlag.

[72] Laurent Mathy and Olivier Bonaventure. QoS Negotiation for Multicast Communications. In *Multimedia Transport and Teleservices*, volume 882 of *Lecture Notes in Computer Science*, pages 199–218, Vienna, Austria, November 13–15, 1994. Springer-Verlag.

[73] Steven McCanne, Eric Brewer, Randy Katz, Lawrence Rowe, Elan Amir, Yatin Chawathe, Alan Coopersmith, Ketan Mayer-Patel, Suchitra Raman, Angela Schuett, David Simpson, Andrew Swan, Teck-Lee Tung, David Wu, and Brian Smith. Toward a Common Infrastructure for Multimedia-Networking Middleware. In *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*, St. Louis, MO, May 1997.

[74] Steven McCanne and Van Jacobson. vic: A Flexible Framework for Packet Video. In *Proceedings of ACM Multimedia'95*, pages 511–522, San Francisco, CA, November 5–9, 1995.

[75] Françoise Mocellin. *Gestion de données et de présentations multimédias par un SGBD à objets*. PhD thesis, Université Joseph Fourier, Grenoble, France, December 12, 1997.

[76] Roger Mohr, Patrick Gros, Bart Lamiroy, Sylvaine Picard, and Cordelia Schmid. Indexation et recherche d'images. In *Proceedings of the 16ème colloque GRETSI sur le traitement du signal et des images*, Grenoble, France, September 1997.

[77] Nicholas Negroponte. *Being Digital.* Alfred A. Knopf, 1995.

[78] Nicholas Negroponte. *L'Homme Numérique.* Robert Laffont, 1995. Édition française de [77].

[79] Steven R. Newcomb, Neill A. Kipp, and Victoria T. Newcomb. The "HyTime": hypermedia/time-based document structuring language. *Communications of the ACM*, 34(11): 67–83, November 1991.

[80] Jason Nieh and Monica S. Lam. The Design and Evalutation of SMART: A Scheduler for Multimedia Applications. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, St. Malo, France, October 5–8, 1997.

[81] Yann Orlarey and Hervé Lequay. MidiShare: A real time multi-tasks software module for Midi applications. In *Proceedings of the International Computer Music Conference*, pages 234–237, Computer Music Association, San Francisco, CA, 1989.

[82] John K. Ousterhout. *Tcl and the Tk Toolkit*. Professional Computing Series. Addison-Wesley, 1994.

[83] François Pachet and Olivier Delerue. MidiSpace: a Temporal Constraint-Based Music Spatializer. In *Proceedings of ACM Multimedia'98*, pages 351–359, Bristol, UK, September 12–16, 1998.

[84] Michael Papathomas. ATOM: An Active Object Model for Enhancing Reuse in the Development of Concurrent Software. Technical Report RR 963-I-LSR-2, LSR - IMAG, Grenoble, France, November 1996.

[85] Ken C. Pohlmann. *Principles of Digital Audio*. McGraw-Hill, third edition, 1995.

[86] B. Prabhakaran and S.V. Raghavan. Synchronisation models for multimedia presentation with user participation. *Multimedia Systems*, 2(2): 53–62, August 1994.

[87] Roger Price. MHEG: An Introduction to the future International Standard for Hypermedia Object Interchange. In *Proceedings of ACM Multimedia'93*, pages 121–128, Anaheim, CA, August 1–6, 1993.

[88] Srinivas Ramanathan and P. Venkat Rangan. Feedback Techniques for Intra-Media Continuity and Inter-Media Synchronization in Distributed Multimedia Systems. *The Computer Journal, Special Issue on Distributed Multimedia Systems*, 36(1): 19–31, February 1993.

[89] P. Venkat Rangan, Srinivas Ramanathan, Harrick M. Vin, and Thomas Kaeppner. Techniques for Multimedia Synchronization in Network File Systems. *Computer Communications Journal*, 16(3): 168–176, March 1993.

[90] Cécile Roisin and Irène Vatton. Merging Logical and Physical Structures in Documents. *Electronic Publishing – Origination, Dissemination and Design, special issue Proceedings of the Fifth International Conference on Electronic Publishing, Document Manipulation and Typography (EP'94)*, 6(4): 327–337, April 1994.

[91] Reza Rooholamini and Vladimir Cherkassky. ATM-Based Multimedia Servers. *IEEE Multimedia*, 2(1): 39–52, Spring 1995.

[92] Jonathan Rosenberg, Eve Schooler, Henning Schulzrinne, and Mark Handley. SIP: Session Initiation Protocol. Internet draft, Internet Engineering Task Force, August 1998. Work in progress.

[93] Franck Rousseau and Andrzej Duda. An Execution Architecture for Synchronized Multimedia Presentations. In *Proceedings of the 3rd European Conference on Multimedia Applications, Services and Techniques (ECMAST'98)*, volume 1425 of *Lecture Notes in Computer Science*, pages 42–55, Berlin, Germany, May 26–28, 1998. Springer-Verlag.

[94] Franck Rousseau and Andrzej Duda. Playing Temporal HTML Documents (Poster presentation and poster paper). In *Posters booklet, ACM Multimedia'98*, Bristol, UK, September 12–16, 1998.

[95] Franck Rousseau and Andrzej Duda. Synchronized Multimedia for the WWW. In *Proceedings of the 7th International World Wide Web Conference (WWW7), Computer Networks and ISDN Systems*, volume 30, pages 417–429, Brisbane, Australia, April 14–18, 1998.

[96] Lawrence A. Rowe and Brian C. Smith. A Continuous Media Player. In *Proceedings of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'92)*, volume 712 of *Lecture Notes in Computer Science*, pages 376–386, San Diego, CA, November 1992. Springer-Verlag.

[97] Lloyd W. Rutledge, Lynda Hardman, Jacco van Ossenbruggen, and Dick C.A. Bulterman. Structural Distinctions Between Hypermedia Storage and Presentation. In *Proceedings of ACM Multimedia'98*, pages 145–150, Bristol, UK, September 12–16, 1998.

[98] James Schnepf, Joseph Konstan, and David H.C. Du. Doing FLIPS: Flexible Interactive Presentation Synchronization. *IEEE Journal on Selected Areas in Communications*, January 1996.

[99] Henning Schulzrinne. RTP Profile for Audio and Video Conferences with Minimal Control. Request for Comments (Proposed Standard) RFC 1890, Internet Engineering Task Force, May 1996.

[100] Henning Schulzrinne. A comprehensive multimedia control architecture for the Internet. In *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*, St. Louis, MO, May 1997.

[101] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. Request for Comments (Proposed Standard) RFC 1889, Internet Engineering Task Force, February 1996.

[102] Henning Schulzrinne, Anup Rao, and Rob Lanphier. Real Time Streaming Protocol (RTSP). Request for Comments (Proposed Standard) RFC 2326, Internet Engineering Task Force, April 1998.

[103] Ralf Steinmetz. Analyzing the Multimedia Operating System. *IEEE Multimedia*, 2(1): 68–84, Spring 1995.

[104] Ralf Steinmetz and Klara Nahrstedt. *Multimedia: computing, communications and applications*. Innovative Technology Series. Prentice Hall, 1995.

[105] Emilia Stoica, Hussein Abdel-Wahab, and Kurt Maly. Synchronization of Multimedia Streams in Distributed Environments. Technical Report TR-97-19, Department of Computer Science, Old Dominion University, Norfolk, Va 23529, February 1997.

[106] P.D. Stotts and R. Furuta. Temporal Hyperprogramming. *Journal of Visual Languages and Computing (Academic Press)*, 1(3): 237–253, September 1990.

[107] Thierry Turletti. H.261 software codec for videoconferencing over the Internet. Technical Report 1834, INRIA, Sophia-Antipolis, France, January 1993.

[108] Jim Urbas and Mark Pfeiffer. Enhancements to HTML for implementing animation effects. Term project for CS-790: Advanced Special Topics - Multimedia Systems, University of Wisconsin, Milwaukee, May 1995.

[109] Guido van Rossum, Jack Jansen, K. Sjoerd Mullender, and Dick C.A. Bulterman. CMIFed: A Presentation Environment for Portable Hypermedia Documents. In *Proceedings of ACM Multimedia'93*, pages 183–188, Anaheim, CA, August 1–6, 1993.

[110] Andrés Vega-García. *Mécanismes de contrôle pour la transmission de l'audio sur l'Internet*. PhD thesis, Université de Nice-Sophia Antipolis, France, October 30, 1996.

[111] Cascading Style Sheets, level 2. Recommendation REC-CSS2-19980512, World Wide Web Consortium (W3C), May 12, 1998. Latest version available at `http: //www.w3.org/TR/REC-CSS2/`.

[112] HyperText Markup Language. Recommendation REC-html40-19980424, World Wide Web Consortium (W3C), April 24, 1998. Latest version available at `http: //www.w3.org/TR/REC-html40/`.

[113] Synchronized Multimedia Integration Language. Recommendation REC-smil-19980615, World Wide Web Consortium (W3C), June 15, 1998. Latest version available at `http: //www.w3.org/TR/REC-smil/`.

[114] Thomas Wahl and Kurt Rothermel. Representing Time in Multimedia Systems. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'94)*, pages 538–543, Boston, MA, May 14–19, 1994.

[115] WAP WAE Specification (Wireless Application Protocol, Wireless Application Environment Specification). Technical report, Wireless Application Protocol Forum, April 1998. Latest version available at `http: //www.wapforum.org/`.

[116] Ron Weiss, Andrzej Duda, and David K. Gifford. Content-Based Access to Algebraic Video. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'94)*, pages 140–151, Boston, MA, May 14–19, 1994.

[117] Y. Yusoff, W. Christmas, and J. Kittler. A Study on Automatic Shot Change Detection. In *Proceedings of the 3rd European Conference on Multimedia Applications, Services and Techniques (ECMAST'98)*, volume 1425 of *Lecture Notes in Computer Science*, pages 177–189, Berlin, Germany, May 26–28, 1998. Springer-Verlag.

Présentations Multimédia Synchronisées pour le WWW

Résumé

Les récentes avancées techniques dans les domaines du logiciel, du matériel et des réseaux ont permis l'apparition du multimédia dans des environnements informatiques classiques. Il devient aujourd'hui envisageable d'utiliser le World Wide Web comme support de diffusion de contenu multimédia. Des extensions temporelles au langage HTML sont proposées sous la forme de quelques nouveaux *tags* permettant la description du comportement temporel et spatial, afin de pouvoir spécifier des présentations multimédia pour le WWW incluant plusieurs objets média. Une architecture de synchronisation a été conçue et mise en œuvre en Java, celle-ci permettant de jouer des présentations complexes définies à l'aide de ces extensions dans un navigateur WWW. Les extensions à HTML sont analysées et transformées en code HTML classique et Java, produisant ainsi un document qu'il est possible de lire dans un navigateur standard. Enfin, le cas des sources de contenu sur réseau délivrant des flots contrôlés par l'architecture de synchronisation a été considéré à partir de l'analyse de la problématique du transport des données multimédia temporelles dans un environnement *best effort* distribué comme l'Internet. À terme l'objectif est de fournir une qualité de présentation optimale en traitant les problèmes au niveau le plus adapté dans la chaîne serveur-transport-client.

Mots clés : multimédia, synchronisation, présentation, WWW, HTML, Java, transport

---

Synchronized Multimedia Presentations for the WWW

Abstract

Recent advances in hardware, software and networks have allowed seamless integration of multimedia within standard computing environments. Currently, the World Wide Web is a framework of choice for handling and disseminating multimedia content. To integrate multimedia within WWW documents, we have proposed temporal extensions to the HTML language that allow to specify the temporal and spatial behavior of documents using new tags. In this way, the HTML documents become multimedia presentations composed of several media objects. We have designed and implemented a synchronization architecture to play such multimedia presentations. A document specified using the HTML extensions is parsed and compiled into standard HTML and Java code, so that it can be shown in a WWW browser. We have also considered the problem of distributed multimedia sources that deliver streams synchronized within a multimedia presentation. We have defined an architecture for negotiating and controlling the transport of temporally sensitive data in a best effort network such as the current Internet. The objective of the architecture is to provide optimal quality of presentation by controlling temporal problems at the most suitable level in the server-transport-client chain.

Keywords : multimedia, synchronization, presentation, WWW, HTML, Java, transport

---

Discipline : Informatique, Systèmes et Communications

Laboratoire : LSR-IMAG – Équipe Drakkar
BP 72, 38402 Saint Martin d'Hères Cedex, France