الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGÉRIENNE DEMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENTS SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université SAAD DAHLAB de Blida

FACULTE DES SCIENCES

DÉPARTEMENT D'INFORMATIQUE

MÉMOIRE

En vue de l'obtention du grade d'Ingénieur d'Etat
Spécialité : Informatique
Option : Software

THÈME

Editeur graphique pour l'édition de documents multimédias cohérent au format SMIL

Présenté par :
BENTAHAR HAKIM.
BENTAHAR NAOUEL.

Soutenu devant le jury composé de :

Président Mr : Hadj Yahia
Examinatrice Melle : Aoussat

Promotrices :

Mme: Bensettiti .S
Mme: Dahmani .D

Année Universitaire 2003 / 2004

Remerciements

En premier lieu, nous remercions DIEU tout puissant de nous avoir donné le courage pour réaliser ce travail.

Nous tenons ensuite à remercier nos promotrices Mme BENSETTITI pour nous avoir accordé de son temps et nous avoir proposé un sujet aussi passionnant, et Mme DAHMANI pour avoir endossé notre projet et de nous avoir guidé durant notre travail et supporté durant nos démarches d'ingénierie. Les pages qui suivent doivent beaucoup à leurs conseils et leurs critiques. Nous espérons avoir été à la hauteur de la confiance qu'elles ont mise en nous.

Un remerciement particulier pour Melle YAHIAOUI Khaoula et Ahlem et Mer MEZZAOUROU Brahim qui nous ont été de grande aide.

Nous ne saurons exprimer tout l'amour et la reconnaissance que nous avons pour nos parents, pour leur soutien qu'ils nous ont apporté, pour nous avoir donné les moyens depuis le début de réaliser cette thèse. Nous tenons à leur dire que chaque jour qui passe nous réalisons un peu plus notre chance et tout ce qu'ils ont fait pour nous.

Enfin, nous remercions toute personne ayant participé de près ou de loin à la réalisation de ce travail.

Dédicaces

Nous dédions ce modeste travail :

À nos très chers parents à qui nous exprimons toute notre gratitude pour leur soutien tout au long de nos études et leur aide pour réaliser ce travail. Nous espérons avoir été à la hauteur de votre confiance, que dieu vous garde.

À nos frères et sœur Yacine, Rayanne et Amira.

À tous les membre de notre famille.

À toute la famille YAHIAOUI.

Je le dédie :

À tous mes amis, en particulier Lamine, Mohamed, Hichem, Kaouthar,... et tous ceux que je n'ai pas cité.

Hakim

Je le dédie :

À tous mes amis, en particulier Nabiha, Samia, Akila, Kelthoum, Fateh,... et tous ceux que je n'ai pas cité.

Naouel ...

Hakim et Naouel

Chapitre II : LA DESCRIPTION DU LANGAGE SMIL LA PREMIÈRE VERSION SMIL 1.0

Chapitre III : LES TECHNIQUES D'ANALYSE DES DOCUMENTS MULTIMÉDIAS

Chapitre IV : CONCEPTION DE NOTRE MODÈLE

Sommaire

Sommaire

Liste des figures

Liste des tableaux

Abréviations et acronymes

Introduction

Chapitre I : ÉDITION DE DOCUMENTS MULTIMÉDIAS

Chapitre V : DESCRIPTION DE SMIL GEN

Chapitre VI : Réalisation de Smil_Gen

Conclusion

Bibliographie et références

Liste des figures

Liste des tableaux

Abréviations et acronymes

CSP	Contraint Satisfaction Problem.
HTML	HyperText Markup Language.
ISO	International Standards Organization.
RT-LOTOS	RealTime – Language of Temporal Ordering Specification.
SMIL	Synchronized Multimédia Intégration Language.
Smil_Gen	Générateur du code SMIL (Smil Generator).
STP	Simple Temporel Problem.
URI	Universal Ressource Locator.
URL	Uniform Ressource Identifier.
W3C	World Wide Web Consortium.
WWW	World Wide Web.
WYSIWYG	What You See Is What You Get.
XML	eXtensible Markup Language.

INTRODUCTION GÉNÉRALE

INTRODUCTION GÉNÉRALE

De nos jours, l'accès à des informations de manière plus performante et efficace devient possible grâce à l'évolution technologique associée aux réseaux de communication. Les documents multimédia occupent une grande partie de ces informations.

Le terme multimédia signifie l'existence de plusieurs médias associés les uns avec les autres par des contraintes de synchronisation temporelles. Donc un document est qualifié de multimédia s'il contient des médias dont l'un est au moins de nature temporisée, comme le son, la vidéo, l'image, le texte,... etc.

Jusqu'à présent, les documents ont été abordés essentiellement sous l'angle de leur structure spatiale. Mais avec l'arrivée du multimédia, la description d'un document multimédia doit prendre en compte différents aspects tels que l'aspect logique et temporel. La structure temporelle décrit l'enchaînement des éléments dans le temps. Ainsi l'aspect temporel d'un document multimédia peut être exprimé sous la forme de relations temporelles entre composants du document multimédia (synchronisation inter-média), et à l'intérieur de l'objet média temporisé (synchronisation intra-média) [SAM03].

L'information traitée par un document multimédia est plus riche et immensément plus explicite par rapport à un document conventionnel. Ainsi les concepteurs, qui utilisaient du texte, du son ou des images figées pour le graphique, ont finalement opté pour les nouvelles possibilités offertes par ce type de données multimédias. De ce fait la création de nouveaux outils multimédias est devenue indispensable.

Parmi les langages de description, nous pouvons citer le langage SMIL (Synchronized Multimedia Integration Language).

Le langage SMIL, est le résultat des travaux entrepris par l'INRIA (Institut National de Recherche en Informatique). Il a été défini par le W3C en 1998 dans le but d'améliorer la fonctionnalité et l'interopérabilité du WEB. SMIL est un langage accessible et puissant pour la conception et la présentation de documents multimédias, il permet de spécifier des documents multimédias intégrant plusieurs types de médias locaux ou résidant sur des serveurs distants.

A l'aide de SMIL, un auteur peut décrire le comportement temporel d'une présentation multimédia, associer des liens hypertextes à des objets médias et décrire la disposition de la présentation sur un écran.

Néanmoins, la présentation d'un document dépend particulièrement, et en grande partie sur la cohérence absolue de ses contraintes de synchronisation. Mais, pour des présentations plus au moins complexes, l'édition basée sur les langages de description peut engendrer des situations d'incohérence. Ainsi, ces incohérences sont interprétées diversement, sachant que chaque lecteur a son propre mécanisme de contrôle (CSP, RT-LOTOS…).

Le travail qui nous a été confié consiste à concevoir un éditeur que nous avons nommé ***Smil_Gen*** intégrant n'importe quel type de média et un système de détection d'incohérences temporelles dans des documents multimédias au format SMIL ; ce système se situe après la phase d'édition. La solution proposée pour la détection d'incohérences temporelles est basée sur l'approche des CSP (Constraints Satisfaction Problem) : une solution qui se rapproche le plus du document multimédia, vu que les relations temporelles entre les médias sont naturellement exprimables avec les CSP. L'approche basée sur les CSP consiste à traduire un document SMIL en un système de contraintes de type STP (Simple Temporel Problem) qui est une classe particulière des CSP. Ce système sera en suite représenté par un graphe temporel. Ainsi, l'analyse de ce graphe permettra la vérification de la cohérence du document.

Plan du mémoire

Ce mémoire est organisé en six chapitres, dont le contenu de chacun sera détaillé ci-dessous :

Dans le *chapitre* I, on présentera une étude globale sur le processus d'édition des documents multimédias, puis on développera les différentes représentations que peut prendre un document au cours d'un processus d'édition. On présentera par la suite les besoins liés aux auteurs, avant d'achever ce chapitre avec les différents langages et outils multimédias.

Dans le *chapitre* II, on étudiera le langage SMIL. Cette étude consiste à présenter SMIL1.0 la première version du langage, en présentant sa syntaxe et les principales possibilités offertes par ce langage.

Dans le *chapitre* III, on parlera des techniques d'analyse des documents multimédias. On abordera deux approches pour la vérification de la cohérence des documents multimédias. En l'occurrence, la technique des CSP (Contraints Satisfaction Problem) et le système RT-LOTOS (Real-Time Language Temporel Ordering Specification).

Le *chapitre* IV sera consacré à la conception de notre modèle temporel. La première partie consiste à traduire un code SMIL en un graphe temporel. La deuxième partie, pour sa part, est consacrée à la vérification de la cohérence des documents SMIL.

Le *chapitre* V est dédié à la description de Smil_Gen. Nous essayerons de donner les meilleurs aspects possibles à notre outil.

Dans le *chapitre* VI on exposera la réalisation de ***Smil_Gen.*** Ainsi, nous donnerons un aperçu global sur le concept de notre architecture, et nous parlerons de la réalisation de quelques services d'édition.

Dans la *conclusion* générale, nous présenterons l'apport de certaines perspectives de notre travail.

Afin de faciliter la lecture, nous précisons que l'état de l'art se retrouve à trois niveaux :

- dans le chapitre I pour l'édition des documents multimédia;
- dans le chapitre II pour l'étude du langage SMIL1.0;
- dans le chapitre III pour les techniques d'analyse des documents multimédias;

Et nos contributions se trouvent à trois niveaux :

- la conception du modèle temporel, dans le chapitre IV;
- la description de notre outil ***Smil_Gen*** présenté dans le chapitre V;
- la réalisation de ***Smil_Gen*** exposé dans le chapitre VI.

Chapitre I

ÉDITION DE DOCUMENTS MULTIMÉDIAS

Chapitre I

ÉDITION DE DOCUMENTS MULTIMÉDIAS

Historique

A chaque fois que l'homme manipule de l'information, l'ordinateur peut l'aider, c'est pourquoi l'informatique s'est largement étendue vue l'essor considérable qu'a pris le développement du multimédia. Avec l'arrivée du Web, l'évolution du matériel, les exigences en traitement et en diffusion de l'information sont de plus en plus importantes.

De nos jours de nombreux travaux sont développés autour de ce thème qui a envahie le monde de l'informatique; Ces travaux concernent la création des périphériques adaptés à la présentation de ces informations, ou la spécification de standards pour encoder l'information et favoriser ainsi les échanges entre les personnes, la définition de réseaux haut débit pour transporter l'information.

Mais cette évolution dans le domaine des documents multimédias n'a pas avancé en parallèle avec l'informatique, car peu de travaux portent sur *l'édition* de ces informations. L'édition consiste à classer et intégrer un ensemble de données numériques, appelées *médias*, de nature diverse (son, vidéo, image, texte…).

Un pareil processus est indispensable lors de la création ou le téléchargement des médias puis leur assemblage pour l'écriture, par exemple, une page Web ou un document purement multimédia. Donc de la même manière, n'importe quel organisme qui veut utiliser le Web ou un document multimédia pour diffuser de l'information, qui doit être structurée et mise à jour régulièrement, sera confronté à une phase d'édition.

L'arrivée d'informations de plus en plus riches, rend les documents qui les contiennent de plus en plus complexes et difficiles à spécifier. De documents purement textuels il y a quelques années, on est arrivé aujourd'hui à des documents interactifs et temporisés. On appellera document multimédia, tout document composé de médias de différents types, et dont la présentation comporte une composante spatiale, hypertexte mais aussi temporelle.

Il devient donc indispensable que l'auteur soit aidé dans cette tâche d'édition par un logiciel. Ces logiciels seront appelés par la suite ***environnements auteur*** ou ***environnements d'édition***.

Afin de rendre la création de documents multimédias accessible à tous, il est nécessaire de proposer des environnements auteur permettant à chacun de nous, et cela sans avoir de compétences informatiques particulières, d'éditer des documents multimédias.

I.1. Étapes de conception et d'édition de documents multimédias

Lors de l'élaboration d'un document multimédia, que se soit dans un contexte professionnel ou familial, le processus affecté à cette fonction suit les mêmes étapes, même si certaines de ces étapes sont plus ou moins complexes et développées en fonction du contexte. Par exemple, si on fait le parallèle avec l'écriture de documents HTML[1], dans un contexte familial l'auteur crée empiriquement la structure de son document alors que dans le cas d'un site professionnel cette structure est formalisée préalablement.

Le processus de conception d'un document multimédia peut être décomposé précisément en sept étapes distinctes : [LAY 97]

1. ***Apparition d'une idée*** *de* document à produire.
2. ***Clarification de cette idée*** *:* Une fois l'idée claire, il est nécessaire de dégager des informations comme les objectifs de la présentation, et toute information essentielle pour la conception du document, comme le type de public concerné, les fonctionnalités attendues, le choix de support de communication utilisé.
3. ***Écriture du scénario*** **:** Dans la rédaction du scénario, on précise les moyens et les méthodes pour les quelles on arrive à résoudre nos objectifs, par exemple la stratégie utilisée pour l'exploration des informations ou la navigation dans le document.

[1] Hypertext Markup Language

4. ***Écriture du scénarimage* (story-board) :** Á cette étape, on décrit de manière visuelle le résultat graphique attendu, et cela permet d'identifier les médias utilisés ainsi que l'interactivité entre eux-mêmes.
5. ***Réalisation de la maquette* :** Après le scénarimage, on réalise une maquette qui permet de tester un scénario et d'avoir une première idée du document dans sa forme finale. À partir de cette maquette, on peut faire des modifications si cela semble nécessaire sur des choix faits précédemment ; cette tâche est réalisée par des informaticiens si le scénario s'avère complexe ou s'il est commerciale..
6. ***Production du document* :** Á la fin des étapes précédentes, il ne reste plus qu'à produire le document. Cette phase se décompose en deux sous tâches :
   - Création ou collecte des médias qui composeront le document.
   - Assemblage des différents médias pour constituer le document.
7. ***Diffusion*** : La dernière étape concerne la diffusion du document une fois produit, cette diffusion peut être réalisée par l'intermédiaire de CD-ROM ou via le Web.

I.2. Représentation d'un document multimédia dans un contexte d'édition

Pour bien comprendre ce chapitre, on va développer les différentes représentations que peut prendre un document au cours d'un processus d'édition :

- ***La structure interne du document* :** Elle représente le document au sein de l'environnement auteur. Le système d'édition permettra à l'auteur de manipuler cette structure de données à travers de fonctions d'édition. On appellera par la suite *formalisme d'édition* l'ensemble des informations que l'auteur perçoit de la structure interne du document.
- ***Format de sauvegarde* :** Il permet de stocker le document dans un fichier. Ce format peut être un format propriétaire du système auteur comme par exemple (Madeus [LAY 97], Director [MAC-DIR 00]) ou il peut être un des standards de sauvegarde (SMIL [SMI 98]). Ce fichier contient les informations liées à l'édition du document et à l'environnement auteur, mais il peut aussi contenir les informations de présentation du document.

- ***Format de restitution*** **:** Il représente le document dans une ou plusieurs formes, distribuée(s) aux lecteurs. Dans le cas d'une diffusion pour plusieurs périphériques[2] deux types de distribution sont possibles :

1) La première solution consiste à distribuer un fichier par périphérique, c'est la solution communément utilisée aujourd'hui, puisque dans ce cas, chaque périphérique a un format spécifique à lui-même.
2) La deuxième solution consiste à fournir un fichier qui contient les informations pour plusieurs périphériques. On dit dans ce cas, que le document est *adaptable*, puisqu'il s'adapte à au moins un périphérique.

   On distingue deux types de fichier de restitution :

   - Le premier est un programme directement exécutable par le système d'exploitation de la machine ou interprétable par une machine virtuelle (Java).
   - Le deuxième est un fichier interprétable par un logiciel spécifique :

     1) Un fichier textuel écrit dans une forme déclarative (HTML, SMIL, Madeus). L'émergence des standards comme XML [XML 98] permet d'uniformiser la manière de décrire des structures de données et en particulier des documents.
     2) Un fichier binaire (AVI, MPEG). Ce fichier de présentation doit contenir les informations de présentation, mais il peut aussi contenir les informations d'édition qui ont mené à cette présentation.

Le rôle d'un environnement d'édition est donc d'offrir les représentations les plus proches possibles les unes des autres pour permettre une édition simple tout en favorisant la pérennité du document. Au cours du processus d'édition, les auteurs pourront manipuler et éditer chacune de ces représentations.

On peut distinguer dans chacune de ces représentations deux entités : une entité donnée (ou contenue) et une entité présentation (ou composition).

- *Donnée* : elle comprend différents médias qui peuvent être statiques ou dynamiques, structurés en (SMIL, HTML…) ou non. La collecte ou la création de ces médias se fait lors de la phase de production du document.
- *Présentation* : cette entité définit les interactions des différents médias les uns avec les autres ainsi qu'avec l'utilisateur. Cette entité comprend la synchronisation spatiale, temporelle et hypermédia mais aussi les interactions du lecteur avec le document. Lors du processus d'édition ces informations sont initialement décrites dans le scénario avant d'être traduites dans le langage utilisé lors de la phase de production.

[2] Téléphones mobiles, ordinateurs, assistants personnels…

I.3. L'édition de documents multimédias [LAY 97]

I.3.1. Définition du processus de création d'un document multimédia

Un système d'édition de documents multimédia comporte principalement deux types de fonctions :

- *Des fonctions d'édition* : elles réalisent les opérations de création, de construction et de modification du document par un auteur. L'opération de composition d'un document consiste à y inclure des objets multimédia de base, ensuite à spécifier des relations entre ces différents éléments. Ces relations peuvent être liées à leur organisation logique comme par exemple une scène composée de trois images et d'un accompagnement musical, à leur disposition spatiale, ou à leur synchronisation temporelle.
- *Des fonctions de présentation* : elles permettent de restituer à un utilisateur le contenu du document une fois que son édition est accomplie. Cette phase consiste à fournir au lecteur un ensemble de commandes permettant d'explorer ou de naviguer dans le document pour découvrir l'information qu'il contient à travers l'espace, le temps et l'interaction avec le document. Les commandes fournies au lecteur du document lui permettent l'arrêt d'une présentation multimédia, son redémarrage, son avance et retour rapide ainsi que la possibilité d'interagir avec certains éléments du document. Ce type d'interaction est aussi fondé sur des liens hypermédias, sur des boutons de présentation contenus dans le document, ou de façon générale sur tous les éléments du document qui l'admettent, comme les éléments multimédia programmés (*applets*)

Ainsi la création d'un document multimédia est un processus complexe qui ne fait pas forcément intervenir un environnement auteur.

On peut distinguer trois principales manières de créer un document multimédia (illustrées par la *Figure I-1*) :

A. *Édition via un environnement auteur*

L'outil offre un support à l'auteur via une interface graphique et un ensemble de services, par exemple :

- Visualisation du document ou de certains aspects du document en cours de construction, en offrant différentes vues comme une vue de la structure, ou de l'organisation temporelle.

- Indépendance de l'auteur vis-à-vis la syntaxe du langage dans lequel sera sauvegardé le document. C'est le cas par exemple des éditeurs de documents au format HTML qui permettent à l'auteur d'écrire des documents sans se soucier de la syntaxe de ce langage.
- Automatisation de certaines tâches : maintenance des liens, vérification de la cohérence.

B. ***Édition via un éditeur de texte***

Ce type d'édition est encore souvent utilisé, car les auteurs ne trouvent pas d'environnement d'édition qui satisfasse pleinement leurs besoins.

C. ***Génération automatique de documents***

Dans ce type de processus, l'intervention de l'auteur est minime voire inexistante. La production du document se fait par génération à partir d'un programme. Ce sont par exemple :

- Les documents construits pour présenter au lecteur le résultat d'une requête à une base de données. Le système doit alors ordonner les résultats et les présenter au lecteur.
- Les documents qui sont produits par traduction d'autres documents.

Les générateurs de documents utilisent aujourd'hui la production automatique de documents. À partir d'un ensemble de données, le générateur produit des informations de présentation pour permettre aux lecteurs de les visualiser. Ce mode de création n'est pas adapté à une édition interactive.

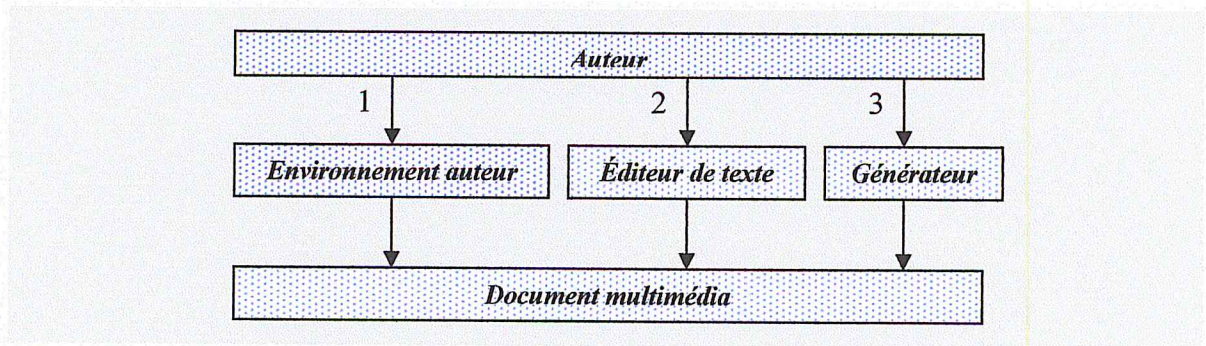


Figure I-1 : Les trois principaux modes de création de documents multimédias.

Notre objectif est d'apporter une réponse pertinente au problème de l'édition interactive de documents multimédias en se plaçant dans le contexte d'un environnement auteur. On espère ainsi éviter le plus possible l'écriture de documents multimédias par l'intermédiaire d'éditeurs textuels. En particulier, il convient de porter une attention particulière au fait

qu'avec l'émergence d'XML [XML 98], de nombreux langages apparaissent. Il est donc nécessaire d'imaginer des environnements auteur qui soient capables de gérer cette multiplicité de formats. Donc, l'objectif est tout d'abord d'identifier les besoins des auteurs de documents multimédias et de confronter ces besoins aux langages et outils existants.

I.3.2. Analyse des besoins pour la création multimédia [TAR 97]

Concernant l'environnement auteur, l'auteur manipule indirectement le langage de sauvegarde. La structure de donnée interne de l'application est une représentation de ce fichier dont le rôle est de simplifier la tâche d'édition de l'auteur. L'environnement auteur a pour responsabilité d'assister l'auteur dans cette édition.

Pour mieux analyser les besoins pour la création d'un document multimédia, il est donc nécessaire de séparer au cours du processus d'édition, les besoins couverts par le langage de sauvegarde et les besoins couverts par les environnements :

- Les besoins couverts par les environnements : plus précisément on identifiera les services que doivent fournir de tels environnements pour offrir à la fois une édition simple et intuitive à l'utilisateur novice et une édition puissante à l'utilisateur expérimenté.
- Les besoins couverts par le langage de sauvegarde : utilisés pour décrire les documents multimédias, et plus précisément pour spécifier et sauvegarder ces documents aux formalismes d'édition.

I.3.2.1. Besoins des auteurs suscités par le processus d'édition

Lors de l'étude des différentes sections précédentes de ce chapitre, on a pu dégager ces cinq besoins qui sont distincts et liés entre eux :

- *Besoins des fonctionnalités du système auteur*[3].
- *Besoins à l'enchaînement des phases d'édition.*
- *Besoins pour la vie du document en dehors d'une session d'édition*[4].

A. Visualisation et présentation

- *Perception du document dans toute sa complexité*

L'auteur a besoin pendant le processus d'édition d'avoir une vue globale et élémentaire des informations relatives aux différentes entités constituant son document. Par exemple, l'auteur doit pouvoir, en fonction de ses besoins d'édition, visualiser les

[3] Visualisation, édition, aide.
[4] Version, stockage, diffusion.

informations spatiales et temporelles de son document. Mais le plus important est que cette visualisation soit sous forme graphique par exemple et non pas sous forme textuelle ou à travers une palette, ainsi, elle doit se faire selon différents modes de représentation.

Ces modes doivent permettre à l'auteur de se faire une représentation mentale du document tel qu'il sera présenté au lecteur. De plus, ces représentations doivent lui fournir une explication sur le placement des objets en visualisant par exemple les relations ou les attributs qui induisent ce placement. Par exemple la dimension spatiale doit être visualisée telle que le lecteur la verra lors de la présentation, cette vue vérifie partiellement le principe du WYSIWYG[5] du fait des nombreux périphériques de lecture possibles pour le document. Dans le cas de la dimension temporelle, celle-ci peut être visualisée à l'aide d'une représentation graphique dans laquelle on peut référencer un temps absolu.

- *Perception de l'espace de résultats*

C'est les différentes présentations en relations qui forment la présentation finale du document. Cependant, la flexibilité de ces relations et la spécification partielle des médias ne définit pas une seule présentation, mais un ensemble de présentations possibles. Donc le système auteur doit rendre accessible à l'auteur cet espace de résultats.

- *Présentation*

Pour des raisons d'aide à la compréhension du document, il est important que l'auteur puisse à chaque étape du processus de conception, visualiser une des présentations possibles du document. Cette étape est essentielle à l'auteur du document car contrairement aux documents classiques, la phase d'édition ne s'effectue pas complètement sur la forme finale du document. En effet, le système auteur doit permettre d'exécuter chaque sous présentation du document de manière unique pour avoir une vue décomposée de ces présentations. Ainsi, l'auteur ne peut visionner complètement l'aspect dynamique de la spécification de son document à tout moment.

B. Fonction d'édition

Comme les niveaux de capacité intellectuelle des auteurs sont différents, le système auteur doit permettre à l'auteur d'effectuer ces quelques tâches nécessaires :

- Éditer le document en fonction des capacités de l'auteur.
- Faciliter la tâche d'édition en utilisant différents moyens comme des supports d'aide ou une interface d'édition simple et graphique.
- Exprimer tout ce que le langage de restitution permet de spécifier.

[5] What You See Is What You Get

Remarque :

Ces fonctions peuvent être des fonctions simples comme l'insertion ou la suppression d'un objet dans le document, ou des fonctions plus évoluées comme la copie d'attributs de présentation ou de synchronisation.

C. Aide à l'auteur

Cette aide se réalise lors des modifications des attributs des objets ou la vérification du document pour d'éventuelles incohérences avec des commentaires sur leurs causes. Parmi ces aides à l'auteur :

- *Formatage* : C'est le calcul des placements spatiaux et temporels de chaque objet à l'aide d'informations données par l'auteur. Ainsi l'auteur n'aura pas à recalculer ces placements à chaque modification de son document, car c'est le système qui effectue cette fonction. Donc, Il est nécessaire de réaliser cette tâche pour afficher à l'écran chacun des objets du document.
- *Vérification de la cohérence :* Le système auteur réalise cette vérification lorsqu'il effectue l'opération d'édition, ou à la fin de l'édition, ainsi il s'assure de la cohérence du document. Ces incohérences sont multiples, comme par exemple des informations contradictoires ou une référence à un objet non défini.
- *Diagnostic :* C'est la détermination de la nature et des raisons des erreurs trouvées lors de la vérification de la cohérence que le système a réalisé.

D. Cycle d'édition

La définition d'un document ne peut pas se faire intégralement, car l'auteur le construit peu à peu en s'appuyant aux services de visualisation et de vérification proposés par l'environnement. C'est pour cette raison qu'on parle de processus d'édition cyclique. Dans ce processus, on peut citer deux points importants :

- *La rapidité de prise en compte des opérations d'édition :* C'est la rapidité du système à effectuer les opérations que demande l'utilisateur. Un temps de réponse qui n'est pas raisonnable peut entraîner l'utilisateur à renouveler son action, ou même pire, à s'ennuyer. Donc, il est nécessaire à ce que le délai de réponse dans un cycle d'édition soit aussi court que possible.
- *La localité des modifications :* Les modifications réalisées par le système sur les résultats pendant une opération d'édition, doivent être le plus centrées que possible, pour qu'il n'y ait pas de conflits lors de la présentation.

E. Cycle de vie du document

Après avoir spécifié les besoins de l'auteur lors de la définition de son document, on va maintenant voir les différents besoins d'un document après son écriture, car le cycle de vie d'un document ne se termine pas à la fin de son écriture, mais il faut pouvoir modifier, remplacer, réutiliser ou diffuser une partie ou bien la totalité du document. Ces différents besoins sont les suivants :

- *Diffusion* : Parmi les besoins d'un document, sa diffusion. Il est très important à ce qu'il soit considérablement diffusé ou bien sur le Web, ou par CD-ROM.
- *Échange avec d'autres personnes ou outils* : Lors de l'opération d'édition d'un document multimédia, l'échange du document avec d'autres personnes ou d'autres outils peu s'avérer bénéfique.
- *Réédition* : Le système auteur doit permettre dans un premier temps à l'auteur de pouvoir rééditer son document, même s'il a déjà été diffusé lors d'une présentation. Et dans un second temps, il doit permettre de sauvegarder sous une forme standard le document multimédia.
- *Réutilisation de parties de document* : Le système permet à l'auteur de séparer certaines parties du document et de les réutiliser à nouveau pendant la conception d'un document multimédia, car souvent on en a besoin. Mais n'empêche que cette opération n'est pas aussi commode que l'on croit à cause de la synchronisation ou des liens externes des parties qu'on veut utiliser.

I.3.2.2. Besoins de l'auteur couverts par les langages [TAR 97]

Lors de l'étude des langages de programmation, et d'après des critères et des propriétés qui aident à l'édition des documents multimédias, ou bien à la numérisation d'un scénario dans un format adéquat, et dans une simplicité maximum. On remarque que cet ensemble de propriétés peut être divisé en deux catégories distinctes :

- ***Expressivité***, signifie ce que le langage permet de définir comme comportements ou propriétés sur les objets de base d'un document.
- ***Capacité de spécification***, signifie la simplicité avec laquelle l'auteur peut définir ces comportements à l'aide du langage.

I.3.2.2.1. Expressivité du langage

C'est le point de départ, car c'est l'étape choix du langage de programmation du document multimédia. Pour cela il faut nécessairement être capable d'identifier des classes de documents, ou plus exactement tout les éléments ou attributs qui forment la grammaire de ce langage, dont leurs manipulations selon des règles strictes donne création d'un document multimédia. Ainsi, on arrive à connaître l'aptitude du langage multimédia en analysant les informations et présentations résultantes, ou nos propres besoins qu'on désire réaliser. Et par la suite, connaître l'aptitude du langage de programmation à couvrir partiellement ou entièrement ces besoins ou les résoudre.

Pour mieux éclaircir ce travail on va présenter deux types de besoins qui nous permettent dans la suite d'identifier trois modèles de documents :

1) Spécification des médias

Elle permet de recouvrir les besoins de l'auteur en termes de définition des composants élémentaires de son document, parmi ces composants indispensables :

- *Paramétrage des attributs associés aux médias :*

  Il est important que l'auteur puisse paramétrer les informations de visualisation de ces médias que ce soit pour les attributs spatiaux (haut, bas, largeur, hauteur), la couleur du fond, et sans négliger l'aspect temporel (début, fin, durée) ou de synchronisation.

- *Médias continus / discrets :*

  L'auteur a besoin de spécifier dans son document à la fois des médias discrets (images, textes) et des médias continus (sons, vidéo).

  Il existe aussi un autre critère qui se base sur la perception, et là aussi, on peut dégager deux catégories : [TEC 98]

  1. *Médias visibles* **:** C'est les médias qui ont la possibilité d'être affichés, comme une image ou une vidéo.
  2. *Médias audibles* : Comme les sons.

- *Médias déterministes/indéterministes :*

  Lorsque l'auteur inclut un média continu dans un document, il doit pouvoir spécifier le comportement qu'il désire. On dira qu'un document multimédia est non seulement caractérisé par des contenus de natures diverses (Statique ou dynamique), mais il est aussi définit par un scénario décrivant l'enchaînement de ses composants. On peut énumérer deux types de scénarios :

1. Scénarios déterministes : Ils décrivent des présentations où la durée de chaque élément est connue. Le système procède aux calculs avant l'appel au lecteur, et le lecteur n'interfère en aucun cas pour définir le début et la fin d'un élément.
2. Scénarios indéterministes : Ils décrivent des présentations où certains éléments ont, soit une durée inconnue, ou que leur durée soit modifiable au moment de leur présentation.

- *Plugins / Applets :*

  L'auteur peut importer dans son document des modules externes qui lui permettent d'afficher ou de manipuler des données structurées ayant déjà leur propre comportement, comme par exemple inclure une page en HTML dans le document.

2) *Les comportements*

Ces comportements permettent à l'auteur du document de modéliser l'enchaînement temporel et spatial, ainsi que toutes les actions autorisées au lecteur, parmi ces comportements on cite :

2.1) *Comportement local*

Définit l'enchaînement temporel et spatial entre les médias ou les groupes des médias comme un groupe « par » ou « seq ».

Exemple :

```
...
<text src="titre.rt" region="3" />
<seq>
  <img src="rose.jpg" region="1" />
  <img src="rose01.jpg" region="1" />
</seq>
...
```

Figure I-2 : Exemple d'utilisation d'un groupe média « seq ».

2.2) *Interactions locales*

Ces interactions désignent que les médias locaux du document sont définis lors de sa spécification. Elles autorisent au lecteur de déclencher des actions sur le document comme le lancement ou l'arrêt d'un objet. Ces interactions modifient le cours de l'exécution en incluant, ou en supprimant des médias dans le cours de la présentation.

2.3) *Navigation*

Comme les interactions locales, la navigation est aussi définie lors de la spécification ; elle permet au lecteur d'explorer le document. Cette navigation peut être structurelle, spatiale ou temporelle. Les actions propres à la navigation sont synchrones au

comportement temporel du document, et les interactions de navigation n'ont pas d'influence qui change le scénario du document, puisqu'elles ne changeront que les instants de présentation, donc l'aspect temporel du média visé.

Exemple :

Dans cet exemple, nous faisons référence d'un nouveau lien démontré par la deuxième ligne de l'exemple, qui représente une page HTML de site Web.

```
...
<a href="http://www.archive.com/test1.smi"/>
      <video src="rtsp://foot.com" region="fenêtre2"/>
</a>
...
```

Figure I-3 : Exemple de navigation.

2.4) Acquisition d'information

Désigne tous les moyens qui permettent l'édition d'informations dans tous ces types, de la part du lecteur par exemple, bouton de sélections, champ de saisie texte…

La manipulation de ces différents modèles (comportement local, interactions locales, navigation, acquisition d'information) permet à l'auteur d'écrire des documents multimédias très riches et variés. Ainsi, on peut les classer en trois grandes classes selon le type de comportements qu'elles admettent :

1. *Document entièrement prédictif* :

Dans ce type de document, l'auteur n'est pas maître de la présentation, par contre il sait tout à fait ce qui va être joué, car le document garde son scénario original, ce qui signifie qu'il n'y a aucune interaction utilisateur. Donc le système de présentation peut calculer les instants « début » et « fin » des objets statiquement, mais seulement si le document contient des objets indéterministes. Ces objets de type médias continus seront coupés ou affichés à l'écran en fonction de la durée calculée ou prévue initialement.

2. *Document prédictif avec navigation :*

Concernant le document prédictif avec navigation, on garde la même définition précédente, en ajoutant le fait que ces documents sont composés d'un ensemble de sous documents prédictifs pour que le lecteur navigue dans la présentation en cours. On a deux types de navigation, comme suit :

- *Navigation temporelle* **:** Par son nom on comprend qu'il s'agit de navigation dans l'espace temporel où le lecteur peut aller à un instant précis de la présentation.
- *Navigation hypertexte* : Ce type de navigation déclanche une sous présentation dans une présentation en cours, dans laquelle le lecteur crée un enchaînement entre ces documents.

Remarque :

On utilise généralement ces types de présentations dans des diaporamas ou présentations hypertexte classique.

3. *Les documents interactifs :*

Contrairement aux documents prédictifs, un document interactif interpose le lecteur, car il change de scénario si on peut l'expliquer ainsi, et ça selon les interactions locales ou par les médias introduites. Puisque les interactions du lecteur peuvent modifier localement le document et remettre en cause les durées calculées initialement par exemple. Ou encore, les objets indéterministes qui ne sont plus forcément interrompus, de cause qu'on ne connaît pas statiquement la durée de présentation, on peut les laisser s'exécuter jusqu'à la fin. Le système devra être en mesure de réévaluer la suite du document pour garder les synchronisations entre les objets. Cela impose une certaine réactivité du système de présentation.

Exemple :

Dans cet exemple, nous mettons en parallèle un diaporama en boucle avec une fin ignorée, et un média audio de type indéterministe. La présentation prend fin quand le clip audio s'achève et donne fin au diaporama en boucle.

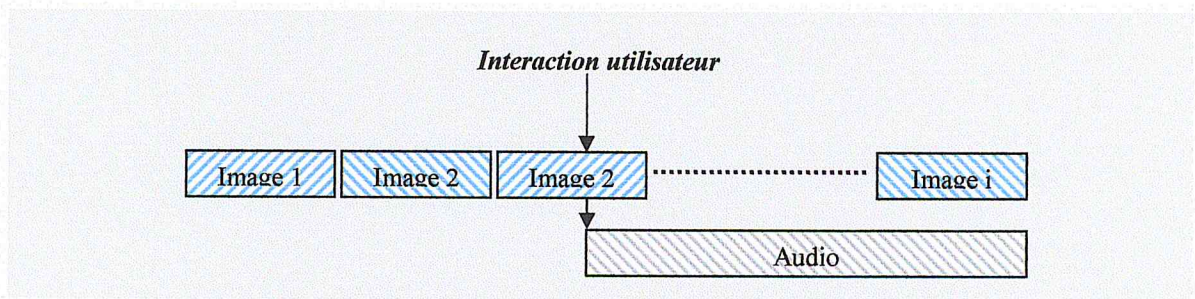


Figure I-4 : Exemple de document interactif.

I.3.2.2.2. Capacité de spécification

Dans cette partie du chapitre, on va dégager tous les besoins de l'auteur qui vont l'aider pour une meilleure spécification de son document multimédia, qu'on peut différencier selon quatre catégories :

1) ***Le moyen de structuration du document***

Pour une bonne organisation du document, l'auteur doit opter pour le découpage de celui-ci, et cela pour simplifier la phase d'édition. Ainsi la structuration permet de définir des groupes dans un document et d'associer des propriétés communes aux objets du groupe. De cette façon, on peut diviser la tâches d'édition en sous tâches plus simples, pour avoir la possibilité d'utiliser certains groupes dans d'autres parties du même document, ou aussi dans d'autres documents. On peut rapprocher cette notion à celle utilisée dans les langages de programmation (java, C++, C,...) par exemple les classes, modules ou même les structures utilisée.

On peut trouver dans un document plusieurs structures qui peuvent être spatiales, hypertextes, logiques et temporelles.

2) Support pour la modification du document

Pour que l'auteur puisse effectuer facilement d'éventuels changements dans une présentation ou dans le contenu d'un document existant.

La modification d'un document dépend de modifier soit :

- les attributs d'un objet
- la structure du document

Remarque :

Les modifications apportées doivent pouvoir se réaliser comme une mise à jours sur le document sans remettre totalement en cause son contenu et le travail déjà effectué.

3) Rapidité de spécification de documents

Pendant l'édition, l'auteur désire exprimer certains comportements spatiaux et temporels de son document avec une difficulté proportionnelle à la complexité du comportement voulu, donc l'auteur doit pouvoir exprimer des comportements complexes sans avoir à exprimer tous les comportements intermédiaires en détail pour la réalisation.

Ainsi on va maintenant présenter quelques instructions indispensables et principales dans la spécification d'un document multimédia par l'auteur :

3.1. Relations entre les objets

Elles permettent à l'auteur de placer des relations prédéfinies entre les objets, lui évitant ainsi d'avoir à spécifier l'ensemble des comportements élémentaires qui sont communs à tous les documents. C'est par exemple le cas des relations temporelles *avant* et *après* ou des relations spatiales *centré* ou *aligné*.

3.2. Événements

Ces événements donnent la possibilité de spécifier des comportements plus complexes, en permettant à l'auteur d'associer des actions à un événement dynamique, sachant qu'au moment de l'écriture du document l'instant d'occurrence n'est pas connu. La notion d'*événement* va définir par la suite l'événement et l'action qui lui sont associés.

3.3. Instructions conditionnelles :

Par nécessité du besoin de présentation, l'auteur se sent forcé de faire des tests sur la valeur de variables, ou d'attributs, et même sur l'état de certains objets (actif, pause, affiché, masqué,…) ou bien sur l'état du système (système de présentation du document multimédia, le système de la machine).

3.4. Boucles finies ou infinies :

Dans la présentation d'un média ou d'un groupe de médias, la définition des boucles peut s'avérer essentielle.

Exemple :

Dans cet exemple, nous spécifions une musique de fond qui se jouera en boucle pendant toute la présentation du document.

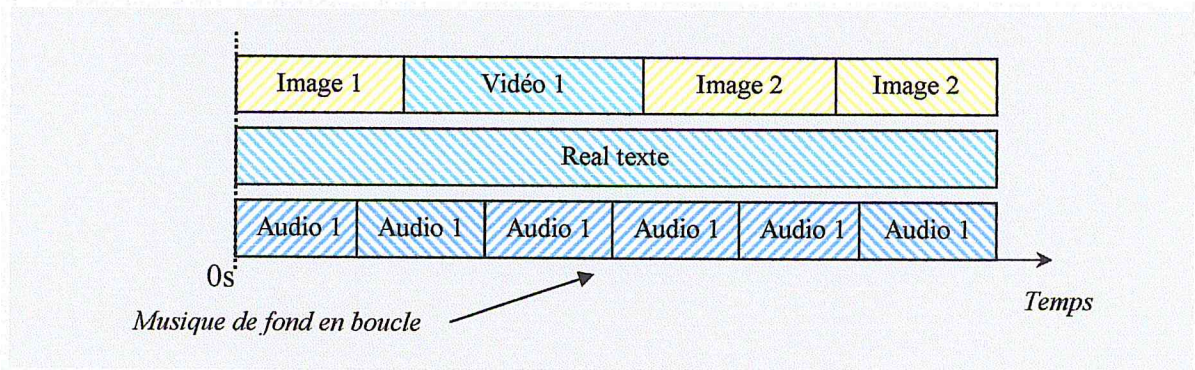


Figure I-5 : Exemple de boucle dans une présentation.

4) Support pour la spécification de documents adaptables

Ce support aide l'auteur à définir un ensemble de comportements qui dépendent des conditions de présentation de son document. On trouve dans cette catégorie de besoins l'écriture des documents qui s'adaptent seuls au contexte de présentation. Or le contexte de présentation n'est en faite que les conditions liées au système ou à l'utilisateur.

4.1. Le système :

- *Le périphérique de restitution* :

Ce paramètre est utilisé pour un affichage du document multimédia favorable quelque soit l'aspect externe. Autrement dit, le document doit être présenté de façon idéale.

- *Le débit réseau :*

Dans ce paramètre, l'auteur doit faire face à l'évolution du débit du réseau lors d'une présentation, en définissant les réactions de son document. On peut citer comme exemple le changement des médias utilisés dans le même document.

- *La charge de la machine :*

Lors d'une présentation, la charge de la machine peut évoluer, ce qui entraîne le changement de la mémoire disponible.

4.2. L'utilisateur :

- *La langue :*

L'auteur peut spécifier des médias pour des langues choisies et cela pour pouvoir créer un document dans plusieurs langues de présentation.

- *Le niveau d'expertise :*

Ce paramètre permet à l'auteur de spécifier le comportement de son document selon le niveau de l'utilisateur.

- *Les déficiences :*

Une lecture favorable du document implique son ajustement pour toutes les déficiences possibles du lecteur. Par exemple, pour un malentendant, le système doit pouvoir utiliser des effets sonores de volume élevé, ou bien de remplacer les médias sonores par des médias textuels.

I.4. Les standards de documents multimédias [TAR 97]

Le processus de standardisation joue un rôle essentiel dans la progression du multimédia. Les standards permettent de stabiliser les formats, et ses interfaces applicatives (API). Un standard de présentation multimédia doit incorporer les différents médias ainsi qu'une présentation permettant de les composer.

À cette partie du chapitre, on va s'intéresser aux langages de sauvegarde, qu'on illustrera à travers d'environnements auteur réels utilisant les formalismes du langage de sauvegarde comme formalisme d'édition.

On distingue plusieurs langages qui décrivent des documents multimédias, néanmoins ils peuvent être classés dans l'un de ces langages de sauvegarde, ou dans une combinaison des quatre catégories suivantes :

- ***L'approche absolue*** *:* Cette approche permet à l'auteur de décrire le positionnement des objets dans le temps et l'espace explicitement.
- ***L'approche programmation :*** L'auteur spécifie dans cette catégorie, à l'aide d'un langage de programmation de type impératif, le comportement des objets médias que comporte son document.
- ***L'approche événementielle :*** Elle permet à l'auteur de définir le comportement de ses objets médias en s'appuyant à des règles précises, comme des événements déclencheurs ou des conditions et des actions.
- ***L'approche relationnelle :*** Dans cette approche, l'auteur décrit des relations élémentaires entre les objets, comme par exemple (avant, après, au-dessous, au-dessus, durée, après, ...).

Ainsi, concernant les quatre catégories d'approche définies brièvement précédemment, on va s'approfondir dans cette analyse, et illustrer chaque approche en lui recommandant un ou plusieurs langages représentatifs, et on dégagera d'une part la façon dont les langages de description de documents multimédias répondent aux besoins auteurs liés au langage, et d'autre part comment un environnement d'édition répond aux besoins de l'auteur couverts par l'environnement d'édition.

On classe ces approches comme suit :

- *En premier*, on présentera l'approche absolue utilisée particulièrement dans Director qui est un des outils les plus utilisés pour concevoir des documents multimédias.
- *Deuxièmement*, on présentera les approches à base de langages de programmation en les illustrant au travers du langage Java le plus connu et utilisé par les auteurs.

- *Troisièmement*, on présentera un langage de description à base d'événements (MHEG [MHE 93]).
- *En dernier*, on présentera deux approches relationnelles :
  - SMIL, qui est une recommandation du W3C [W3C 98] pour spécifier le comportement temporel des documents.
  - Madeus qui est un langage à base de contraintes développé au sein du projet OPERA [LAY 97].

I.4.1. L'approche absolue [TAR 97], [DIR 00]

La particularité principale de cette catégorie est que l'auteur spécifie de manière absolue le placement spatial et temporel des objets. Cela se traduit par le fait que les instants de début des objets sont placés par rapport au début du document, et que la position spatiale des objets est donnée par rapport à un point de référence de l'écran par défaut en haut à gauche. Cette approche est utilisée particulièrement dans Director[6]

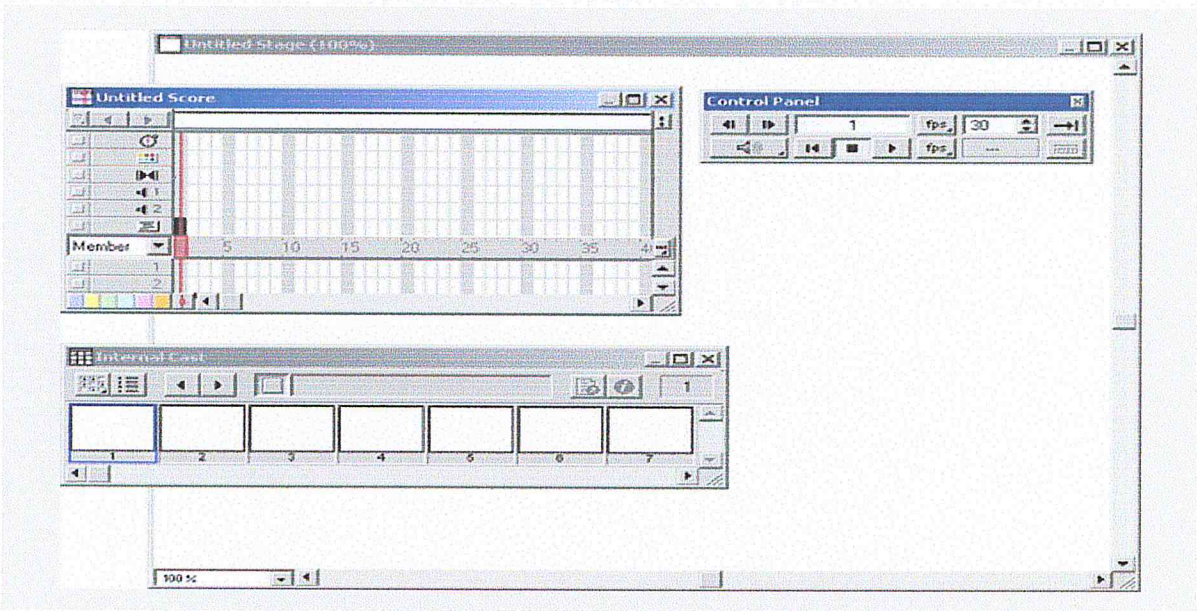


Figure I-6 : Interface de Director.

Exemple :

Dans la (*Figure I-7*) on a un exemple décrit de façon absolue. Ce document est composé de trois éléments textes qui s'affichent successivement dans le temps toutes les 10 secondes. Les trois éléments disparaissent ensemble de l'écran à la trentième seconde du document. Spatialement, les objets sont affichés les uns en dessous des autres sur l'axe Y, et les uns à côté des autres sur l'axe X.

[6] Director est l'un des premiers outils dédiés à l'édition de documents multimédias et ça depuis son apparition à la fin des années 80.

```
...
Temporel :
      Objet Titre1
            Début = 0
            Durée= 30
      Objet Titre2
            Début = 10
            Durée= 20
      Objet Titre3
            Début = 20
            Durée= 10
Spatial :
      Objet Titre1
            X = 120
            Y = 60
            Texte = "Institut"
      Objet Titre2
            X = 180
            Y=80
            Texte = "d'Informatique"
      Objet Titre3
            X = 240
            Y=100
            Texte = "de Blida"
...
```

Figure I-7 : Description absolue du placement temporel et spatial.

I.4.2. L'approche programmation [TAR 97], [MAC-LIN 00]

Cette approche est particulièrement essentielle pour la conception de documents multimédias car elle utilise des langages de programmation. Donc, l'auteur doit impérativement avoir le contrôle sur le langage qu'il va utiliser. On distingue dans les langages de programmation deux catégories :

- La première catégorie consiste à utiliser des langages de script (langage interprété). Ces langages, ont la possibilité de s'intégrer ou non dans des environnements auteurs. On peut citer comme exemple **Lingo** qui est intégré dans l'environnement Director ou plus encore JavaScript qui admet des changements sur les images ainsi qu'il effectue des animations.
- La deuxième catégorie utilise des langages généraux tel que C++, Java, ... Grâce aux nouvelles versions de ces langages qui sont mises au point spécialement pour la manipulation de différents médias donc la création de documents multimédias, et même la réduction du temps et le coût de conception. On peut citer comme exemple la bibliothèque : (Java Media Framework, [JMF 00]), qui est une version de Java et elle est portable sur différentes plates-formes.

Remarque :

Les langages de programmation eux-mêmes sont divisés en deux catégories : les langages compilés par exemple C, C++, Pascal, et les langages interprétés comme Prolog, Java :

- Les langages compilés nécessitent une phase de compilation du programme pour toutes les plates-formes si l'on désire rendre les documents accessibles aux lecteurs. L'avantage de cette approche réside dans le fait que le document produit est adapté pour une présentation sur la plate-forme cible, donc unique.
- Les langages interprétés, peuvent être exécutés sur tous les systèmes d'exploitation qui possèdent une machine virtuelle de ce langage permettant ainsi au programmeur d'être déchargé de la gestion de la portabilité du code écrit. Parmi les langages interprétés Java qui offre la possibilité au programme de ce langage d'être intégré dans une page Web, le rendant ainsi accessible par le biais *d'applet.*

Définition :

Une *applet* permet à un programme Java d'être intégré dans une page Web, le rendant ainsi accessible.

Les langages interprétés sont les langages les plus utilisés dans ce genre d'activité, car ils ont la possibilité de restitution ainsi que différentes plates-formes de conception.

L'un des langages les plus utilisés dans la conception de documents multimédia est le langage Java. Développé au début des années 90, le langage Java a réussi à intégrer totalement la manipulation des protocoles réseau, de la sécurité, ainsi que des médias (JMF), ainsi pour ce dernier il nous offre un ensemble de bibliothèques évoluées, par exemple, pour manipuler les médias en deux ou trois dimensions il y a Java 2D et Java 3D.

Exemple :

Dans l'exemple qui suit (*Figure I-8*) on distinguera deux versions d'un même document multimédia appelé *Test* :

- La première partie (scène_absolue), utilise une spécification absolue, nécessitant ainsi une description plus longue et moins facile à modifier du document multimédia.
- La deuxième partie (scène_relative), utilise une spécification par relation, caractérisée par sa souplesse lors de l'édition. Notons que cette spécification nécessite l'écriture d'un formateur temporel et spatial pour calculer le placement réel des médias.

```
public class DocumentUSDB
void CreerMenu() {
...
}
void Scene1 absolue() {
// le constructeur de Media text prend 4 paramètres : le texte à
afficher,
// les coordonnées spatiales de l'objet, l'instant de début de l'objet,
// et l'instant de fin.
MediaText T1 = new MediaText("Université",120,60,0,50);
//Valeur, X, Y, Début, Fin
MediaText T2 = new MediaText("Saad",180,80,10,50);
MediaText T3 = new MediaText("Dahleb",240,100,20,50);
MediaText T4 = new MediaText("de Blida",300,120,30,50);
T1.Start(); T2.Start();T3.Start(); T4.Start();
}
void Scene relative() {
// le constructeur de Media text prend 4 paramètres : le texte à
afficher,
// les coordonnées spatiales de l'objet, l'instant de début de l'objet,
// et l'instant de fin.
MediaText T1 = new MediaText("Université",120,60,0,50);
//Valeur, X, Y, Début, Fin
MediaText T2 = new MediaText("Saad");
// les autres attributs ne sont pas spécifié
MediaText T3 = new MediaText("Dahleb");
MediaText T4 = new MediaText("de Blida");
DécaléSpatialement(T1,T2,20);
DécaléSpatialement(T2,T3,20);
DécaléSpatialement(T3,T4,20);
// Dans ce cas, le programmeur doit implémenter un formateur temporel
// et spatial pour calculer les attributs sur tous les médias.
T1.Start(); T2.Start();T3.Start(); T4.Start();
}
static public void main(String argV) {
CreerFenetreGraphique();
CreerMenu();
Scene relative(); // ou Scene absolue();
}
```

Figure I-8 : Spécification du document USDB en Java.

Remarque :

On spécifie un document multimédia à l'aide d'un programme, dont l'auteur doit avoir le contrôle absolu car il doit gérer explicitement la manipulation de l'environnement de présentation du document en plus du placement temporel et spatial de son document. Pour mieux éclaircir ce point, il doit donc suivre les étapes :

- Création d'une fenêtre dans laquelle le document va être joué.
- Facilité de la réutilisation et de la manipulation du document avec des aides dans le programme.

L'approche de programmation peut avoir des avantages, comme elle peut avoir des inconvénients. On peut évoquer comme avantage important la facilité de décrire n'importe quel document multimédia, puisqu'on dispose d'un outil de programmation puissant.
Et pour les inconvénients, on peut citer :

- Difficulté de spécification, l'auteur est obligé de programmer tous les comportements des objets médias du document.
- Accès limité à la programmation, c'est-à-dire qu'un non informaticien ne peut parvenir à utiliser cette approche.
- Absence d'affichage de l'ensemble du document, de sa dimension temporelle et de sa structure en particulier.

I.4.3. L'approche événementielle [TAR 97]

Dans ce type d'approche, on comprend par son appellation qu'il permet l'utilisation des règles de type événements, conditions, actions. De tel façon que ces règles utilisent une sémantique très simple, puisque si l'événement est déclenché et que les conditions sont vérifiées alors les actions sont effectuées. Ces événements peuvent être associés au début et fin d'un objet, au changement de comportement comme l'attribut couleur, longueur, largeur, hauteur ou bas, …, ou encore à une action déclanchée par l'utilisateur comme un clic sur un objet média par exemple.
Le MHEG est un langage de programmation connu par l'approche événementielle et même privilégié. Avec ce langage, la description du comportement spatial et temporel se fait au moyen :

- d'objets composites,
- de primitives de composition : Qui permettent de donner un comportement temporel simple à tous les objets d'un composite,
- de liens : Les liens ne sont pas de simples liens hypertextes, mais permettent de spécifier des événements temporels et spatiaux et le lien est constitué de deux parties :
  - une partie condition (*LinkCondition*)
  - une partie action (*LinkEffect*) qui correspond à une liste d'actions à effectuer sur une liste d'objets.

```
<?xml version="1.0"?><!DOCTYPE MHML PUBLIC "mhml.dtd" "mhml.dtd" >
<MHML>
<BODY>
<DIV y="0" x="0" width="640" height="480" id="Document">
<TXT y="0" x="0" width="100" height="100" id="USDB"/> USDB </TXT>
      <DIV y="0" x="0" width="640" height="480" id="Groupe">
            <TXT y="180" x="80" id="USDB0"/>  Université </TXT>
            <TXT y="240" x="100" id="USDB1"/> Saad       </TXT>
            <TXT y="300" x="120" id="USDB2"/> Dahleb     </TXT>
            <TXT y="360" x="140" id="USDB3"/> de Blida   </TXT>
      </DIV>
</DIV>
</BODY>
<BEHAVIORS>
<START LINK>
      <ACTION>
            <RUN propagation="true" target="Document"/>
      </ACTION>
</START LINK>
<LINK div="Document">
      <RUN EVENT target="Document" value="true"/>
      <ACTION>
            <RUN propagation="true" target="USDB0"/>
      </ACTION>
</LINK>
<-- Evénement exemple -->
<-- Définition d'un événement sur le composite groupe ; Cet événement est
un événement temporel qui sera déclenché au début du document (temps
égale à 0). L'action associée à cet événement sera une action de
présentation (run), qui aura pour effet de démarrer l'objet USDB0. La
valeur propagation = true, signifie que si l'objet USDB0 est un objet
composite, cet événement sera automatiquement propagé à ses fils. -->
<LINK div="Groupe">
<TIME EVENT target="USDB0" operator="equal" value="0" />
      <ACTION>
            <RUN propagation="true" target="USDB1"/>
      </ACTION>
</LINK>
<-- fin de l'événement exemple -->
<LINK div="Groupe">
      <TIME EVENT target="USDB0" value="10" operator="equal"/>
      <ACTION>
      <RUN propagation="true" target="USDB2"/>
      </ACTION>
      </LINK>
<LINK div="Groupe">
      <TIME EVENT target="USDB0" value="20" operator="equal"/>
      <ACTION>
      <RUN propagation="true" target="USDB3"/>
      </ACTION>
      </LINK>
<LINK div="Document">
      <TIME EVENT target="USDB" value="40" operator="equal"/>
      <ACTION>
      <STOP propagation="true" target="USDB"/>
      <STOP propagation="true" target="USDB0"/>
      <STOP propagation="true" target="USDB1"/>
      <STOP propagation="true" target="USDB2"/>
      <STOP propagation="true" target="USDB3"/>
      </ACTION>
      </LINK>
</BEHAVIORS>
</MHML>
```

Figure I-9 : Spécification événementielle du document USDB en MHML.

I.4.3.2. Synthèse sur l'approche événementielle

A part leurs pouvoirs d'expression très important et qu'ils permettent de décrire tous les types de documents, les langages événementiels ou cette approche événementielle trouve la difficulté majeure dans la maîtrise du comportement du document. En effet, avoir une vue complète du comportement est relativement difficile à cause de l'aspect non prédictif de certains événements et de par le grand nombre d'événements qu'il faut spécifier pour décrire le comportement de ce document.

De nos jours peu de systèmes auteur permettent d'avoir une édition évoluée de tels langages, on se ramène souvent à la programmation classique comme celle de Lingo de Director.

I.4.4. L'approche relationnelle [LAY 97], [LAY 99]

L'arrivée de cette approche apparaît pour faciliter l'écriture des documents multimédias en donnant à l'auteur l'éventualité de spécifier des relations entre les objets, évoquons comme :

- L'enchaînement temporel de type séquence de telle façon que les objets sont joués les uns après les autres.
- Le positionnement spatial défini par des placements relatifs de type centrage, alignement, décalage.

L'auteur ne spécifie donc pas de manière absolue la position spatiale et temporelle de tous les objets, c'est le système qui, à partir de toutes les informations qu'il possède, calcule ce positionnement, cette étape appelée le formatage, précédemment ; et ces relations introduites sont de nature flexibles ou non flexible.

Définition :

Une relation flexible permet de définir un ensemble de placements possibles, ces placements peuvent être des valeurs d'attribut de positionnement spatial ou des valeurs de temps. Comme dans notre cas d'étude du code SMIL[9], on peut prendre comme exemple d'une relation flexible *begin* illustrée dans la (*Figure I-10*) qui spécifie qu'un objet B commencera à l'instant début de l'objet A, sans pour autant spécifier explicitement où se situent ces deux objets l'un par rapport à l'autre, et même la valeur début de l'objet B.

[9] (Synchronized Multimedia Integration Language) est un document XML modélisé comme des présentations multimédia, il permet d'intégrer et synchroniser un ensemble d'objets médias dans une présentation.

Pour rendre plus évident et plus clair cette approche, on présente comme exemple MHML qui est une version de MHEG dont la syntaxe a été mise sous forme XML[7].

I.4.3.1. Le langage MHML [MHM 98]

Le langage MHML [MHM 98] est un langage événementiel qui utilise une syntaxe XML [XML 99], ainsi un document MHML se compose de trois parties :

- Une partie *Area*, qui permet de définir des zones réactives pour les médias.
- Une partie *Body*, qui permet de définir des groupes d'objets dans le document. Sur chaque objet élémentaire du document, l'auteur peut définir un ensemble d'attributs permettant par exemple de décrire le placement spatial de l'objet et le style.
- Une partie *Behavior*[8], qui permet de définir le comportement du document.

Ce comportement est composé d'un ensemble de comportements élémentaires. Un comportement élémentaire est défini par :

- *Un événement* : Déclenché par le système ou par un objet du document. Un événement peut être associé à la position d'un objet, à son état d'activité, et enfin à la sélection d'un objet dans le document par le lecteur.

- *Une condition* : Permet d'évaluer un certain nombre de paramètres sur le système ou sur des objets du document. Les conditions couvrent le même éventail de valeur que les événements, c'est-à-dire que l'utilisateur a la possibilité de tester la position d'un objet, s'il est sélectionné ou pas, s'il est en train d'être exécuté ou non. Il peut aussi combiner les conditions via les opérateurs and et or.

- *Une liste d'actions* : Cette liste d'action sera à réaliser si l'événement est déclenché et si les conditions d'application des actions sont vérifiées. Dans la liste des actions que l'utilisateur peut décrire, on trouve le déclenchement et l'arrêt d'un objet ainsi que le changement d'attributs spatiaux.

Exemple :

Dans l'exemple suivant (*Figure I-9*) nous présentons en MHML le même exemple de document que celui de la (*Figure I-8*).

[7] (Extensible Markup Language) est un métalangage de spécifications des langages réguliers de marquage, il permet de décrire les informations caractérisant une classe de documents.

[8] La partie Behavior comporte un événement particulier, le Start_Link qui permet de définir les objets qui sont lancés au début de la présentation.

Exemple :

```
…
<video src="" id="A" dur="6s" begin="3s">
<img src="" id="B" begin="begin(A)">
<img src="" id="C" dur="3s" begin="end(B)">
…
```

Figure I-10 : Exemple de relation flexible.

Or les relations non flexibles permettent de définir des placements relatifs fixes, comme le démontre l'exemple dans la (*Figure I-11*).

Exemple :

```
…
<video src="" id="A" dur="6s" begin="3s">
<img src="" id="B" begin="4s">
<img src="" id="C" dur="3s" begin="8s">
…
```

Figure I-11 : Exemple de relation non flexible.

Le fait d'introduire des relations flexibles permet de modéliser des documents plus facilement adaptables.

SMIL [SMI 98], CMIFed [HAR 93], ISIS [KIM 95], TIEMPO [WIR 95], sont des langages qui illustrent l'approche relationnelle, et permettent plus aisément à l'auteur de modifier le document multimédia, puisque c'est le système qui s'occupe du calcul de la nouvelle valeur des attributs de tous les objets, et ça lors de l'ajout ou le retrait d'un objet.

Remarque :

On distingue dans cette approche deux catégories comme suit :

- La première catégorie basée sur des arbres d'opérateurs permet à l'utilisateur de définir par exemple un arbre temporel où les noeuds représentent des relations et les feuilles des médias ; utilisée dans SMIL et CMIF.
- La deuxième catégorie d'approche, permet à l'auteur de générer des groupes d'objets ainsi que de définir des relations binaires entre les objets d'un même groupe, cette méthode est utilisée dans ISIS, TIEMPO, MADEUS [LAY 97].

I.4.4.1. Une approche à base arbre d'opérateur [SMIL] [COU 98]

Le langage SMIL [SMI 98] est une recommandation du World Wide Web Consortium (W3C), sa syntaxe est décrite par une DTD XML [XML 98], on discerne deux versions de SMIL, la première SMIL1.0 [SMI 98] et la deuxième SMIL2.0 [SMI-20], qui vont aider à mieux éclaircir cette approche.

SMIL donne l'avantage d'une spécification relative et facilement modifiable. De telle sorte si on veut insérer un nouvel objet, on a juste à le spécifier avec l'instant de début désiré. Et on n'a pas alors à modifier la durée des objets, puisque celle-ci étant définie de façon relative auparavant.

L'approche à base d'arbres d'opérateurs n'est pas formellement une approche relationnelle pure, puisqu'un objet ne peut être impliqué que dans une seule relation. De ce fait, lorsque l'on désire ajouter une relation, on est souvent amené à restructurer l'arbre d'opérateurs. D'une autre part on trouve une difficulté importante avec cette approche et c'est de bien concevoir son document pour permettre des modifications futures, donc l'auteur ne peur prévoir toutes les modifications futures qu'il voudra apporter à son document multimédia.

Pour toutes ces raisons les remises en cause de la structure de l'arbre d'un document SMIL sont fréquentes, et on sait que restructurer un arbre n'est pas une chose simple.

Néanmoins, un des avantages importants de cette approche est qu'elle permet de définir rapidement un comportement temporel simple de tous les médias ; En plus le langage SMIL permet de définir des documents intégrant des médias indéterministes tout en gardant certaines synchronisations comme *endsync*.

Exemple :

On peut voir dans cet exemple un fichier complet d'un document multimédia.

```
<smil>
  <head>
   <layout>
      <region id="région1"  top="240"  left="10"  width="800"
      height="600" />
      <region id="région2"  top="360"  left="50"  width="800"
      height="600" />
      <region id="région3"  top="350"  left="50"  width="800"
      height="600" />
      <region id="région4"  top="120"  left="10"  width="800"
      height="600" />
   </layout>
  </head>
  <body>
      <par dur="40">
      <audio src="Musique.mp3" >
      <text src="Texte1.rt"  id="V1"  region="region1"/>
      <text src="Texte2.rt"  id="V2"  begin="10s"  region="region2"/>
      <text src="Texte3.rt"  id="V3"  begin="20s"  region="region3"/>
      <text src="Texte4.rt"  id="V4"  begin="30s"  region="region4"/>
      </par>
  </body>
</smil>
```


Figure I-12 : Exemple complet d'un document multimédia SMIL.

I.4.4.2. Une approche à base de relations binaires [Madeus] [LAY 97]

Le système d'édition et de présentation Madeus met en oeuvre, dans un environnement interactif, un modèle temporel de documents multimédia, de ce fait le langage Madeus [LAY 97] a été développé au sein du projet Opéra. On peut considérer que le processus d'édition d'un document est une combinaison des quatre fonctions suivantes :

- ***Saisie et structuration*** **:** Cette fonction permet à insérer de nouveaux objets multimédia dans le document et à organiser son contenu, ces médias seront référencés dans les sections spatiales et temporelles.
- ***Composition temporelle*** **:** Elle permet de définir les relations temporelles entre les objets d'un document multimédia, ou entre groupes d'objets.
- ***Placement spatial*** **:** Cette opération consiste à définir l'apparence graphique du document à l'écran et le positionnement géométrique entre ses objets en termes de relations spatiales.
- ***Structure de navigation*** **:** Cette opération consiste à superposer à la structure logique d'un document des parcours de navigation qui traversent son organisation hiérarchique comme des sauts intra et inter documents, ainsi que de son scénario temporel comme des sauts d'intervalle de temps.

Exemple :

Dans cet exemple (*Figure I-13*) nous reprenons la présentation précédente de l'USDB, dans un format du langage Madeus, or contrairement à SMIL, on remarque que notre document Madeus est réparti en trois parties visibles :

- La description des médias,
- La description de la structure temporelle,
- La description de la structure spatiale,

Dans les deux dernières structures, on utilise des relations comme *Finishes*, et *AlignéAGauche* pour définir le positionnement des objets, soit pour une structure temporelle ou spatiale. Or la grande différence avec la structure du langage SMIL est que l'on a des groupes d'objets et non pas un arbre d'opérateurs.

En conséquence, la modification du placement temporel des objets, ne touchera pas la structure du document, car le changement ne va être effectué qu'au niveau des relations entre les objets ; mais par contre le seul inconvénient qui réside dans ce langage, c'est la

Le but de SMIL est de simplifier et faciliter l'intégration de médias de différents types, et même sans négliger le moyen de spécifier les dimensions spatiale et temporelle du document multimédia. On découvrira par la suite en détaille dans le chapitre II et III, les grandes lignes et caractéristiques de ce langage.

D'un point plus général, on peut exposer la principale hiérarchie de SMIL comme suit :

Un fichier SMIL se compose d'un élément *head* et d'un élément *body*.

- Dans la partie *head*, c'est un en-tête du fichier qui permet la spécification des informations d'affichage de notre document. Par exemple, pour définir le positionnement spatial des objets qui se fait au biais de régions.

Une *région* c'est une zone d'affichage à l'écran. Cette zone d'écran peut être partagée par plusieurs objets multimédias, c'est-à-dire qu'une région peut être la zone d'affichage en séquence de plusieurs médias discrets, comme des images, ou des vidéos.

- Avec le *body*, ou encore le corps du document, c'est la partie complémentaire du head puisqu'il contient la description de l'ensemble des objets manipulés dans le document multimédia, cette description peut être de type organisation temporelle des différent médias ou une définition des liens hypertexte contenus dans le document multimédia.

La partie *body* des documents SMIL est basée sur une structure hiérarchique comme suit :

- Les nœuds sont des opérateurs temporels qui décrivent la façon avec laquelle s'enchaînent temporellement les fils du noeud. Les opérateurs peuvent être soit l'opérateur *par* ou *seq* ; L'opérateur *par* permet de décrire une exécution en parallèle des fils, l'opérateur *seq* permet de décrire une exécution en séquence des fils.
- Les feuilles sont des médias.

Toutes les informations relatives à ces opérateurs temporels et à ces attributs de médias, l'auteur peut les compléter ou les modifier avec différents attributs comme :

- L'attribut *dur* permet de spécifier explicitement la durée d'un objet, or par défaut les objets discrets ont une durée infinie.
- Les attributs *begin* et *end* permettent, eux, de spécifier de façon absolue, ou relative, c'est-à-dire par rapport au début ou à la fin d'autres objets dans le même document, le début ou la fin des objets.

Remarque :

L'emplacement de ces attributs dans le corps du document permet de définir des synchronisations temporelles plus fines entre les objets.

localité des relations qui ne se fait que dans un groupe d'objet, et pas pour tout les objets du document.

Mais par contre cette localité est avantageuse pour l'auteur puisque il n'aura pas à revérifier toute la structure du document, mais rien que le groupe d'objets en cause.

Exemple :

```
<Madeus>
<Media>
     <text Id= "O1" value="Université" >
     <text Id= "O2" value="Saad" >
     <text Id= "O3" value=" Dahleb" >
     <text Id= "O4" value="de Blida" >
</Media>
<Spatial>
     <Groupe Id="Animation">
     <Zone ID="z1" media="O1" X="60" Y="120">
     <Zone ID="z2" media="O2">
     <Zone ID="z3" media="O3">
     <Zone ID="z4" media="O4" >
     <Relation>
     <AligneAGauche param1="z1" param2="z2" delta="+60">
     <AligneAGauche param1="z2" param2="z3" delta="+60">
     <AligneAGauche param1="z3" param2="z4" delta="+60">
     <Under param1="z1" param2="z2" delta="+20">
     <Under param1="z2" param2="z3" delta="+20">
     <Under param1="z3" param2="z4" delta="+20">
     </Relation>
     </Groupe>
</Spatial>
<Temporal>
     <Groupe Id="AnimationT" dur="40s">
     <Zone ID="T1" X="60" Y="120" media="O1">
     <Zone ID="T2" media="O2">
     <Zone ID="T3" media="O3">
     <Zone ID="T4" media="O4">
     <Relation>
     <finishes param1="z1" param2="z2" delay="10s">
     <finishes param1="z2" param2="z3" delay="10s">
     <finishes param1="z3" param2="z4" delay="10s">
     </Relation>
     </Groupe>
</Temporel>
</Madeus>
```

Figure I-13 : Spécification relationnelle du document USDB en Madeus.

Remarques :

- Madeus ne permet de décrire que des documents prédictifs avec des liens de navigation.
- L'insertion des objets indéterministes lève de nouveaux problèmes pour le formatage ([LAY 97]) non encore résolus de manière satisfaisante à ce jour.
- Un des avantages majeurs de l'approche est qu'elle permet à l'auteur de décrire des relations flexibles.
- Un autre avantage est la facilité de modification due en partie aux intervalles de durées définies sur les objets.

I.4.5. Synthèse sur les approches relationnelles [DIS 98]

Les approches relationnelles actuelles sont restreintes dans le pouvoir d'expression. Ces approches n'admettent pas la spécification des interactions complexes entre l'utilisateur et le document, et elles ne permettent pas aussi des enchaînements complexes tels que des répétitions conditionnelles. Toutefois, l'extension de ces langages à une expressivité plus grande ne pose pas de problème, puisqu'on reste toujours dans le cadre de documents non complexes.

Dans ce cas là, les langages relationnels et les langages tels que MHML ont le même pouvoir d'expression ; Contrairement aux approches événementielles, les relations de haut niveau sont présentes dans les approches relationnelles, ainsi elles permettent de définir des comportements entre les objets, ce qui les rendent avantageuses. Cet avantage a le pouvoir de faire une description du comportement plus brève et dense ; en plus de ça, la sémantique de la structuration du document est plus forte. A comparer entre l'approche de SMIL et celle de Madeus, il existe une différence au niveau de la structure; Ainsi, dans SMIL, la structuration est une structuration temporelle à base d'arbre d'opérateurs ; par contre, dans Madeus, la structure temporelle est indépendante des relations temporelles à base de groupes. La structure à base d'arbre d'opérateurs peu avantageuse pour la description de documents non complexes, cependant la structure de groupes permet une plus grande extensibilité pour les modifications et l'évolution du document. On peut citer un autre avantage de ces approches qui est la définition des objets dans des intervalles de valeurs ; ce qui facilite l'adaptation de la spécification, ainsi que l'automatisation du processus de formatage.

La nouveauté dans l'approche relationnelle n'a pas encore fait ses preuves, particulièrement pour les documents complexes. Toutefois, il semble que cette approche offre de plus grands potentiels, comme du bon compromis qu'elle offre entre facilité de spécification et pouvoir d'expression, du point de vue de l'édition.

I.4.6. Synthèse sur les langages de documents multimédias

Dans la section précédente, on a vu que l'auteur de documents multimédias dispose d'un ensemble de formalismes pour spécifier ses documents. Tout au long de cette analyse, on a eu l'opportunité de voir les différences qui existent entre ces approches en terme d'expressivité et d'utilisation :

- Les approches absolues : Ces approches ont la possibilité de donner une spécification de documents multimédias sans synchronisation temporelle relative.
- Les approches programmation et événementielle : Ces approches ont le pouvoir de faire une description des trois classes de documents décrites dans la *section 3.2.1.1.* (expressivité du langage) ; Elles sont très expressives, mais leur utilisation est difficile.
- Les approches relationnelles : Ces approches permettent de spécifier un choix large de documents de manière relativement précise et intuitive ; ainsi, elles proposent un meilleur compromis entre expressivité et simplicité. Il existe deux sous-classes pour ces approches qui sont :
  - *Les arbres d'opérateurs :* ils rendent plus complexes certaines opérations, au profit de la simplicité.
  - *Les relations dans des groupes :* Ces relations proposent une plus grande souplesse de modification.

I.5. Discussion

La présence des standards (formats de compression, langage de spécification, modèles de synchronisation, plates-formes d'exécution) permet de stabiliser et de caractériser les besoin que doit combler la technologie multimédia. Par contre l'émergence de standard semble loin de se stabiliser, ce qui a pour conséquence que les applications et les outils de construction doivent prendre en compte des extensions et des nouveaux critères des standards.

Les standards proposent des langages de spécifications qui permettent la définition de documents : structure, médias contenus, disposition spatiale et temporelle, caractéristiques du « décor ».

I.6. Les outils d'édition

Dans cette partie de notre chapitre, on a pour but d'identifier les fonctions générales offertes par les environnements d'édition, de façon à les confronter aux besoins des auteurs. Néanmoins, cette opération présente un certain nombre de limites du fait de la dépendance entre l'outil et le langage de sauvegarde utilisé.

Les outils d'édition de documents sont issus de plusieurs origines, les plus essentielles sont :

- *Les documents textuels :* C'est des documents qui contiennent tous les types de médias, sans qu'il y ait de synchronisation temporelle entre eux ; On peut citer comme exemple Thot ou Word.
- *Les documents hypertextes :* Avec l'apparition d'Internet, des outils accompagnent l'arrivée de standards comme DHTML [DHT 98], flash [MAC-FLA 00], c'est le cas par exemple de RealNetworks [REA 98] et d'outils comme SmilWizard [REA 99].
- *Les bases de données :* Ce sont des bases de données qui ont la possibilité d'avoir des outils plus complets, ainsi ces outils permettent de construire de nos jours des requêtes complexes et de les présenter sous une forme multimédia.
- *Les documents multimédias :* Avec l'arrivée de ce type de documents, plusieurs outils ont permis de les éditer, et qui ont changé avec les besoins des auteurs, comme Director et Toolbook qui offrent des outils complets.
- *Les langages de programmation :* Ces langages facilitent la manipulation des médias et la visualisation de boîte à outils complexes, on peut donner comme exemple le langage Java.

Grâce à ces origines on peut classer les environnements en trois catégories qui sont :

1. Les approches à base de schémas : C'est une catégorie qui regroupe les outils originaires des documents structurés qui simplifient la tâche de l'auteur ; Ainsi les auteurs ne remplissent que des documents prédéfinis.
2. Les approches à base de programmation : Dans cette catégorie, l'auteur spécifie un document comme une application à part entière.
3. Les approches classiques : Ces approches sont issues de l'édition des documents textuels ou multimédias, où l'auteur définit d'une extrémité à l'autre son document avec l'aide du système.

Conclusion

A travers ce chapitre, on vient de faire une étude sur les différents systèmes et langages multimédias.

Le but de cette étude préliminaire consiste d'une part, à servir de base pour la conception et la création de notre outil d'édition dans les chapitre qui vont suivre, et d'autre part, un aperçu très intéressant, vu qu'il montre certaines techniques utilisées pour la spécification de la dimension temporelle des documents multimédias. On a aussi vu que chaque outil possède sa propre manière de répondre aux besoins d'un auteur.

Pour notre part, on remarque qu'un outil est susceptible de répondre aux besoins d'un auteur, ou plus tôt nos besoins pour la création de notre éditeur, s'il intègre les paramètres suivants :

- Choix d'un formalisme d'édition efficace, c'est à dire, un bon format de description de scénario comme le langage SMIL.
- Hiérarchisation du document, afin d'ordonner le travail et d'avoir un meilleur aperçu.
- Une interface de représentation temporelle de type Time Line. Vu qu'un time line permet à l'auteur d'avoir un aperçu intuitif sur le comportement temporel de ses éléments.

Dans les chapitres suivants on entrera dans le vif du sujet en abordant le langage SMIL et les différents outils qui se basent sur ce langage.

Chapitre II

LA DESCRIPTION DU LANGAGE SMIL LA PREMIÈRE VERSION SMIL 1.0

Chapitre II

LA DESCRIPTION DU LANGAGE SMIL

LA PREMIÈRE VERSION

SMIL 1.0

Historique

SMIL [SMI 98] a été développé par le World Wide Web Consortium (W3C). Les premières ébauches remontent à novembre 1997, la publication officielle est du 15 juin 1998 par la recommandation technique du W3C.

Il existe divers moyens d'apporter du contenu dynamique, alliant son, texte et animations sur un site web, Flash par exemple. De son côté, dans le cadre d'études sur la généralisation du format XML [XML 99], le W3C a mis sur pied des groupes de travail chargés de définir un format de données, fondé sur XML, permettant de créer des animations multimédia sur Internet. Ces groupes de travail ont donc défini SMIL.

C'est un langage de balisage permettant de synchroniser des fichiers de nature différente pour en faire des objets multimédias aptes à être intégrés dans une page Web ; SMIL à prononcer "smile" (sourire en anglais) est l'acronyme de « Synchonised Multimedia Integration Langage », « Langage d'intégration synchronisée du multimédia ».

SMIL est une application XML normalisée par le W3C, le support privilégié du transport d'informations dans le Web est la " page HTML ", qu'elle soit statique ou dynamique, qu'elle ne possède que des données ou qu'elle encapsule également les moyens de les traiter par le poste destinataire, les informations ainsi véhiculées ont été appelées par un internaute et envoyées à celui-ci pour une consultation à sa guise.

Donc il ajoute à ce mécanisme, sans rien lui ôter, la possibilité supplémentaire de présenter des informations, à l'internaute qui les a demandé, selon un assemblage spatio-temporel préparé par leur auteur de telle sorte que celles-ci soient communiquées avec la plus grande intégrité d'interprétation ; Pour cela, il permet de communiquer un même contenu sémantique sur des média différents : animations, images, textes, sons, vidéos, et d'organiser leur apparition de façon spatiale à l'écran et temporelle dans l'enchaînement et la progression des informations à transmettre, par exemple un auteur peut :

- Décrire le comportement d'une présentation dans le temps ;
- Décrire la disposition d'une présentation à l'écran ;
- Associer des hyperliens à des objets média.

II.1. SMIL, pour faire quoi ?

Les caractéristiques de SMIL en font un outil privilégié pour :

1. la transmission d'événements en direct, par exemple :
   - conférences, congrès,
   - soutenances de thèses,
   - cours magistraux,
2. la transmission d'événements édités en différé et l'édition de contenus complexes :
   - extraits montés de conférences ou de soutenances de thèses,
   - démonstrations et expériences de laboratoire,
   - livres de connaissances multimédia,
   - montage de contenus de formation à partir de banque de données pédagogiques,
   - création de banques de données pédagogiques.

II.2. Apports de SMIL1.0

Ce langage permet à l'auteur :

- Une description des différentes régions (mini fenêtre) et leurs dispositions spatiales par rapport à la fenêtre principale.
- Une description du comportement temporel (time line) des différents éléments qui composent une présentation.
- La création de présentations multimédias interactives qui s'adaptent à la vitesse de connexion.
- L'association des liens hypertextes avec des objets multimédias.
- D'inclure dans une même présentation des objets locaux, des objets d'un serveur temps réel, aussi bien que des objets d'un serveur http.

II.3. Les avantages du langage SMIL1.0

- *La disponibilité du média* : Les médias sont référencés et non inclus, l'avantage de cette méthode est que les éléments médias sont disponibles pour plusieurs présentations.
- *Les hyperliens* : SMIL permet de gérer des liens hypertexte en fonction du temps. Ce qui offre plusieurs choix dans une présentation, par exemple on peut afficher des sous titres dans une langue ou transmettre une traduction synchrone dans une autre langue.
- *La synchronisation* : Qui est le point fort du SMIL. Les différents médias d'une présentation peuvent être synchronisés d'une façon discrète ou continue. Autrement dit, on décide quand et où sur l'écran les contenus sont présentés. SIML donne le contrôle de la mise en page et du cadre temporel de présentation multimédia. De cette façon, on peut harmoniser la lecture de fichier son et vidéo de plus on peut ajouter des éléments textuels à un emplacement quelconque du film.
- *Un langage déclaratif* : Quand l'intégration du contenu du document média est souvent la province des scriptes ou autre définition de programme, SMIL est développé pour demeurer entièrement déclaratif plutôt que procédural sur cela, la description de SMIL ne définie pas ''comment'' la présentation est implémentée, mais plutôt ''quels'' sont les besoins de l'auteur.

II.5.1. L'élément head

L'élément ***head*** contient des informations qui n'ont pas de rapport avec le comportement temporel de la présentation. ; Cette partie appelée l'en-tête contient des informations relatives à la description du contenu, telles que les méta-données (balises <meta>), les informations sur le "découpage" de la zone principale et la disposition des différents éléments. Chaque élément peut être identifié avec un identificateur unique (attribut id="nom_identificateur").

Attribut	Valeur	Fonction
id	Caractères	Fixe l'identifiant du **head**, et il est unique

Tableau II-2 : Attribut de head.

L'élément **head** accepte les éléments fils suivants :

- **layout** : il est unique dans la partie **head**.
- **meta** : élément qui donne des informations concernant le document smil et on trouve plusieurs fils **meta** dans le **head.**
- **switch :** élément de choix peut contenir plusieurs **layout**.

II.5.1.1. L'élément meta

L'élément ***meta*** sert à donner des informations à propos de la présentation SMIL, telles que l'auteur, le titre ou un résumé. Elles ont comme attributs possibles, ***name*** et ***content***.

Remarque :

L'élément meta est un élément vide car il n'a pas d'éléments fils.

Attribut	Valeur	Fonction
name	Caractères	Indique le nom de la propriété.
content	Caractères	Indique la valeur de la propriété.

Tableau II-3 : Attributs de meta.

Exemple :

```
...
<meta name="Réaliser par " content="Hakim et Naouel"/>
<meta name="Titre" content="Exemple sur l'élément meta"/>
<meta name="Institut" content="Informatique"/>
...
```

Figure II-2 : Exemple d'élément meta.

II.5.1.2. L'élément layout

L'élément ***layout*** détermine la manière dont les éléments, dans le corps du document, se positionnent sur une surface de rendu abstraite (soit visuelle, soit acoustique).

Quand un document n'a pas d'élément layout, le positionnement des éléments de corps dépend de l'implémentation dans le corps du document SMIL.

Un document SMIL peut contenir de multiples dispositions alternatives en englobant plusieurs éléments layout dans un élément switch ; Ceci peut être employé, par exemple, pour décrire la mise en forme du document à l'aide de différents langages de mise en forme.

Si l'auteur d'un document SMIL souhaite sélectionner les valeurs de disposition par défaut pour tous les éléments des objets média, il doit introduire un élément layout vide dont la valeur de l'attribut type est "text/smil-basic-layout" tel que :

```
...
<layout type="text/smil-basic-layout"> </layout>
...
```

Figure II-3 : Exemple d'élément layout vide.

L'élément layout a comme attributs possibles, l'attribut ***id*** et ***type*** ; et il contiens les éléments suivants :

- region
- root-layout

Attribut	Valeur	Fonction
id	Caractères	Fixe l'identifiant du **layout**, et il est unique
type	Caractères ou "text/smil-basic-layout"	Cet attribut spécifie le langage de mise en forme utilisé dans l'élément **layout**. Si le lecteur ne reconnaît pas ce langage, il doit sauter tout le contenu jusqu'à la prochaine balise « </layout> ».

Tableau II-4 : Attribut de layout.

II.4. Le code

Quelques notions de base :

- Dans un fichier SMIL[1], le code ressemble à celui d'un fichier HTML. Il comporte donc des balises ouvrantes <balise> et fermantes </balise>. Cependant, contrairement au HTML, en SMIL, toute balise ouverte doit être fermée. Certaines balises sont dites "auto fermantes", c'est-à-dire qu'il n'y a qu'une seule balise, qui contient la barre oblique de fermeture à la fin (ex: <img />).
- Dans une balise, on peut trouver certains paramètres (attributs) qui permettent de définir plus précisément l'élément. Les attributs s'expriment de la manière suivante: attribut="valeur".
- Autre différence avec le langage HTML : les balises sont systématiquement en minuscules.
- Comme dans HTML, il est recommandé d'utiliser les commentaires (à écrire de la façon suivante : <!--Commentaires -->) pour expliquer à quoi correspondent les éléments.
- Un fichier SMIL s'enregistre avec les extensions "smi" ou "smil".

II.5. Structure du fichier [SMI 00]

Un fichier SMIL commence par <smil> et finit par </smil>. Entre ces balises, on trouve, comme dans un fichier HTML, une partie <head> et une partie <body>.

Exemple :

```
<smil>
  <head>
    (...)
  </head>
  <body>
    (...)
  </body>
</smil>
```

Figure II-1 : Exemple de structure d'un fichier SMIL.

L'élément ***smil*** admet les attributs suivants :

Attribut	Valeur	Fonction
id	Caractères	Identifie un élément de manière unique dans un document. Sa valeur est un identifiant XML.

Tableau II-1 : Attribut de smil.

L'élément **smil** accepte comme deux éléments fils, l'élément **head** et **body**

[1] Prononcé "smile".

II.5.1.2.1. L'élément root-layout

La fenêtre destinée à accueillir l'ensemble des animations est définie par l'élément vide <**root-layout**/>, puisque il ne contient pas d'éléments fils :

Exemple :

```
...
<root-layout width="300" height="200" background-color="blue"
title="Animations"/>
...
```

Figure II-4 : Exemple d'élément root-layout.

Cet élément permet de définir par ses attributs ***width*** et ***height*** la taille de la fenêtre de base, mais également sa couleur de fond, son titre, etc. A l'intérieur de cette fenêtre, tout comme à l'intérieur d'une fenêtre de navigateur, le positionnement d'un objet se fait par rapport au coin supérieur gauche. Cet élément admet les attributs: id, title, height, width, background-color.

Remarque :

Si un document contient plus d'un élément ***root-layout***, il s'agit d'une erreur et le document ne devrait pas être affiché.

II.5.1.2.2. L'élément region

L'élément ***region*** définit des zones de présentation pour les objets médias visuels, chacune décrivant les dimensions et la position d'une zone[2].

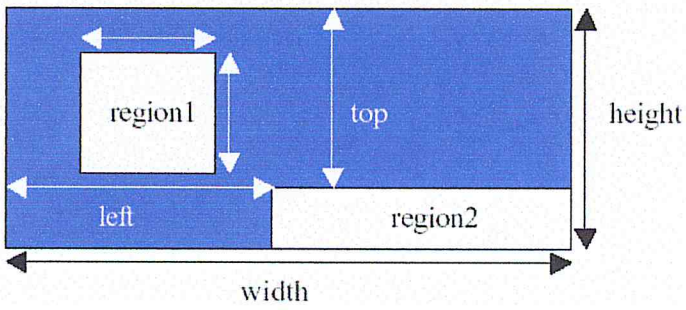


Figure II-5 : Exemple de région dans la zone d'affichage.

La balise <**region**> peut avoir les attributs suivants: *id, background-color* (couleur de fond), *title* (un titre pour la zone), *top, left, width, height, z-index, fit.*

2 Les zones peuvent se superposer.

Exemple :

```
...
<region id="region1" left="75" top="50" width="32" height="32"/>
...
```

Figure II-6 : Exemple d'élément region.

On attribue un nom à la zone (attribut **id**). Ceci permettra d'y faire référence dans la partie <**body**>, pour situer les éléments. Nous verrons plus loin dans le *chapitre-1* ses utilisations ; Les attributs ***top*** et ***left***, qui indiquent la position de la zone par rapport au coin supérieur gauche de la zone principale du fichier, ne sont pas obligatoires. Leur valeur est "0" par défaut, donc, s'ils ne sont pas spécifiés, l'élément sera positionné en haut à gauche de la zone principale SMIL.

L'élément ***region*** admet les attributs suivants :

Attribut	Valeur	Fonction
id	Caractères	Fixe l'identifiant de l'élément ***region.***
left	Pourcentage ou pixels	Fixe en absolu la coordonnée horizontale du coin supérieur gauche de region.
top	Pourcentage ou pixels	Fixe en absolu la coordonnée verticale du coin supérieur gauche de region.
right	Pourcentage ou pixels	Fixe en absolu la coordonnée horizontale du coin inférieur droit de region.
bottom	Pourcentage ou pixels	Fixe en absolu la coordonnée verticale du coin inférieur droit de region.
height	Pourcentage ou pixels	Fixe la hauteur de ***region.***
width	Pourcentage ou pixels	Fixe la largeur de ***region***.
z-index	Entier	Voir fonctionnalités de l'attribut fit dans la section : *5.1.2.2. L'élément region / Superposition des zones*
background-color	Identifiant de couleurs (red, green,…) ou code spécifiant la couleur.	Définit la couleur du fond de region.
fit	Fill, hidden, meet, scroll et slice; La valeur par défaut de fit est hidden.	Voir fonctionnalités de l'attribut fit dans la section : *5.1.2.2. L'élément region / Ajustement de la taille de l'objet*

Tableau II-5 : Attributs de region.

- ***Ajustement de la taille e l'objet***

L'attribut ***fit*** spécifie le comportement adopté si la hauteur et la largeur intrinsèques d'un objet média diffèrent des valeurs spécifiées pour les attributs ***height*** et ***width*** de l'élément ***region***, sa valeur par défaut est "***hidden***" , et il peut avoir les valeurs suivantes :

1. **slice**

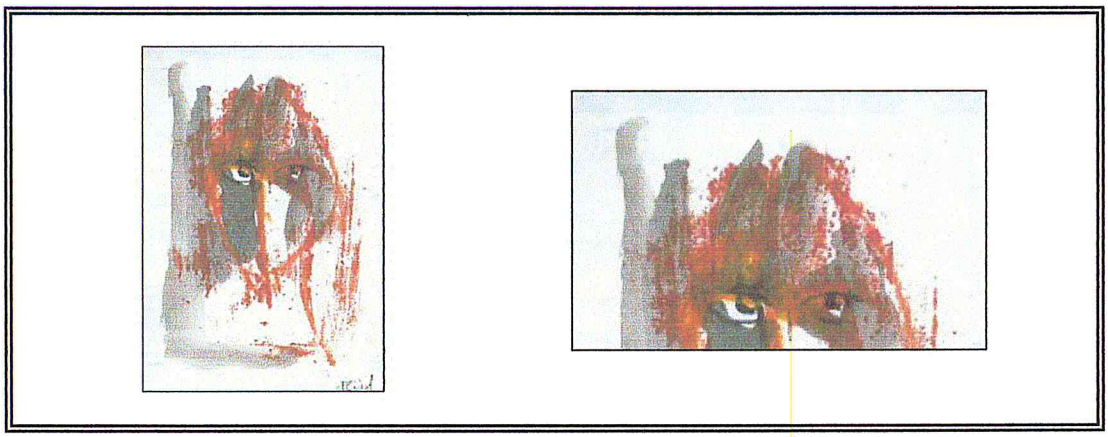


Figure II-7 : Exemple d'affichage slice.

Ajuste l'objet média visuel tout en préservant son ratio d'aspect de manière à ce que sa hauteur, ou sa largeur, soit égale à la valeur spécifiée par l'attribut height, ou width, des parties du contenu pouvant être rognées. Selon la situation donnée, une tranche de l'objet média soit horizontale, soit verticale, sera affichée. Un débordement en largeur est rogné à partir de la droite de l'objet média, un débordement en hauteur est rogné à partir du bas de l'objet média

2. **hidden**

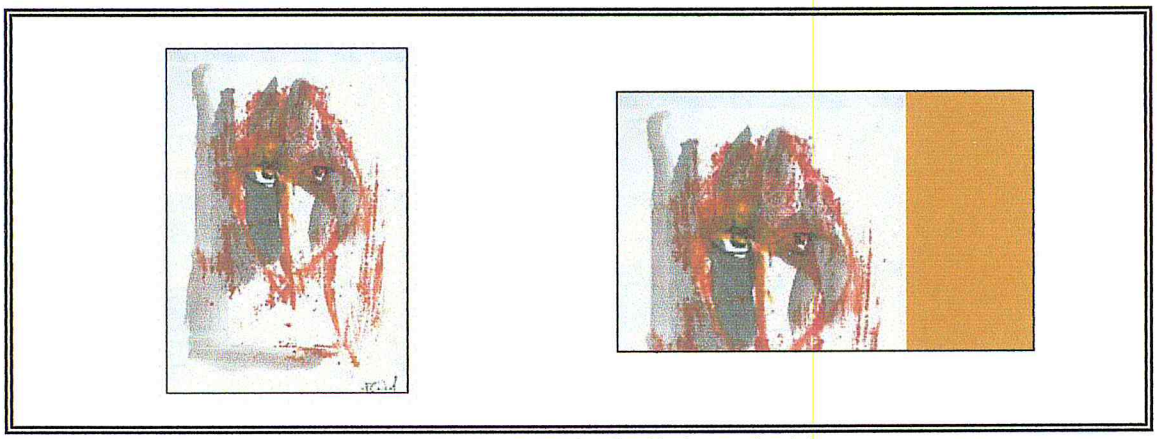


Figure II-8 : Exemple d'affichage hidden.

Si la hauteur (ou la largeur) intrinsèque de l'élément de l'objet média est inférieure à la valeur de l'attribut height (ou width) définie dans l'élément region, rend l'objet à partir du

bord du haut (ou de gauche) et remplit le reliquat de hauteur (ou de largeur) avec la couleur d'arrière-plan.

Si la hauteur (ou la largeur) intrinsèque de l'élément de l'objet média est supérieure à la valeur de l'attribut height (ou width) définie dans l'élément region, rend l'objet à partir du bord du haut (ou de gauche) jusqu'à ce que la valeur de height (ou de width) définie dans l'élément region soit atteinte, puis rogné la partie de l'objet situé en dessous (ou à la droite) de la hauteur (ou de la largeur).

3. meet

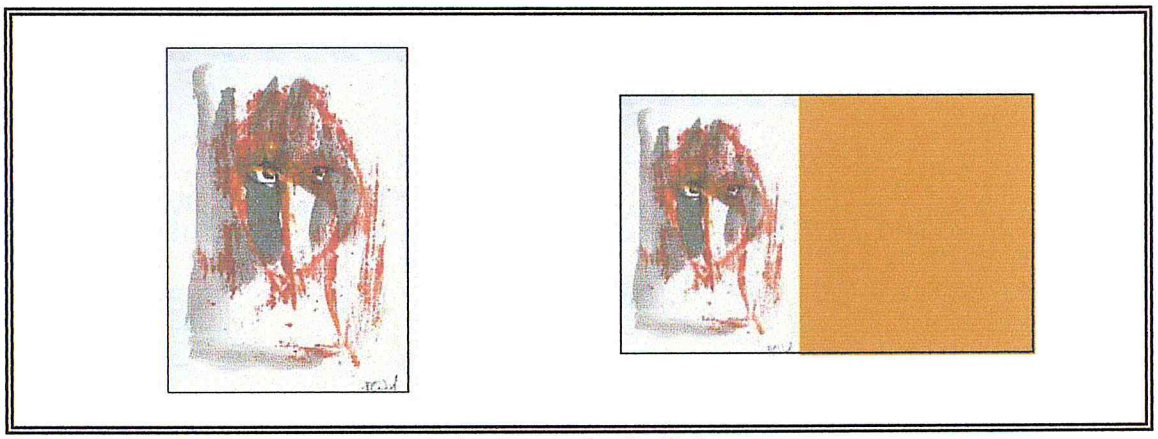


Figure II-9 : Exemple d'affichage meet.

Ajuste l'objet média visuel tout en préservant son ratio d'aspect jusqu'à ce que la hauteur, ou la largeur, soit égale à la valeur spécifiée par l'attribut height, ou width, aucune partie du contenu n'étant rognée. Le coin supérieur gauche de l'objet se place sur les coordonnées en haut à gauche de la boîte et l'espace vide à gauche ou en dessous est rempli par la couleur d'arrière-plan.

4. scroll

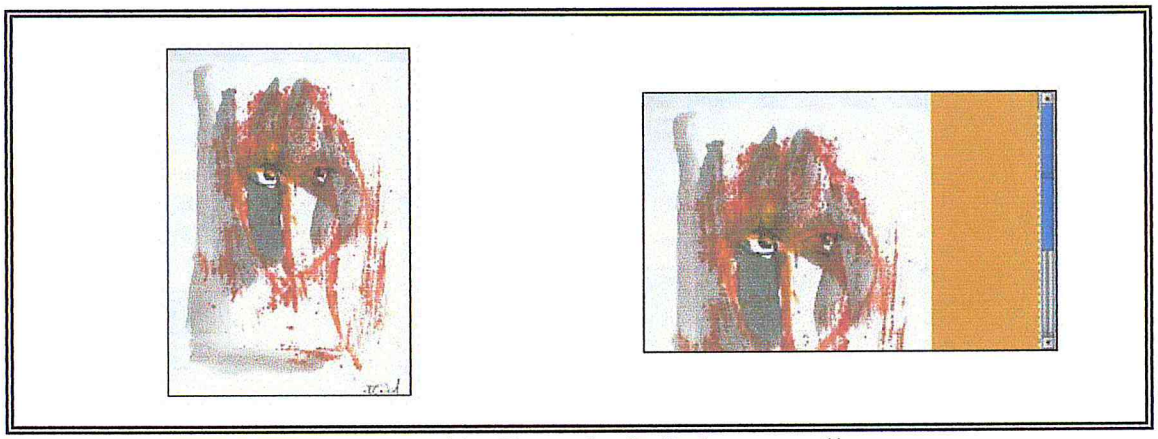


Figure II-10 : Exemple d'affichage scroll.

Si l'élément est plus grand que la région où il se trouve, des barres de défilement apparaîtront, permettant de faire défiler l'élément. L'élément garde ses dimensions, il n'est pas déformé.

5. fill

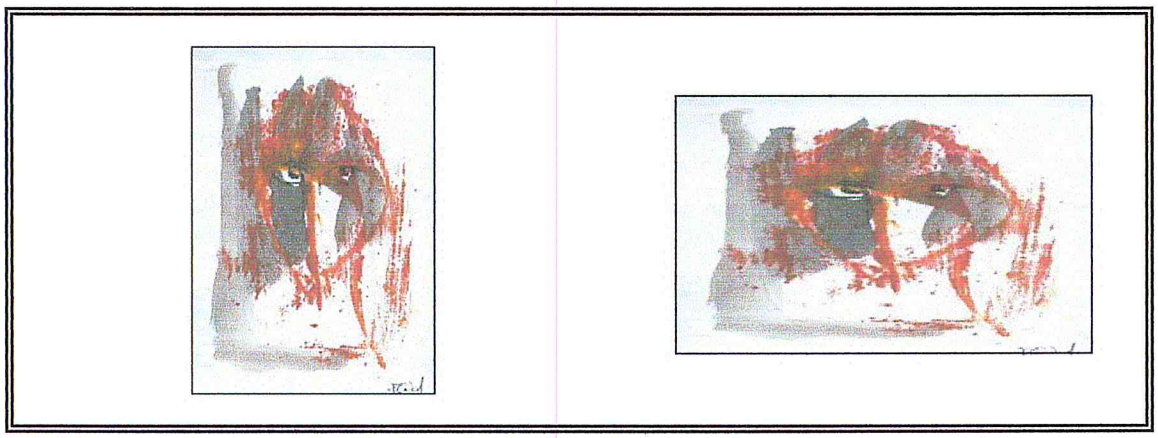


Figure II-11 : Exemple d'affichage fill.

L'élément est agrandi ou réduit pour remplir complètement la région où il se trouve. Une image pourrait être déformée.

- ***Superposition des zones***

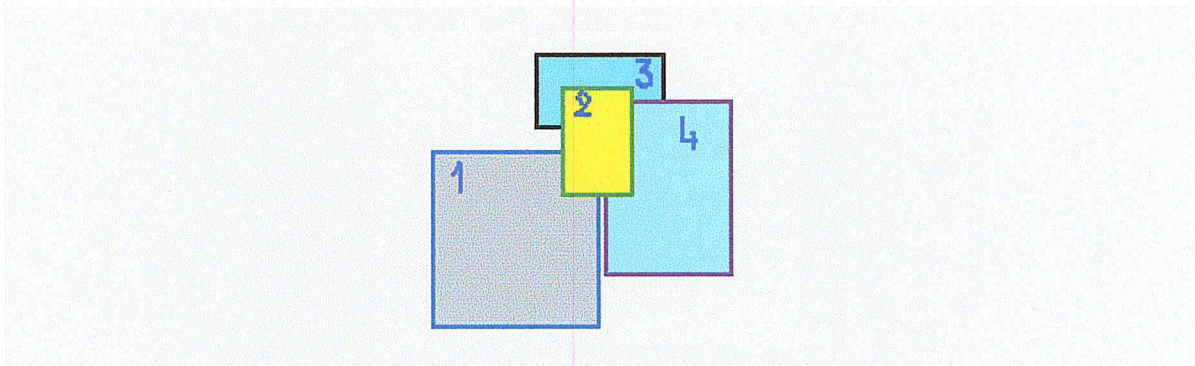


Figure II-12 : Superposition des zones.

Dans un cas où des zones (**région**) se superposent, un attribut ***z-index*** permet de préciser l'ordre de superposition des éléments ; La zone ayant le ***z-index*** avec la valeur la plus élevée sera au-dessus des autres. Si l'attribut ***z-index*** n'est pas précisé, et que des zones se superposent, la dernière zone dans le code sera au-dessus des autres.

Exemple :

On remarque dans l'exemple qui suit, que les régions définis ; leurs limites s'empilent les uns aux autres, pour cela on a donné une priorité à chaque région pour quelle soit visible vis-à-vis d'une autre région, dans cet exemple on a une priorité de l'ordre : région 2, région 4, région 1, région 3.

```
...
<region id="region1" left="75" top="50" width="100" height="150"
z-index="3"/>
<region id="region2" left="100" top="175" width="40" height="60"
z-index="1"/>
<region id="region3" left="60" top="58" width="60" height="40"
z-index="4"/>
<region id="region4" left="40" top="80" width="80" height="150"
z-index="2"/>
...
```

Figure II-13 : Exemple d'élément z-index.

II.5.1.3. L'élément switch (head)

C'est un élément de choix qui permet à l'auteur de spécifier une suite d'éléments parmi lesquels seulement un sera retenu. L'évaluation porte sur des attributs de test qui retournent comme résultat un booléen, le premier élément dont l'attribut renvoi la valeur vraie est sélectionné ; Les autres sont ignorés.

Cet élément admet les attributs suivants :

Attribut	Valeur	Fonction
id	Caractères	Fixe l'identifiant de l'élément **switch.**
title	Caractères	Donne un titre pour **switch**.

Tableau II-6 : Attributs de switch.

Remarque :

Si l'élément **switch** est utilisé dans un élément **head**, il peut contenir l'élément fils **layout** ; celui-ci peut survenir dans l'élément **switch** plusieurs fois.

II.5.2. L'élément body

Cette partie appelée le corps du document SMIL contient les éléments multimédia eux-mêmes, indique la localisation des fichiers composant le document SMIL, et le comportement[3] de chacun, et il admet un seul attribut.

Attribut	Valeur	Fonction
id	Caractères	Fixe l'identifiant du **body**, et il est unique

Tableau II-7 : Attributs de body.

[3] La durée et le moment d'apparition.

L'élément body peut contenir plusieurs éléments fils présenté dans le tableau en dessous :

Élément	Fonction
img, audio, text, textstream, video, animation, ref	Définissent le type d'objet média.
seq	Permet d'apposer en parallèle un groupe d'éléments.
par	Permet d'apposer en séquence un groupe d'éléments.
a	Définit un lien de navigation.
switch	Spécifie un groupe d'éléments alternatifs, selon l'évaluation de l'attribut de test.

Tableau II-8 : Éléments fils de body.

II.5.2.1. Objets médias de base

Les éléments objets médias permettent d'introduire des objets médias dans une présentation SMIL. Les objets média sont inclus par référence (en utilisant un URI).

Il existe deux types d'objets média :

- Ceux avec une durée intrinsèque, par exemple vidéo et audio[4].
- Ceux sans durée intrinsèque, par exemple texte et image[5].

Objet média	Élément	Format (extensions)
img	<img/>	Images (.jpg, .gif, .png).
audio	<audio/>	Fichier son (.rm, .wav, .aif, .mov, .mp3).
text	<text/>	Référence textuelle (.txt).
textstream	<textstream/>	Flux de texte (.rt).
video	<video/>	Clip vidéo ou real video (.rm, .avi, .mov, .asf, .viv, .mpeg).
animation	<animation/>	Clips d'animation tel que les fichiers flashplayer (.swf).
ref	<ref/>	Une référence sur un objet média (.rp).

Tableau II-9 : Éléments média de base.

Exemple :

Pour chaque objet média on a un élément qui le référence comme pour introduire l'objet audio *musique1* on utilise *audio,* ou encore l'élément *img* pour un objet de type image.

[4] Désignés aussi par « médias continus ».
[5] Désignés aussi par « médias discrets ».

```
...
<ref src="n'importe-qu'elle-média.???" ... />
<text src="file.html" ... />
<textstream src="stock.rtx " ... />
<img src="graph.jpg" ... />
<audio src="musique1.wav" .../>
<video src="http://www.biblio.org/SYMM/vid.rm" .../>
<animation src="cute.anim" .../>
...
```

Figure II-14 : Exemple de référence d'objet média.

Ces éléments équivalents d'objets média, admettent les attributs suivants :

Attribut	Valeur	Fonction
id	Caractères	Contient l'identifiant unique de l'objet.
title	Caractères	Indique le titre du média.
region	Un identifiant de region	Spécifie l'identifiant de la surface d'affichage attribuée à l'objet.
alt	Caractères	Contient un texte de remplacement pour les média qui ne peuvent pas être visualisés.
src	Adresse URI	Sert à localiser et récupérer le média associé à un emplacement (URI).
begin	Un délai ou un événement	Définit l'instant de l'activation de l'élément
end	Un délai ou un évènement	Définit l'instant de fin de l'élément
dur	Un délai ou un événement	Spécifie la durée d'un élément média.
clip-begin	Voir dans la section	Indique le commencement d'une sous-séquence d'un objet média continu.
clip-end	Voir dans la section	Indique la fin d'une sous-séquence d'un objet média continu.
repeat	Entier	Indique q'un élément doit être rejoué pour un nombre spécifique de fois.
fill	**freeze** ou **remove**	Utiliser pour la persistance d'un objet média sur l'écran, il peut contenir les valeurs : freeze : garder (figer) la dernière image sur l'écran après sa terminaison. remove : effacer l'objet de l'écran dès sa terminaison. NB : par défaut elle prend la valeur remove.
Attribut de test	Voir valeurs et fonctionnalités des attributs de test dans la section : *6. Attributs de test :*	

Tableau II-10 : Attributs d'éléments média.

II.5.2.2. Éléments de lien

II.5.2.2.1. L'élément a

Tout comme dans HTML, l'élément <**a**> permet de gérer les liens. Par contre avec le langage SMIL, l'avantage c'est qu'il lui rajoute un attribut **show** qui contrôle le comportement temporel de la source bien sur quand le lien est suivi de l'attribut **show**.

Remarques :

Pour des raisons de synchronisation, l'élément **a** est transparent, i.e., il n'influe pas sur la synchronisation de ses éléments enfants. Les éléments **a** ne peuvent pas être imbriqués, et il doit avoir un attribut href, qui contient adresse l'URI de la destination du lien.

Exemples :

Dans cet exemple, l'exécution de la deuxième ligne remplace le lien en cours par une référence décrite dans la première ligne, vers n'importe quel sous arbre valide d'une présentation SMIL.

Le lien lance une nouvelle présentation en plus de celle en cours de lecture.

Par exemple, cela permet à un lecteur SMIL de se diviser en dehors d'un navigateur HTML.

```
<a href="http://www.archive.com/test1.smi" show="new"/>
     <video src="rtsp://foot.com" region="fenêtre2"/>
</a>
```

Figure II-15 : Exemple avec la valeur new.

II.5.2.2.2. L'élément anchor

L'élément <**anchor**> a pour effet de rendre cliquable une partie d'élément visuel seulement.

Si l'on souhaite qu'une partie "spatiale" d'un élément visuel soit cliquable, on utilise l'attribut **coords**, qui va définir le rectangle cliquable de la façon suivante: Les deux premières valeurs vont déterminer le coin supérieur gauche du rectangle, et les deux dernières vont déterminer le coin inférieur droit. Les coordonnées sont relatives au coin supérieur gauche de l'élément. Chaque valeur est séparée de la suivante par une virgule. On peut fournir les coordonnées en pourcentage comme pour les régions dans le **layout**.

Remarques :

- Il est également possible grâce à l'élément ***anchor*** de séparer un objet en sous partis temporels. On va pour cela utiliser les attributs ***begin*** et ***end***. Il ne faut pas oublier de nommer chaque sou parti au moyen de l'attribut ***id***.
- Les valeurs de l'attribut ***coords*** suivent la syntaxe suivante :

```
valeur-coords ::= left-x "," top-y "," right-x "," bottom-y
```

Exemple :

La région occupée par le *Clip-vidéo* est partagée en deux sections, un lien différent est associé à chacune d'elles.

```
...
<video src=" http://www.archive.org/clip"/>
 <anchor id="zone1" href="http://www.archive.org/ClipAlgeria"
                    coords="0, 0, 250, 250"/>
<anchor id="zone2" href="http://www.archive.org/ClipUniversité"
                   coords="250, 250, 0, 0"/>
</video>
...
```

Figure II-16 : Association de liens aux sous-ensembles spatiaux.

La durée du clip vidéo est partagée en deux sous intervalles, un lien différent est associé à chacun d'eux.

```
...
<video src=" http://www.archive.org/clip"/>
     <anchor href="http://www.archive.org/ClipAlgeria"
                              begin="0s" end="5s"/>
     <anchor href="http://www.archive.org/ ClipUniversité "
                              begin="5s" end="10s"/>
</video>
...
```

Figure II-17 : Association de liens aux sous-ensembles temporels.

II.5.2.3. Éléments de synchronisation

En général, les éléments sont inclus dans les balises **<par>** pour "parallèle", ou **<seq>** pour "séquence", qui servent à gérer la synchronisation des éléments ; Mais, ils peuvent aussi être insérés directement dans la balise **<body>**. Ils apparaissent alors les uns après les autres, dans l'ordre où ils sont dans le code.

II.5.2.3.1. L'élément par

L'élément **par** est utilisé lorsque au moins deux éléments doivent être diffusés simultanément. Leur durée déroulement peut varier, mais par défaut, le moment du début est le même pour tous les éléments inclus entre les balises **<par>** et **</par>**.

L'élément **par** peut avoir les attributs suivants :

Attribut	Valeur	Fonction
id	Caractères	Contient l'identifiant unique de l'élément.
title	Caractères	Indique un titre.
begin	Un délai ou un événement	Spécifie l'instant du début explicite d'un élément.
end	Un délai ou un évènement	Définit l'instant de fin de l'élément
dur	Un délai ou un événement	Spécifie la durée d'un élément.
endsync	first, id (valeur-id), valeur par défaut est last.	Définit l'instant de fin de l'élément
repeat	Entier, valeur par défaut est 1	Indique qu'un élément doit être rejoué pour un nombre spécifique de fois.
Attribut de test	Voir valeurs et fonctionnalités des attributs de test dans la section : *6. Attributs de test :*	

Tableau II-11 : Attributs de l'élément par.

L'élément **par** peut contenir plusieurs éléments fils présenté dans le tableau en dessous :

Élément	Fonction
img, audio, text, textstream, video, animation, ref	Définissent le type d'objet média.
seq	Permet d'apposer en parallèle un groupe d'élément.
par	Permet d'apposer en séquence un groupe d'élément.
a	Définit un lien de navigation.
switch	Spécifie un groupe d'éléments alternatifs, selon l'évaluation de l'attribut de test.

Tableau II-12 : Éléments fils de par.

Exemple :

La présentation de ces trois objets en parallèle se termine à la fin de la durée d'affichage de l'objet texte, donc le groupe ***par*** se termine quand le premier objet multimédia de type *texte* achèvera les cinq secondes.

```
...
<par endsync="first">
  <text src="sous-titre.rt" region="zone-1" dur="5s" />
  <video id="v1" src="doc.mpg" region=" zone-2" begin="2.5s" />
  <audio src="audio.wav" region="music" begin="id(v1)" />
</par>
...
```

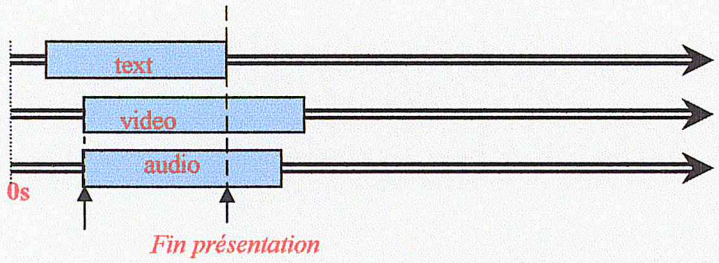


Figure II-18 : Exemple d'utilisation de l'attribut endsync avec la valeur first.

Cependant la présentation suivante des mêmes trois objets en parallèle, se termine quand l'objet référencier (`endsync="id(v1)"`) se termine, donc le groupe ***par*** se termine, quand la vidéo finisse son déroulement.

```
...
<par endsync="id(v1)">
<text src=" sous-titre.rt" region="zone-1" dur="5s" />
<video id="v1" src="doc.mpg" region="zone-2" begin="2.5" />
<audio src="audio.wav" region="music" begin="id(v1)(3.5s)" />
</par>
...
```

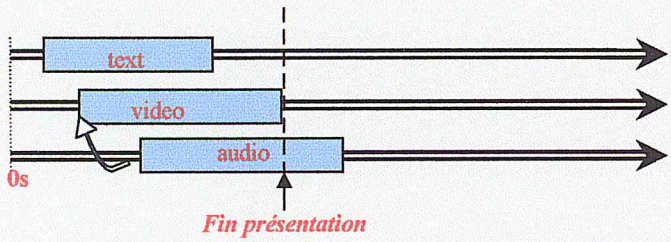


Figure II-19 : Exemple d'utilisation de l'attribut endsync avec référence.

II.5.2.3.2. L'élément seq

Les éléments inclus entre les balises <**seq**> et </**seq**> sont diffusés les uns après les autres, dans l'ordre où ils apparaissent dans le fichier source.

L'élément **seq** admet les attributs suivant (***abstract, author, begin, copyright, dur, end, id, region, repeat, Attributs de test***) se sont les mêmes attributs avec les même valeurs et fonctionnalités de l'élément ***par***, sauf ***endsync***, qui n'est pas un attribut de ***seq***, et le contenu de l'élément et le même de l'attribut ***par***.

Remarque :

Un élément **seq** se termine quand son dernier fils se termine, si son dernier fils a une fin indéterminée alors l'élément **seq** a une fin indéterminée.

Exemples :

Le bloc *seq* se termine quand l'élément *text* se termine.

```
...
<seq>
  <video src="Clip-1.mpg" region="zone-1" />
  <video src=" Clip-2.mpg" region="zone-1" />
  <text src="titre.rt" region="zone-3" dur="indéfinis" />
</seq>
...
```

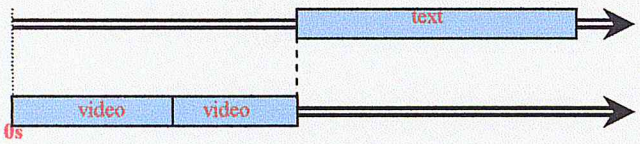


Figure II-20 : Exemple d'utilisation de l'élément seq.

```
<smil>
  <head>
    <layout>
      <region id="zone" height="600" width="800"/>
    </layout>
  </head>
  <body>
    <switch>
      <seq system-bitrate="44000">
           <img dur="5s" src="green.jpg" region="zone"/>
       </seq>
       <seq>
           <img dur="5s" src="blue.jpg"  region="zone"/>
       </seq>
    </switch>
  </body>
</smil>
```

Figure II-21 : Exemple d'utilisation d'attribut de teste avec seq.

II.5.2.4. L'élément switch

L'élément **switch** est utilisé pour proposer différents choix à l'utilisateur, par exemple, un texte ou un enregistrement vidéo en différentes langues. La balise <**switch**> englobe alors les différents éléments proposés. Nécessairement Chaque élément contient un attribut de test. Les tests seront effectués les uns après les autres jusqu'à l'élément qui correspond à l'environnement de l'utilisateur, duquel un seul qui soit acceptable devrait être choisi.
Cet élément admet les attributs suivants :

Attribut	Valeur	Fonction
id	Caractères	Fixe l'identifiant de l'élément **switch.**
title	Caractères	Donne un titre pour **switch**.

Tableau II-13 : Attributs de switch.

L'élément **switch** peut contenir plusieurs éléments fils présenté dans le tableau qui suit mais ces éléments doivent contenir tous un attribut de test.

Élément	Fonction
img, audio, text, ref textstream, video, animation,	Permettent l'inclusion d'objets média dans un document SMIL.
seq	Permet d'apposer en parallèle un groupe d'élément.
par	Permet d'apposer en séquence un groupe d'élément.
switch	Spécifie un groupe d'éléments alternatifs, selon l'évaluation de l'attribut de test. Voir section : *6. Attributs de test :*

Tableau II-14 : Éléments fils de switch.

II.6. Attributs de test

Généralement les attributs de test sont introduits avec n'importe quelle élément média de base ou élément de synchronisation ou encore dans un bloc ***switch***, ainsi l'élément de lien **a,** afin d'évaluer les capacités et les paramétrages des systèmes, ces attributs effectuent des tests booléens. Quand l'un des attributs de test spécifiés sur un élément est évalué à une valeur de "false", alors l'élément porteur de cet attribut est ignoré. On mentionne brièvement comme attribut de test :

- ***system-language*** : C'est la langue de la ressource, la valeur de l'attribut est le nom de langue, par exemple pour anglais (en), français (fr), allemand (de), hollandais (nl).

- ***system-bitrate*** : Permet de tester la bande passante de la connexion réseau de l'utilisateur.
- ***system-screen-size*** : Teste si l'écran est capable d'afficher la taille spécifiée, ça valeur et de la forme : hauteur-écran" x "largeur-écran en pixel.

II.7. Horloge SMIL

L'horloge SMIL, propre à l'application de présentation, cette horloge fonctionne sous le contrôle de l'utilisateur, d'une manière qu'il puisse l'arrêter et de figer la présentation, de la faire avancer en accéléré ou de la faire revenir en arrière.

- **Expression des durées sur l'horloge SMIL**

Deux syntaxes permettent d'exprimer des durées d'horloge

- La première syntaxe utilise l'abréviation (h, m, s, ms)

1. *Valeurs de durée :*

   `1.5h` = 1 heures 30 minutes.

   `3.75min` = 3 minute et 45 secondes.

   `40.55s` = 40 secondes et 550 millisecondes.

   `260ms` = 260 millisecondes.

Exemple :

```
...
<audio src="dossier/musique/clip2.mp3"dur="4.25min"/>
<img src="dossier/bmp/nature.jpg"dur="03:45"/>
...
```

Figure II-22 : Exemple de syntaxes d'horloge avec « h, m, s, ms ».

2. *Valeur d'horloge complète :*

   `01:20:41` = 1 heures, 20 minutes et 41 secondes.

3. *Valeur d'horloge partielle :*

   `05:23` = 5 minutes et 23 secondes.

- La seconde syntaxe se décompose en deux modèles, chacun à un spécifiant métrique différent (npt, smpte)

4. *Le modèle npt :*

Appelée *temps de lecture normal* [Normal Play Time], puisqu'il est exprimé en fonction des valeurs d'horloge SMIL, il est utilisé avec les attributs **begin, end, clip-begin, clip-end**, ça syntaxe est de la forme: `"npt=hh:mm:ss.xyz"`

x = Dixième de seconde,

y = Centième,

z = Millième

L'indication des heures est optionnelle, puisque si ça valeur est égale à zéro, on peut ne pas l'écrire comme suit :

"npt=00:04:18" → "npt=04:18" (4 minutes 18 secondes) "npt=00:18.60" → "npt=18.60" (18 secondes et 600 millisecondes)

Exemple :

```
...
<audio src="boublage-ar.aif" begin="0.230min" />
<audio src="doublage-fr.aif" begin="npt=00:15.999" />
<video src="film.mpg" clip-begin="npt=00:15.999" />
...
```

Figure II-23 : Exemple de syntaxes d'horloge avec « npt=hh:mm:ss.xyz ».

5. *Le modèle smpte :*

- **smpte :**

Le spécifiant métrique **smpte** est utilisé avec des objets média continue de type vidéo puisque c'est une suite d'images qui passent avec une vitesse précise par exemple 30 images par seconde ou moins pour que le clip vidéo soit présenté en temps d'affichage réel, pour plus de contrôle de l'instant début d'affichage on utilise ce modèle de syntaxe, puisque si on utilise cette écriture clip-begin="npt=04:18" on sais que le clip vidéo débute à la 4ème minute et 18 seconde,mais on sais que pour une seconde de temps on a 30 images qui ce défilent, donc pour cela on utilise la deuxième notion de syntaxe qui permet de choisir parmi c'est 30 images celle voulue. Ça syntaxe est de la forme: "smpte=hh:mm:ss:im.si"

im = nombre d'image qui défile par seconde de la vidéo.

si = le numéro de la sous images de la seconde choisi "ss".

On a aussi deux autres normes **smpte** :

- **smpte-30-drop :**

C'est le même principe sauf qu'ici on a un défilement de 30 images par seconde donc la valeur des sous images accepte des valeurs de 0 à 29, et on n'a pas besoin de spécifier le champ **im**, qui aura la syntaxe suivante : "smpte=hh:mm:ss:si"

- **smpte-25 :**

C'est le même principe sauf qu'ici, on a un défilement de 25 images par seconde donc la valeur des sous images accepte des valeurs de 0 à 24, et on n'a pas besoin de spécifier le champ **im**, aura la syntaxe suivante : `"smpte=hh:mm:ss:si"`

Remarque :

Si la valeur d'image `im` = 0, alors les sous images se mesurent en centièmes d'image, **si**= `[0, 99]`, comme si on avait : **im=100** images par seconde.

Exemple :

Dans cet exemple l'objet média vidéo est découpé avec les attributs **clip-begin** et **clip-end** en une séquence vidéo, et il n'y a que cette séquence qui va être jouée, et elle débutera à la 19ème seconde et de la 10ème image.

```
...
<video src="film.mpg" clip-begin="smpte=00:15:19:24.10"      clip-
end="npt=00:30:24.56" region="fenêtre-1" fill="freeze"/>
...
```

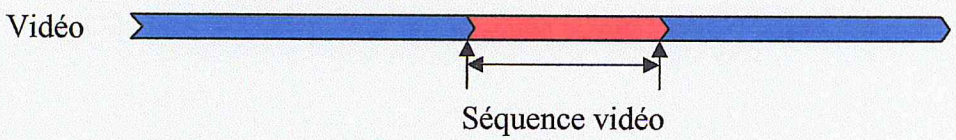


Figure II-24 : Exemple de découpage d'un objet média continue.

II.8. Le modèle temporel

Chaque élément a forcément des attributs de temps, même s'ils ne sont pas donnés par l'utilisateur dans certain cas, donc on a le début, la durée et la fin, et qui sont soit implicites, explicites ou effectives.

Le début, la durée et la fin effective spécifient ce que l'utilisateur va percevoir, dans ce cas là il n'a pas de contrôle sur ces valeurs de temps.

Les valeurs implicites et explicites sont des valeurs additionnelles introduites pour définir les valeurs effectives.

Les valeurs implicites sont propres à l'objet média et sont gérées par le langage SMIL.

Les valeurs explicites sont tout ce que l'utilisateur introduit comme paramètres de temps (begin, end, dur).

```
begin = "valeur de début explicite"
end = "valeur de fin explicite"
dur = "valeur de durée explicite"
```

Exemples :

Dans cet exemple l'utilisateur n'a pas donné de valeur pour début, fin ou durée, donc elles sont implicites et on peut dire aussi effectives puisqu'ils sont équivalents.

```
...
  <body>
    <video src="VR3.mpg" region="region1" />
  </body>
...
```

Figure II-25 : Exemple de valeur implicite.

Dans cet exemple l'utilisateur définit un début et une durée à l'objet vidéo, donc ce sont des valeurs explicites.

```
...
  <body>
    <video src="VR3.mpg" region="region1" begin="00:12:18
           dur="15min" />
  </body>
...
```

Figure II-26 : Exemple de valeur explicite.

II.9. Les fichiers Real-Text

Ce sont des fichiers qui se jouent seulement avec un lecteur multimédia de type **Real** et ils ont une extension "**.rt**", ils servent à afficher des textes dans une région d'un root-layout, ces textes sont de deux types, (texte fixe, texte dynamique car il défile dur l'écran)

Pour crée un fichier Real-Text il suffit d'introduire le texte dans la partie spécifier dans le code en dessous et fixer les paramètres (*écrite en bleu dans l'exemple*) de police et temporel, taille, couleur, duration, begin,...; Le fichier sera sauvegardé à la fin sous l'extension **.rt.**

II.9.1. Texte fixe

L'exécution de ce fichier va permettre de générer une séquence de trois blocs de texte comme un générique de film.

```
<window type="generic"
      duration="0:46.5"
      width="270" height="95" />
      <font face="arial" size="3" >
      <time begin="0" end="9.0" /> <pos x="2" y="5" />

                  --------Tapez votre texte 1-------

      <time begin="9.0" end="14.0" /> <pos x="2" y="5" />

                  --------Tapez votre texte 2-------

      <time begin="14.0" end="17.0" /> <pos x="2" y="5" />

                  --------Tapez votre texte 3-------

         </font>
</window>
```

Figure II-27 : Exemple du code pour Real text fixe.

Remarque :

Pour la génération de fichier de type *Text* il suffit de modifier le programme précédant de la *Figure II-28,* et ça selon nos besoins personnels d'affichage et de synchronisation, or on écris se programme à l'aide d'un éditeur de texte simple comme *Bloc-notes* et on lesauvegarde avec une extension « .txt ».

II.9.2. Texte dynamique

L'exécution de se fichier va permettre de générer un défilement du texte dans un couloir spécifié par l'utilisateur. On remarquera le défilement des trois blocs de texte en séquence du premier au dernier.

```
<window
      type="tickertape" duration="3:00" bgcolor="#444444"
      underline_hyperlinks="false"
      link="red" crawlrate="40" loop="false"
      width="450" height="20">
      <tl color="#cfb59c">
         <time begin="0"/> <clear/> <br/> <font face="arial">

               ----------Tapez votre texte 1 -------

         </font>
         <time begin="30"/> <clear/> <br/> <font face="arial">

               ----------Tapez votre texte 2 -------

         </font>
         <time begin="1:00"/> <clear/> <br/> <font face="arial">

               ----------Tapez votre texte 3 -------

         </font>
      </tl>
</window>
```

Figure II-28 : Exemple du code pour Real text dynamique.

Remarque :

De même que la remarque faite à propos des fichiers de type *texte fixe*, pour la génération de fichier de type *Real-Text* il suffit de modifier le programme de la *Figure II-29*, et ça selon nos besoins personnels d'affichage et de synchronisation, or on écris se programme à l'aide d'un éditeur de texte simple comme *Bloc-notes* et on le sauvegarde avec une extension « .rt »

II.10. Intégrer SMIL dans une page HTML

Pour intégrer un fichier SMIL dans une page web, de façon à ce que le fichier soit lu directement dans le navigateur en faisant appel au module externe nécessaire et non pas dans une fenêtre d'un player comme RealPlayer, il faut utiliser les balises <object> ou <embed>. Ces deux balises ont le même but, mais les navigateurs Netscape et Internet Explorer reconnaissent chacun l'une d'entre elles seulement. Voici le code qu'il faut taper dans la page HTML:

```
<OBJECT ID=RVOCX CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"
height="600" width="600">
<PARAM NAME="src" VALUE="Nom_Fichier_Smil.smi">
<PARAM NAME="CONTROLS" VALUE="ImageWindow">
<PARAM NAME="CONSOLE" VALUE="one">
<param name="autostart"  VALUE="true">
<EMBED SRC=" Nom_Fichier_Smil.smi" WIDTH=600 HEIGHT=600 NOJAVA=true
CONTROLS=ImageWindow CONSOLE=one>
 </OBJECT>
```

Figure II-29 : Exemple d'intégration du Smil dans une page Web.

II.11. Les outils nécessaires

Pour travailler avec SMIL, 2 types d'outils sont nécessaires:

- Les éditeurs
- Les players

Les éditeurs :

SMIL, tout comme HTML ou bien d'autres langages, peut être créé avec un éditeur de texte simple, tel que Bloc-Notes. Mais bien sûr, il peut être long et fastidieux de taper tout le code à la main, sans compter le risque d'erreurs de frappe. C'est pourquoi il est intéressant d'utiliser des éditeurs spécialisés. En voici une liste de quelques éditeurs :

- Tagfree 2000 SMIL Editor
- SMIL-Editor
- Fluition
- SMILGen

- ***Les players :***

SMIL n'est pas lisible directement par les navigateurs, il est nécessaire d'avoir un module externe pour lire les fichiers correctement. Comme on l'a vu plus haut, il est quand même possible d'afficher du SMIL dans un navigateur en faisant appel à un module externe. Voici la liste des modules externes utiles pour afficher SMIL:

- RealPlayer G2
- RealOne Player, de RealNetworks.
- Apple Quicktime 5

Conclusion

On a présenté dans ce chapitre la description de la première version du langage SMIL qui se résume en la composition de sa structure tant que sur le plan spatial et temporel. Concernant l'aspect spatial défini dans l'entête du document, on peut dire qu'il englobe toutes les déclarations des *régions* dans une fenêtre principale *root_layout*. Alors que l'aspect temporel défini dans le corps du document, sert à décrire la synchronisation entre les objets médias grâce aux éléments *seq*, et *par*, qui permettent une spécification des comportements plus complexes.

Dans le document SMIL, on trouve aussi des éléments permettant à l'auteur une meilleure représentation tels que *switch* ou encore des éléments hyperliens.
Enfin, l'emplacement temporel des éléments de base du langage SMIL, nous amène à remarquer quelques limitations telles que la probabilité d'avoir des cycles en introduisant les attributs temporels événementiels ou la référence à des objets inexistants, telles situations renvoient des documents incohérents.

Néanmoins, SMIL est un langage de synchronisation des objets médias qui peut servir à la création de scénarios très complexes, et contrairement à d'autres langages, il ne demande aucune connaissance particulière en programmation. De plus, il offre aussi l'avantage d'être indépendant du système d'exploitation des machines sur lesquelles il est utilisé.

Chapitre III

LES TECHNIQUES D'ANALYSE DES DOCUMENTS MULTIMÉDIAS

Chapitre III

LES TECHNIQUES D'ANALYSE DES DOCUMENTS MULTIMÉDIAS

On a vu dans le chapitre précédent que le langage SMIL permet la synchronisation temporelle entre plusieurs objets média. Mais une relation temporelle mal exprimée entre ces objets rend le document incohérent. Ainsi, une description précise et cohérente conditionne, en effet, la restitution correcte du document à l'utilisateur. Donc, il est nécessaire d'adopter une analyse descendante des différents problèmes.

C'est dans cette perspective qu'on propose, dans ce chapitre, d'aborder les principales approches dédiées à la cohérence des documents multimédia. Ces approches vont porter principalement sur le processus de composition temporelle.

III.1. Approche basée sur les problèmes de satisfaction de contraintes (CSP)

Les problèmes de satisfaction de contraintes (CSP) ou réseaux de contraintes sont un modèle maintenant classique de représentation et de résolution des problèmes dans plusieurs domaines de l'informatique. Ils fournissent un cadre simple et formel, dans lequel sont développées des techniques de raisonnement automatique. Donc de nombreux problèmes liés à l'informatique peuvent facilement être formulés en termes de **CSP**, et résolus.

Un problème de satisfaction de contraintes comporte :

- Des ***variables*** x_i, dont chacune possède un ***domaine*** de valeurs admissibles d_i.
- Des ***contraintes*** c_j sur les valeurs/combinaisons des valeurs des variables.

Formellement, un problème de satisfaction de contraintes binaires (agit sur la combinaison des valeurs d'un couple de variables) est défini par P = (X, D, C, R), tel que :

- $X = \{x_1, ..., x_n\}$ est un ensemble de variables.
- $D = \{d_1, ..., d_n\}$ est un ensemble de domaines, où D_i est le domaine de la variable X_i.
- $C = \{c_{i,j} / 1 \leq i, j \leq n, c_{i,j} = \{(x_{i1}, x_{j1}),\ldots, (x_{ip}, x_{jp})\}$ est un ensemble de contraintes sur l'espace $D_i \times D_j$, $1 \leq p \leq n\}$.
- $R = \{R_{i,j} / 1 \leq i, j \leq n\}$ est un ensemble de relations. $R_{i,j}$ est l'ensemble des combinaisons de valeurs qui satisfont $C_{i,j}$.

Trouver une solution à un CSP revient à affecter à chaque variable x_i une valeur de son domaine d_i de manière à ce que toutes les contraintes c_j soient respectées.

Nous allons d'abord définir ce qu'est une contrainte, ensuite nous parlerons des problèmes de satisfaction de contraintes temporelles.

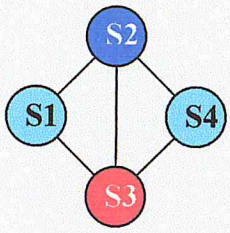
III.1.1. Notion de contrainte

Pour comprendre la notification «CSP», il vaut mieux savoir qu'est-ce qu'une contrainte ? Pour cela, on va énoncer cet exemple pour avoir une vue globale d'une contrainte.

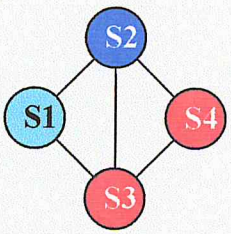
Exemple :

Dans cet exemple, nous avons un ensemble de points que nous appelons sommets ; certains de ces sommets sont reliés entre eux par des droites que nous appellerons arrêtes. Nous décidons d'attribuer une couleur à chaque sommet. Nous nous imposons une première contrainte : nous n'avons que trois couleurs. Cette restriction est du même type que celle vue à l'introduction du chapitre avec la fonction racine carré. Dans notre nouveau

problème, le but consiste à ce que chaque paire de sommets reliés par une arrête n'ait pas la même couleur, pour cela, sur un sommet S, nous devons vérifier la couleur des sommets qui sont reliés à S, et lui attribuer une couleur adéquate.



Graphe1



Graphe 2

Figure III-1 : Exemple de représentation de contrainte sous forme de graphe.

Dans le *Graphe1* de la *figure III-1*, toutes les contraintes sont satisfaites, aucuns des sommets qui sont reliés par des arrêtes n'ont la même couleur. Mais par contre dans le *Graphe 2*, une contrainte a été enfreint : les sommets S2 et S3 qui sont reliés par des arrêtes, et ont une même couleur d'attribuée.

III.1.1.1. Règles pour l'analyse des contraintes

Pour analyser un système de contraintes, on doit établir des règles générales qui permettront de travailler sur tous les types de CSP.

Lorsque on a des contraintes, on a des valeurs qui peuvent varier pour satisfaire ses mêmes contraintes, tel que ce sont des variables, qu'on appellera V l'ensemble de ces variables. Une variable par définition, peut prendre plusieurs valeurs, on appellera D, l'ensemble de ces valeurs, c'est le domaine de définition.

Exemple :

Pour la fonction "racine carré" de x, V est constitué de la variable x uniquement, et le domaine D correspond à l'ensemble des réels positifs.

Ainsi, pour le problème des couleurs expliqué ci avant, V correspond à l'ensemble des sommets, et un sommet dont les voisins n'ont pas encore de couleurs, peut prendre une de ces trois couleurs du domaine (D(s) = {Rouge, Bleu, Vert}. Dans ce problème, les affectations de couleurs aux voisins du sommet s réduisent le domaine D(s).

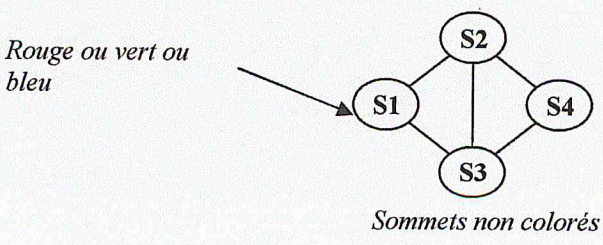


Figure III-2 : Exemple de présentation du domaine de s.

Cette réduction est due à la contrainte principale du problème, cette contrainte s'exprime par la liaison qui existe entre deux sommets, c'est une relation entre deux variables. Nous appellerons R, l'ensemble des différents types de relations.

Dans le problème de la coloration de graphe, il y a un seul type de relation que nous appellerons R1. R1 correspond à un doublet de sommets qui *ne peuvent avoir la même couleur*. R1 est une relation binaire car elle ne prend que deux sommets en attributs. Mais une relation peut posséder plus de variables.

Pour appliquer ces relations, il faut repérer l'ensemble des cas où nous utilisons ces relations. Cela correspond à l'ensemble des contraintes que nous appellerons C.

Imaginons que, dans notre problème de la coloration de graphe, nous ayons quatre sommets S1, S2, S3, S4. S1 est relié à S2 et S4, S2 à S1 et S4, S3 à S4 et enfin S4 à S1, S2 et S3.

Nous écrirons C ainsi : C= {R1 (S1, S2); R1 (S1, S4); R1 (S2, S4); R1 (S3, S4)}, chaque liaison n'étant exprimée qu'une fois : R1 (S1, S2)=R1 (S2, S1). Nous avons vu que l'application de l'ensemble des contraintes pouvait influencer l'ensemble D.

Nous avons donc besoin, pour travailler sur un CSP, des ensembles V, D, R et C.

Ce quadruplet correspond à un système de contraintes. Nous appellerons ce quadruplet P.

Remarques :

Ce système de notation qui est utilisé par les informaticiens travaillant sur les CSP est très proche du système de notation des mathématiciens.

III.1.1.2. Algorithme résolvant un CSP

On peut, dans un premier temps, définir la notion d'algorithme (ou procédure) du dictionnaire, c'est une :"Suite de raisonnements ou d'opérations qui fournit la solution de certains problèmes.". En informatique, c'est une suite d'instructions qui permet à l'ordinateur d'effectuer un calcul complexe à partir d'opérations de base.

Le but de notre algorithme est de trouver une solution correcte à un CSP. Pour cela, il doit vérifier que les variables issues de V prennent des valeurs qui appartiennent bien au domaine D des variables. Ces mêmes valeurs doivent satisfaire les contraintes contenues dans C.

La procédure doit donc vérifier que la solution corresponde au système de contraintes P. Mais la principale fonctionnalité de notre algorithme est de générer des solutions qui vérifient le système P (soit toutes les solutions possibles, soit la meilleure solution

possible, soit la première qui se présente). L'algorithme doit affecter des valeurs aux variables tout en vérifiant que les valeurs proposées ne s'opposent pas aux contraintes.

On a, pour cela, plusieurs types d'algorithmes. Certains fonctionnent de manière systématique en testant toutes les solutions possibles jusqu'à ce qu'il trouve une bonne solution. D'autre, plus intuitifs, construisent la réponse à partir d'une solution non satisfaisante quelconque en la modifiant selon les contraintes.

Exemple :

Avant d'appliquer un algorithme à un CSP, nous devrons définir à quoi correspond V, D, R et C. Ensuite, il faudra trouver l'algorithme le plus adapté au problème. Mais beaucoup de CSP ne peut être résolus par des procédures spécifiques. Il faut utiliser des algorithmes généraux applicables à n'importe quel CSP.

Nous allons expliquer ce qu'est un algorithme résolvant des CSP avant d'étudier quelques algorithmes connus.

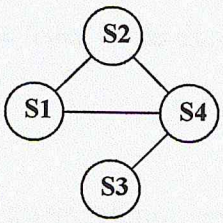


Figure III-3 : Exemple d'un système de contrainte.

Nous avons donc au début :

P= {V, D, R, C}, V= {S1, S2, S3, S4}, R= {R1}

C= {R1 (S1, S2); R1 (S1, S4); R1 (S2, S4); R1 (S3, S4)}

D (S1)=D (S2)=D (S3)=D (S4)= {Rouge, Vert, Bleu}

Puis au fur et à mesure de l'avancer du calcul de l'algorithme, chacun de ces ensembles D de valeurs se réduira en fonction des affectations de couleurs des sommets.

Pour notre part, nous traitons de l'information temporelle. Ce qui nous ramène à nous intéresser aux CSP spécifiques à ce type d'information.

III.1.2. TCSP : Les CSP temporels

Un problème de satisfaction de contraintes temporelles est un CSP dont les variables prennent des dimensions temporelles (instants ou intervalles). Il y a deux types de gestion des contraintes temporelles :

Une contrainte $C_{i,j}$ entre X_i et X_j, est une disjonction de différences bornées du type :

$$a^1_{i,j} \leq X_i - X_j \leq b^1_{i,j} \vee \ldots \vee a^n_{i,j} \leq X_i - X_j \leq b^n_{i,j}$$

Par conséquent, une contrainte $C_{i,j}$ définit un ensemble d'intervalles de validité :

$$\{I^1_{i,j}, \ldots, I^n_{i,j}\} \text{ où } I^k_{i,j} = [a^k_{i,j}, b^k_{i,j}]$$

Un élément multimédia peut donc être modélisé dans ce formalisme par une union de domaines de validité.

III.1.2.2. Les STP (*Simple Temporel Problem*) [DEC 91]

Le seul cas de TCSP qui peut être résolu en temps polynomial est celui d'un TCSP ne comportant qu'un **intervalle** par **contrainte**. Un tel TCSP est appelé **STP**[1]. Une contrainte $C_{i,j}$ entre deux variables X_i et X_j d'un **STP** est représentée par une différence bornée :

$$a_{ij} \leq X_i - X_j \leq b_{ij}$$

ou par une paire d'inégalités :

$$X_i - X_j \leq b_{ij}$$
$$X_i - X_j \leq -a_{ij}$$

Le cadre des STP permet à cette classe de problème d'être abordé à travers plusieurs approches. Tout d'abord, trouver une solution dan un STP peut être considéré comme résoudre un système d'inéquations linéaires. C'est un problème bien connu dans le domaine de la recherche opérationnelle qui peut être résolu par la méthode du simplex [DAN 62]. Cette résolution est réalisée en temps exponentiel.

La classe des inéquations linéaires qui caractérise les STP a la particularité d'admettre des solutions plus efficaces que celle du simplexe : les inéquations représentant les contraintes temporelles sont structurées sous forme d'un graphe auquel un algorithmique de graphe peut être appliqué. Cette approche par la théorie des graphes est la plus naturelle, elle est moins coûteuse et permet de bénéficier de nombreuses propriétés.

Exemple :

Expression du scénario temporel sous forme de contraintes :

Dans le cas d'un scénario temporel exprimé sous forme de contraintes, chaque objet est défini par deux variables (*ses instants de début et de fin*), le domaine de ses variables est [0, +∞ [. Chaque objet flexible introduit une contrainte correspondant à son intervalle de durées possibles :

- *Borne-Inf* ≤ Fin - Début ≤ *Borne-Sup*

[1] Simple Temporel Problem

Et chaque relation du scénario est traduite en terme de contraintes qui sont toutes sous la forme d'une différence de la forme :

- min <= X - X' <= max

Tel que *min* et *max* pouvant prendre leurs valeurs dans l'intervalle [0, +∞ [, et où X et X' sont des instants de début ou de fin.

Par exemple la relation *dur* entre les objets 1 et 2 sera traduite par :

- $1 \leq \text{Fin2} - \textit{Fin1} \leq +\infty$
- $1 \leq \text{Début1} - \textit{Début2} \leq +\infty$

Définition et théorèmes :

Formellement, à chaque STP est associé un graphe étiqueté orienté appelé *graphe de distance* :

Définition d'un graphe de distance temporelle :

Un graphe de distance temporelle $G_d = (V, E_d)$ est défini par un ensemble de sommets 0, .., n représentant des variables $V = \{X_0, ..., X_n\}$ et un ensemble d'arcs E_d, où chaque arc $X_i \rightarrow X_j$ relie deux sommets distincts X_i et X_j du graphe. Chaque arc est étiqueté par une valeur entière. Si $[a_{i,j},b_{i,j}]$ est l'intervalle de validité associé à la distance entre X_i et X_j, alors l'arc $X_i \rightarrow X_j$ est étiqueté par $b_{i,j}$ et l'arc $X_j \rightarrow X_i$ est étiqueté par $-a_{i,j}$.

Chaque chemin d'un sommet *i* à un sommet *j* dans le graphe G_d, $i = i_0, ..., i_k = j$, introduit la contrainte temporelle suivante sur la distance $X_j - X_i$: $X_i - X_j \leq \sum a_{ij-1, ij}$

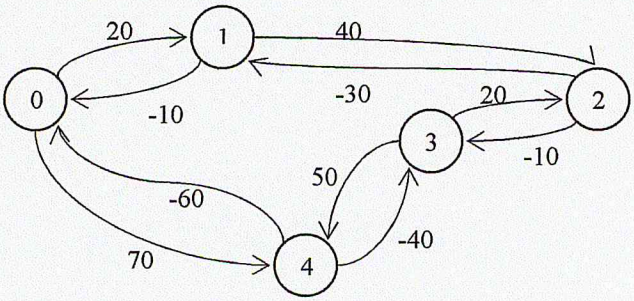


Figure III.4 : Graphe de distances d'un STP.

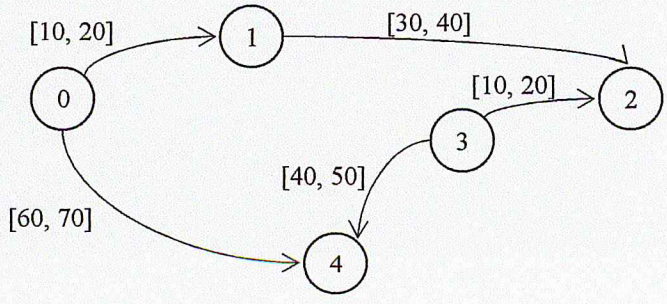


Figure III.5 : Graphe de contraintes.

- La gestion d'un scénario à partir de relations *qualitatives* ou *symboliques* (TCSP symboliques).
- La gestion d'un scénario à partir de relations *quantitatives* ou *numériques* (TCSP numériques).

Les relations *symboliques* sont en général plutôt associées aux relations à base *d'intervalle*. Mais les relations à base *d'instants* peuvent aussi exprimer des contraintes symboliques : $t_1 < t_2$ en est un exemple (t_1 et t_2 sont deux variables d'instant). Les relations symboliques permettent d'exprimer des contraintes donnant un positionnement relatif entre deux intervalles ou deux instants.

Exemple :

Nous avons deux scénarios A et B.

"A se termine après B" est une relation symbolique.

Les relations *quantitatives* introduisent des valeurs numériques et permettent d'exprimer des contraintes temporelles à base de dates ou de durées.

Exemple :

A et B deux scénarios.

"A se termine 5 secondes après B" est une relation quantitative.

Ces deux catégories de relations sont toutes deux très utiles et complémentaires : une contrainte symbolique pourra avoir comme effet de définir implicitement une contrainte numérique et inversement.

Les relations symboliques permettent de définir des contraintes souples, alors que les relations numériques sont plus précises.

Nous allons dans la suite définir les TCSP numériques qui sont des systèmes numériques manipulant explicitement des nombres, ce qui leur permet de couvrir l'aspect métrique du temps, puis nous allons parler des STP, et cela correspond à la modélisation de chaque élément multimédia par un seul domaine de validité.

III.1.2.1. TCSP numériques [DEC 91]

Les contraintes temporelles liées aux domaines de validité et aux relations temporelles symboliques peuvent s'exprimer sous forme de différences bornées entre dates d'instants. D'où la possibilité de représenter les problèmes temporels sous forme de TCSP numériques.

Un TCSP numérique est défini par un ensemble de variables X_1, .., X_n représentant des instants ayant des valeurs de dates, et un ensemble de contraintes binaires.

S'il existe plus d'un chemin de i à j, alors il est facile de vérifier que l'intersection de toutes les contraintes de chemins donne : $X_j - X_i \leq d_{ij}$[2]

En se basant sur cette observation, la condition fondamentale suivante pour la cohérence d'un problème STP peut être établie :

Théorème 1 :

Cohérence d'un STP : Shostak [*SHO-81*], Liao et Wong [*LIA-93*], Leiserson et Saxe [*LEI-93*]

Un STP est cohérent si et seulement si son graphe de distance Gd ne contient pas de cycle négatifs.

Ce premier théorème important pour les STP, concerne la vérification de la cohérence. Il exprime la possibilité de vérifier la cohérence du STP en appliquant des algorithmes de recherche du plus court chemin.

Corollaire :

Soit G_d le graphe de distance d'un STP cohérent, Il existe aux moins deux solutions cohérentes qui sont : $S_1 = (d_{01}, ..., d_{0n})$ et $S_2 = (-d_{10}, ..., -d_{n0})$

Ces deux solutions représentent une instanciation des variables à leur date d'occurrence au plus tôt et au plus tard respectivement.

Deux problèmes STP sont dits *équivalents* s'ils représentent le même ensemble de solutions et un STP est dit *minimal* s'il n'existe aucun STP équivalent dont les contraintes soient plus restrictives.

- **Algorithmes classiques associés aux STP :**

De nombreux algorithmes de coût polynomial ont été décrits autour des STP. Certains proviennent de la théorie des graphes : les algorithmes de plus court chemin, d'autre s'appuient sur la notion de cohérence locale des CSP, en général l'arc-cohérence ou le chemin-cohérence. Ils servent à obtenir un graphe minimal, à calculer la cohérence d'un réseau ou encore à calculer une solution.

L'un des algorithmes les plus connus de la famille *plus court chemin* est celui de ***Floyd-Warshal***. Il permet de construire le graphe de distance d'un STP. Cet algorithme peut être considéré comme un algorithme de relaxation : à chaque pas la valeur d'un arc est mise à jour d'un montant qui dépend uniquement de celles des arcs adjacents. L'algorithme permet de trouver une solution. Il détecte les cycles négatifs (*garant de la cohérence d'un STP*) en examinant le signe des éléments diagonaux d_{ii} de la diagonale de la matrice qui représente le graphe de distances.

[2] dij représente la durée du plus court chemin entre i et j.

Algorithme du plus court chemin (Floyd-Warshall) [*PAP-82*] :

1. Pour i := 1 à n faire $d_{ii} \leftarrow 0$
2. Pour i, j := 1 à n faire $d_{ij} \leftarrow a_{ij}$
3. Pour k := 1 à n faire
4. Pour i, j := 1 à n faire
5. $d_{ij} \leftarrow \text{Min}\ \{d_{ij}, d_{ik} + d_{kj}\}$

Résoudre un CSP, c'est-à-dire savoir s'il y a des solutions, puis en trouver une. On a vu qu'un grand nombre de méthodes ont été développées pour essayer de maîtriser le coût de cette résolution. Elles procèdent un certain niveau de cohérence au CSP.

Dans ce chapitre, on a mieux approché et étudié les **STP,** où on est arrivé à placer des simplifications.

Malgré ces simplifications, ces cadres restent relativement riches et disposent d'algorithmes nombreux.

A ce titre, la classe des STP semble être un bon compromis entre efficacité et pouvoir de représentation.

III.2. Approche basée sur la description formelle RT-LOTOS [COU 03]

III.2.1. Présentation du langage LOTOS

LOTOS est une abréviation de (Language Of Temporal Ordering Specification). C'est une technique de description formelle, approuvée au rang de standard ISO [ISO 89] en 1988, et utilisée pour la spécification et la vérification des systèmes répartis et des protocoles de communication. Ce langage a été conçu par l'union de deux formalismes complémentaires : ACT-ONE pour la partie donnée, et CCS/CSP pour la partie contrôle (ou comportementale) [SAM 03].

Il existe plusieurs extensions temporelles de LOTOS telles que LOTOS-T, Timed LOTOS, ET-LOTOS, et E-LOTOS.

RT-LOTOS (Real-Time LOTOS [COU 00]) est une autre extension de LOTOS, qui s'est inspirée de Timed LOTOS ainsi de son successeur ET-LOTOS. Ces langages comptent plusieurs similitudes au niveau sémantique vis-à-vis du traitement des aspects temporels.

En LOTOS, un système est considéré comme un processus global composé (en général) de plusieurs sous processus qui communiquent à travers des portes de communication selon

un mécanisme sophistiqué de synchronisation. Ces sous processus sont également des processus LOTOS qui peuvent être encore décomposés successivement en d'autres sous processus Ainsi, une spécification LOTOS décrit un système par une hiérarchie de définitions de processus.

L'un des points fondamentaux de LOTOS demeure dans son mécanisme de synchronisation. Toutes les communications entre processus sont effectuées par la synchronisation de leurs actions.

III.2.2. Présentation de RT-LOTOS

RT-LOTOS propose particulièrement trois opérateurs pour décrire, de manière intuitive, l'expression du temps dans le comportement de processus LOTOS :

- L'opérateur **delay (d)** permet de retarder un processus d'une certaine quantité de temps d.
- L'opérateur **a** $\{t\}$ est un opérateur de restriction temporelle qui limite le temps pendant lequel une action observable « a » peut être offerte à son environnement.
- L'opérateur **latency (l)** qui est le plus important opérateur proposé par LOTOS et qui est particulièrement adapté pour exprimer de manière générale le non-déterminisme temporel. C'est un opérateur de latence, qui détermine une action interne peut être offerte de manière non déterministe au sein de l'intervalle de latence ''l'' qui peut lui être associé.

En RT-LOTOS, les actions observables ne sont pas urgentes et leurs occurrences dépendent donc de l'environnement. En conséquence, si une action g $\{t\}$ ne peut pas se produire pendant sa période de validité temporelle (c'est à dire avant l'échéance de t), le processus qui attend cette action se transforme en stop, en absence d'autre alternative. Afin de rester plus proche possible du standard LOTOS, la partie de donnée de RT-LOTOS s'appuie toujours sur le formalisme ACT-ONE.

Cependant étant donné la complexité de ce formalisme, en RT-LOTOS seule les déclarations des variables (la signature des types abstraits) sont effectuées en utilisant la syntaxe de ACT-ONE. La partie correspondant à la sémantique des types de données est, quant à elle, implémentée en utilisant les langages C++ et Java.

La définition de la sémantique formelle de RT-LOTOS a été définie dans [COU 00].

Cette sémantique permet de prendre en compte des comportements contraints par le temps, et donc de décrire des systèmes intégrants des contraintes temporelles.

L'implémentation de cette sémantique dans l'outil RTL (Real-Time LOTOS Laboratory) permet la simulation et la vérification de tels systèmes.

III.2.3. L'environnement RT-LOTOS

Dans cette section on va présenter l'outil RTL (Real-Time LOTOS Laboratory) qui est un des principaux responsables du succès du langage RT-LOTOS pour la modélisation et la vérification formelle des systèmes répartis contraints par le temps. RTL est écrit en C++ , les types de données de base(naturels , entiers , logiques et temps) sont incorporés au code principal, tandis que d'autres types doivent êtres incorporés au code au moyen de librairie statique ou dynamique par l'utilisateur, en utilisant le langage C++ ou Java.

La preuve de propriétés d'un système peut être faite par l'analyse de la représentation du comportement global de la spécification du système via un système de transition étiqueté dérivé de la spécification RT-LOTOS. Cet outil permet la validation des systèmes modélisés en RT-LOTOS en utilisant deux techniques de base :

- ***La simulation*** **:** Cette approche consiste à produire et à analyser des résultats de la simulation associés à la spécification globale du système.
- ***La vérification*** **:** Cette approche permet à l'utilisateur de prouver qu'une propriété déterminée peut être vérifiée sur la spécification du système.

Le principal avantage de l'approche de simulation est qu'elle permet l'exécution rapide des différents scénarios puisqu'il n'est pas nécessaire de comparer un état accessible à tous les états précédents. Cela facilite le « débogage » de spécifications complexes (par exemple, pour détecter des situations de blocage ou des comportements non déterministes créés de façon erronée par l'utilisateur).

Mais malgré ses avantages, la simulation ne peut être considérée comme une preuve formelle, car il n'existe aucune garantie que l'espace d'états ait été complètement couvert.

L'approche de vérification permet l'analyse complète du modèle d'un système spécifié en RT-LOTOS. Le but consiste à décrire et analyser le comportement global du système au moyen du graphe minimal d'accessibilité. Le graphe d'accessibilité peut être obtenu directement à partir des règles de la sémantique opérationnelle du langage utilisé ou indirectement à partir d'un modèle intermédiaire dans lequel le langage est traduit. Des nouvelles fonctionnalités ont également été ajoutées à RTL permettant d'engendrer un modèle (automate temporel) pour l'ordonnancement en s'appuyant sur les résultats de la vérification d'un système.

Les principaux avantages de cette approche est qu'elle permet la vérification de la cohérence temporelle des documents multimédias et la définition de la flexibilité du scénario temporel de ce document permettant la gestion du non-déterminisme du document (*en temps de compilation*) et l'ordonnancement de ce document adoptant une approche prédictive pour le pré-chargement en mémoire des objets média nécessaires pour la présentation (*en temps d'exécution*).

Conclusion

Le travail effectué pour l'analyse des documents multimédias nous a guidé comme prévue vers l'aspect temporel. Or c'est le point important de notre travail, comme on a vu les différents modèles de présentation spatial et temporel, puis les méthodes de vérification de la cohérence pour résoudre les différents problèmes liés à l'aspect temporel et plus précisément les STP. Malgré les recherches effectuées dans ce sens, il reste un vaste chantier, car toutes les solutions proposées jusqu'à présent souffrent d'un handicap majeur qui est le temps de calcul, ce qui limite leur utilisation dans un environnement interactif.

Chapitre IV

CONCEPTION DE NOTRE MODÈLE

Chapitre IV

CONCEPTION DE NOTRE MODÈLE

Lors de l'étude effectuée dans le chapitre II relatif au langage SMIL1.0, nous avons remarqué que les codes SMIL peuvent comporter quelques problèmes ou plus exactement incohérences de types temporels, ensuite nous avons vu que les relations temporelles peuvent être modélisées sous forme de contraintes.

Nous arrivons ainsi au cœur de notre travail qui consiste à étudier la cohérence des documents SMIL. Nous commencerons ce chapitre par quelques définitions concernant la cohérence. Nous présenterons ensuite notre modèle.

IV.1. La vérification de cohérence

IV.1.1. Définition [ENC 02]

Une présentation d'un document multimédia est cohérente si l'action caractérisant le début de la présentation du document est forcement suivie, après un certain temps fini, par une action caractérisant la fin de la présentation du document [COU 96].

La complexité des documents multimédias est telle que l'auteur ne peut pas appréhender toutes les répercussions possibles d'une modification qu'il a effectué. Le système doit donc être capable de vérifier en temps réel la cohérence d'un document. A chaque modification[1] du document, le système d'édition doit vérifier la cohérence du document et calculer une nouvelle solution. Quand une incohérence est détectée, le système d'édition doit informer l'auteur de l'erreur, et lui donner un diagnostic aussi complet que possible sur l'erreur de manière à ce que celui-ci puisse rendre le document de nouveau cohérent.

Comme il a été dit précédemment, les codes SMIL peuvent comporter des incohérences de type temporel. Dans ce qui suit, nous allons dresser la liste des types d'incohérences temporelles que peuvent comporter les codes SMIL.

IV.2. Les types d'incohérence

1. ***Incohérence qualitative :*** une incohérence qualitative est détectée dans les cas suivants :
   - Si l'auteur met des relations incompatibles entre elles, quelle que soit la valeur de chaque attribut défini sur les objets.
   - Si l'auteur définit une relation avec un objet qui n'existe pas.
   - Si l'auteur définit des liens temporels incohérents.

   Exemple : un lien pointe vers la 10ème seconde d'un document qui ne dure que 7 secondes, ainsi que la détection des objets qui ne seront jamais joués. Par exemple, un objet qui commence à la 10ème seconde d'un objet composite qui ne dure que 20 secondes.

2. ***Incohérence quantitative :*** Une incohérence quantitative peut avoir lieu lorsque l'auteur insère une nouvelle relation dans le document sachant que ce dernier est qualitativement cohérent, mais une valeur sur un attribut fait que le document soit incohérent. Ce type d'incohérence est lié aux contraintes numériques.

[1] Intégration d'un nouvel objet ou changement de contrainte.

Exemple :

C'est par exemple lors de la spécification d'une valeur de début supérieur à la valeur de fin d'un *Clip* média, on se trouve dans une situation d'incohérence quantitative car si comme si on demande l'exécution de la fin avant le début du *Clip*.

3. ***Incohérence causale :*** Dans cette partie, l'incohérence concerne les relations causales. Une incohérence causale peut être provoquée à cause des contraintes qui expriment des relations de synchronisation entre deux fils d'un par.

   Exemple :

   Dans l'exemple, nous avons l'élément A attend le début de B, et l'élément B attend le début de A, comme suit :

   begin (A) = begin (B)

   begin (B) = begin (A)

4. ***Incohérence événementielle :*** Il s'agit de la réalisation des actions associées à l'événement. C'est-à-dire que les objets impliqués dans les actions doivent être actifs[2]. Dans le cas particulier des documents prédictifs, on se ramène à vérifier les incohérences quantitative et qualitative. Il faut par exemple vérifier que les objets qui sont la destination des événements soient présents au moment ou l'objet source déclanchera l'événement.

   Exemple :


```
<a href="http://www.archive.com/test1.smi"/> // lien n'existe pas.
      <video src="rtsp://foot.com" region="fenêtre2"/>
</a>
```

Figure IV.1 : Exemple d'incohérence événementielle.

[2] Ils doivent être en en cours de présentation au moment de l'activation du lien.

IV.3. Conception de notre modèle

Nous avons vu précédemment que le langage SMIL permet la spécification des besoins d'édition et des contraintes temporelles de synchronisation. Donc, l'utilisation des objets dynamiques et la définition des relations temporelles sur ces objets, sont facilement employées par l'utilisateur. Nous avons vu aussi que ces relations forment un système de contraintes de synchronisation qui doivent être satisfaites. Il est à noter qu'une situation d'incohérence peut avoir lieu car se sont des erreurs sémantiques que le langage SMIL ne détecte pas. Ainsi, afin d'analyser un document SMIL, nous allons le traduire en un graphe temporel.

IV.4. Modélisation sous forme de graphe temporel

Pour la construction d'un graphe temporel, nous allons donner quelques définitions.

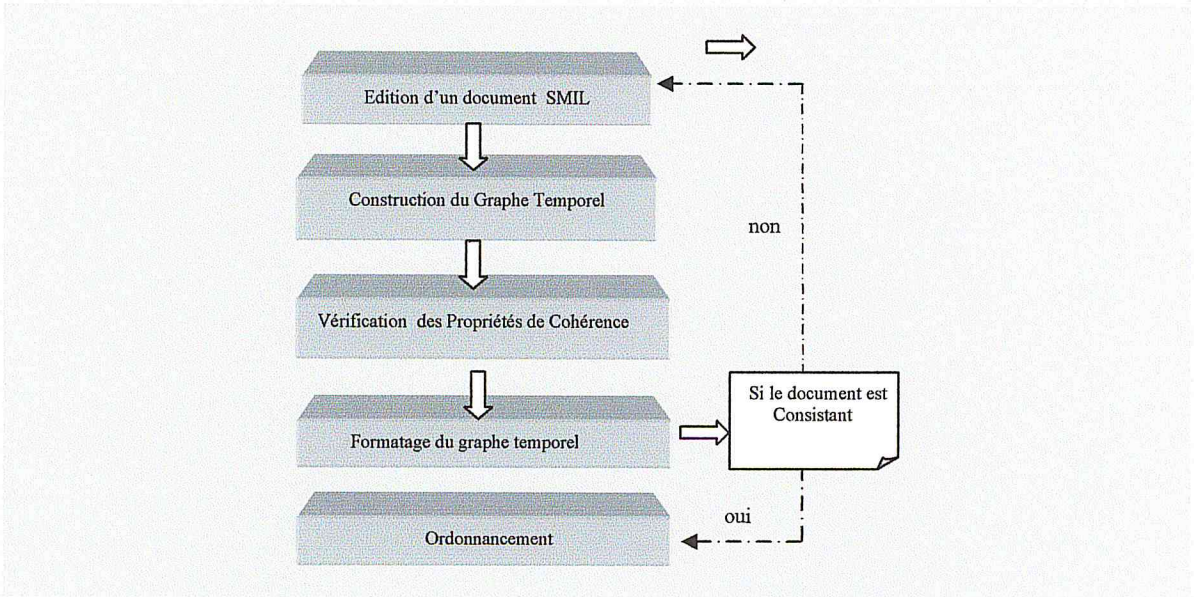


Figure IV.2 : Étapes de contrôle de documents SMIL.

Définition 1: (début et fin d'un objet multimédia)

Un objet multimédia *X* sera représenté par ses événements *begin X* et *end X* qui représentent respectivement le début et la fin de l'objet *X*.

Définition 2: (les médias)

Un média peut être de base ou composite tel que:

- un média de base est un objet multimédia comme une audio, une vidéo, une image,...
- un média composite est composé de média de bases qui sont liés par des opérateurs *par* et *seq* .Ainsi un média composite séquentiel commence par la balise <seq>, et un média composite parallèle commence par la balise <par>.

Définition 3: (graphe temporel)

Un graphe temporel est un couple G= (X, U) où

- **X**, est l'ensemble de nœuds. Chaque nœud représente un événement et caractérise l'instant de début et de fin d'un objet multimédia, élément composite.
- **U**, est l'ensemble fini et non vide d'arcs orientés et étiquetés par une valeur numérique, $U = \{(X_1, X_2, d)$ tel que $X_1 \in X$ et $X_2 \in X$ et $d \in N \cup \{\perp\}\}$, avec N est l'ensemble des entiers naturels et $\perp$ représentant une valeur inconnue.

Un triplet $(X_1, X_2, d) \in U$, avec $d \in N$, signifie que l'événement X_2 a lieu après ou avant l'événement X_1 de d unités de temps, selon que d soit positif ou négatif. Dans le cas où d est nul, l'événement X_2 a lieu immédiatement après X_1. Si $(X_1, X_2, \perp) \in U$ alors X_2 a lieu soit avant soit après X_1 mais la durée qui sépare les deux événements est inconnue.

Notation:

Nous utiliserons pour le début et la fin d'un objet média les notations suivantes:

- x^- et x^+ représentent le début et la fin d'un média de base X.
- seq^- et seq^+ représentent le début et la fin d'un média composite séquentiel.
- par^- et par^+ représentent le début et la fin d'un média composite parallèle.
- Nous introduisons un point important dans la représentation graphique d'un STP : c'est la notion d'un arc étiqueté d'un temps ε infiniment petit séparant deux événements, correspondant au cas où $d = 0$. Cela signifie que l'événement X_j est déclenché immédiatement après X_i.

IV.5. Traduction du code SMIL en graphe temporel

Cette traduction s'effectue à l'aide du modèle *STP* vue dans le chapitre III, ainsi le code SMIL sera modélisé sous les contraintes du modèle, or se dernier peut être traduit en un graphe temporel orienté.

IV.5.1. Représentation graphique des éléments simples

La traduction d'un média de base, noté *x*, est définie comme suit :

1. Insertion des nœuds schématisant les événements *début* et *fin* de x :

$X = X \cup \{x^-, x^+\}$

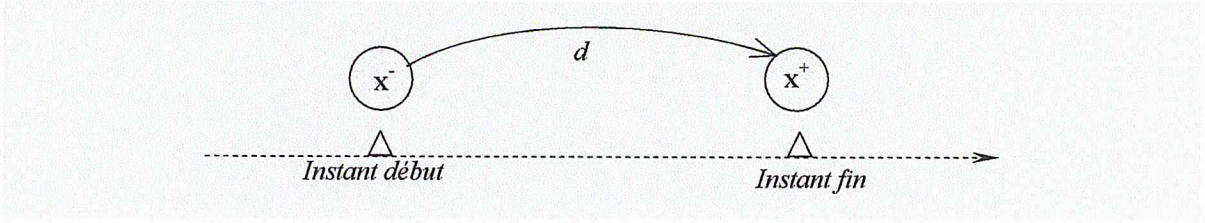


Figure IV.3 : représentation graphique d'un élément simple.

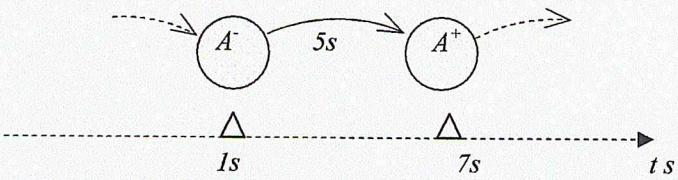
En outre, un graphe temporel doit posséder des couples composés d'un seul noeud source et un seul noeud destination. Chaque nœud représente respectivement les événements de début et de fin du scénario complet modélisé. Il est possible de fixer $x^- = 0s$, et l'instant x^+ représentera ainsi la durée totale de présentation du scénario.

2. Etiqueter l'arc menant de x^- à x^+ de la manière suivante :

2.1. Si *d* est la valeur spécifiée par l'attribut *dur* ou calculée à partir des valeurs spécifiées par les attributs *begin* et *end (d= end – begin)*; et si les trois attributs existent, *d* sera le minimum entre les valeurs de *dur* et de (*end – begin*)

Alors $U = U \cup \{(x^-, x^+, d)\}$

Exemple :



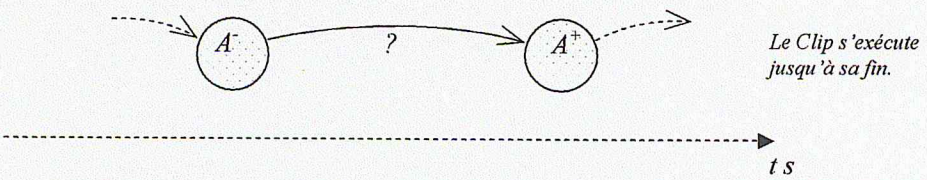

```
...
<par>
<audio id="A" src="..." begin="1s" dur="5s" end="7s" />  <!--Clip-1-->
</par>
...
```

Figure IV.4 : Choix de durée minimal.

2.2. Si aucun paramètre dans la définition de X ne permet de calculer *d*

Alors $U = U \cup \{(x^-, x^+, \perp)\}$

Exemple :




```
...
<par>
<video id="A" src="..." />   <!--Clip-1-->
</par>
...
```

Figure IV.5 : Durée de présentation inconnue.

IV.5.2. Représentation graphique des média composites : Seq et Par

a. Média Composite Séquentiel

La traduction d'un média composite séquentiel, noté Seq_p[3] et correspondant au code SMIL "<*seq*> x_1 ... x_n </*seq*>", où les x_i sont des média de base, définie comme suit :

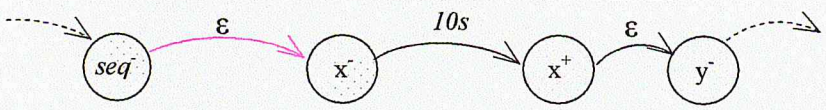
1. Insertion des nœuds schématisant les événements *début* et *fin* de Seq_p :

$X = X \cup \{Seq_p^-, Seq_p^+\}$

2. Etiqueter l'arc menant de Seq_p^-, Seq_p^+ de la manière suivante :

2.1. Si le début de x_1 n'est pas donné, alors le début de Seq_p se confond avec celui de x_1

$U = U \cup \{(Seq_p^-, x_1^-, 0)\}$

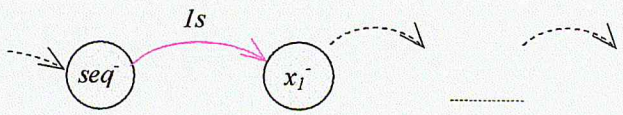



```
...
<seq>
<video id="x" src="..." dur="10s" />    <!--Clip-1-->
<video id="y" src="..." />              <!--Clip-2-->
</seq>
...
```

Figure IV.6 : Exemple-1 de relation du ***seq⁻*** *avec son premier successeur.*

2.2. Si le début de x_1 est spécifié par la valeur *d*, alors x_1 a lieu *d* unités de temps après le début de Seq_p

$U = U \cup \{(Seq_p^-, x_1^-, d)\}$

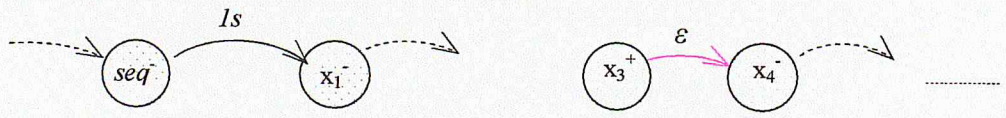



```
...
<seq>
<video id="x1" src="..." begin="2s" />   <!--Clip-1-->
...
</seq>
...
```

Figure IV.7 : Exemple-2 de relation du ***seq⁻*** *avec son premier successeur.*

[3] "p" correspond au numéro du composite en cours de traduction.

2.3. Si le début de x_i, pour i >1, n'est pas donné, alors le début de x_i se confond avec la fin de x_{i-1} $U = U \cup \{(x_{i-1}^+, x_i^-, 0)\}$

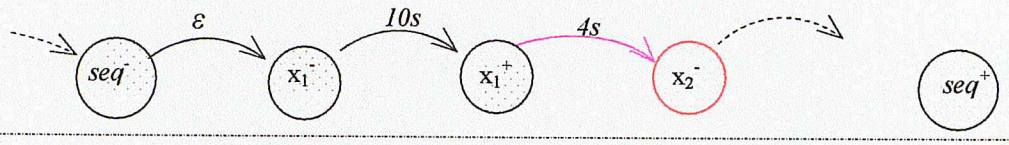



```
...
<seq>
<img id="x1" src="..." begin="1s" dur="4s" />     <!--Clip-1-->
...
<img id="x3" src="..." dur="4s" />                <!--Clip-3-->
<img id="x4" src="..." dur="8s" =/>               <!--Clip-4-->
</seq>
...
```

Figure IV.8 : Exemple-1 de relation entre fils d'un composite seq.

2.4. Si le début de x_i , pour i >1, est spécifié par la valeur *d*, alors x_i commence *d* unités de temps après la fin de x_{i-1}

$U = U \cup \{(x_{i-1}^+, x_i^-, d)\}$




```
...
<seq>
<img id="x1" src="..." dur="10s" />               <!--Clip-1-->
<img id="x2" src="..." begin="4s" dur="4s" />     <!--Clip-3-->
...
</seq>
...
```

Figure IV.9: Exemple-2 de relation entre fils d'un composite ***seq***.

2.5. La fin du composite Seq_p se confond avec la fin de son dernier fils, soit e_n

$D = D \cup \{(e_n^+, Seq_p^+, 0)\}$


```
...
<seq>
<img id="x1" src="..."  ...  dur="10s" />         <!--Clip-1-->
...
<img id="x2" src="..." begin="4s" end="8s" />     <!--Clip-2-->
</seq>
...
```

Figure IV10: Exemple de relation du ***seq***$^+$ *avec son dernier fils.*

b. Média Composite Parallèle

La traduction d'un média composite parallèle noté Par_p[4] et correspondant au code SMIL "*<par> x_1 ... x_n </par>*", où les x_i sont des média de base, définie comme suit :

1. Insertion des nœuds schématisant les événements *début* et *fin* de Par_p :

$X = X \cup \{Par_p^-, Par_p^+\}$

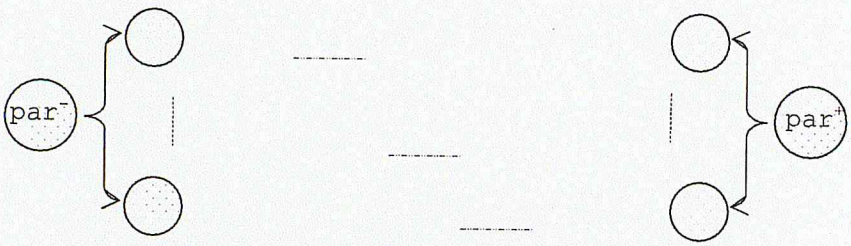
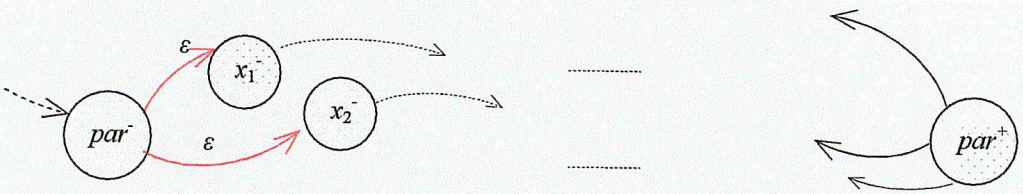


Figure IV.11 : Représentation graphique d'un composite ***par.***

2. Etiqueter les arcs menant de Par_p^-, Par_p^+ de la manière suivante :

2.1. Pour tout x_i, $i \in \{1... n\}$, dont le début n'est pas donné, alors le début de Par_p se confond avec celui de x_i :

$U = U \cup \{(Par_p^-, x_i^-, 0)\}$



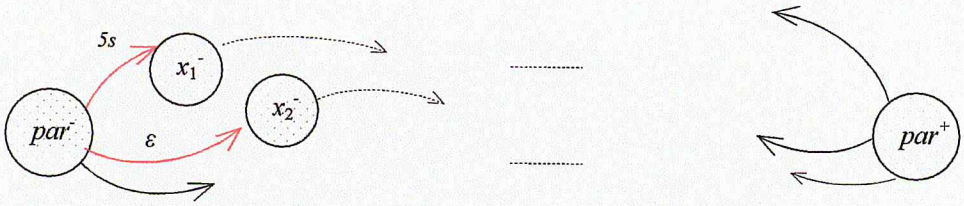

```
...
<par>
<audio id="x1" src="..." dur="13s"/>    <!--Clip-1-->
<img id="x2" src="..." dur="13s"/>      <!--Clip-2-->
</par>
...
```

Figure IV.12 : Exemple-1 de relation entre par- et x-.

[4] "p" correspond au numéro du composite en cours de traduction.

2.2. Pour tout x_i , $i \in \{1,\ldots,n\}$, dont le début est spécifié par la valeur *d*, cela signifie que x_i aura lieu *d* unités temps après le début de Par_p :

$U = U \cup \{(Par_p^-, x_i^-, d)\}$

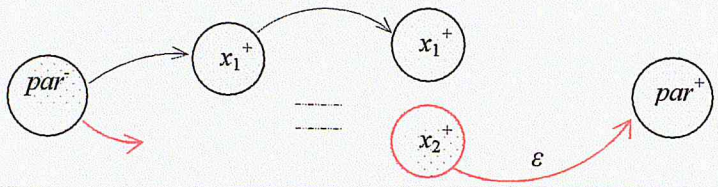



```
...
<par>
<img id="x1" src="..." begin="5s" dur="20s"/>   <!--Clip-1-->
<img id="x2" src="..." dur="25s"/>               <!--Clip-2-->
...
</par>
...
```

Figure IV.13 : Exemple-2 de relation entre par^- et x^-.

2.3. Pour tout x_i, $i \in \{1,\ldots,n\}$, la fin du composite Par_p se confond avec la fin de tous ses fils x_i :

$U = U \cup \{(x_i^-, Par_p^+, 0)\}$




```
...
<par>
...
<ref id="x1" src="..." begin="5s" dur="8s" />          <!--Clip-1-->
...
<img id="x2" src="..." dur="25s" />                    <!--Clip-2-->
</par>
...
```

Figure IV.14 : Exemple de relation entre par^+ et X^+.

IV.6. Détection d'incohérences

Deux cas d'incohérences peuvent être détectées dans un document SMIL, qualitative et causale, qu'ils auront lieu dans les situations suivantes :

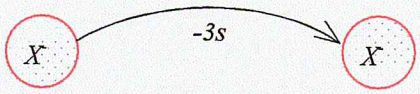
1. Si l'attribut *dur*, possède une valeur négative de manière directe ou non[5], dans ce cas le graphe possède un arc négatif.
2. Si le graphe temporel possède au moins un cycle.

Remarque :

Le cas d'incohérence qualitative n'aura pas lieu dans un document SMIL, puisque ce dernier n'utilisent pas des relation qualitatives comme ***meet, finish***..., dans la spécification de son code SMIL, et les incohérences évènementielle eux aussi ne sont pas traité dans notre modèle puisque c'est impossible à notre égard de vérifier l'existence d'un lien hypermédia, donc on se trouve dans deux cas distinct.

Exemple :

Cet exemple sera impossible a joué puisque le poids de l'arc est négatif.



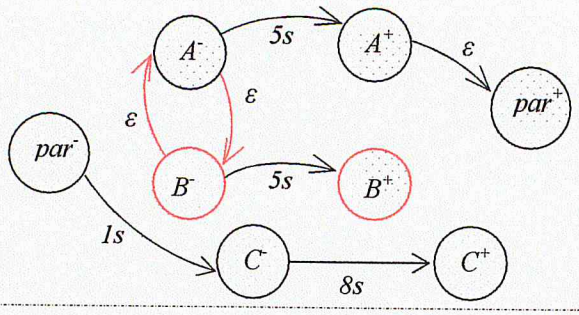

```
...
<video id="X" src="..." begin="7s" end="4s" /> <!--Clip-1-->
...
```

Figure IV.15 : Représentation graphique d'une Incohérence quantitative.

Exemple 1:

Jamais la présentation évoquée en dessous va être jouée puisque les *Clips* 1 et 2 s'attendent mutuellement, c'est-à-dire que le début de A se déclanche lors du début de B, et vice-versa, et parmi tout les arcs qui existent on trouve pas un chemin reliant le nœud *par-* jusqu'à *par+*.

[5] Calculer ou saisit.

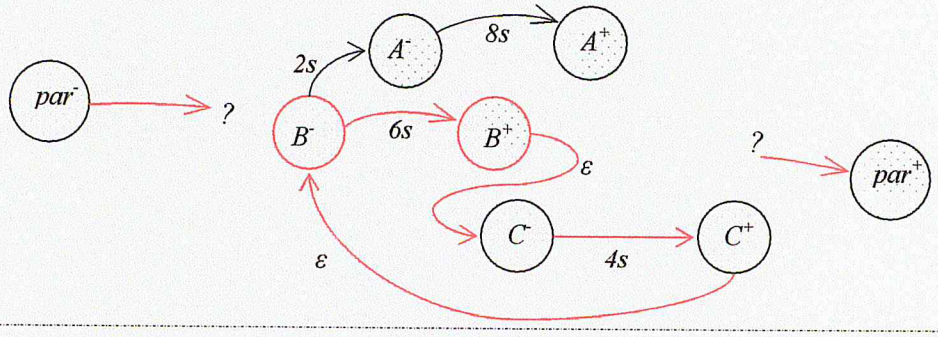



```
...
<par endsync="A">
<img id="A" src="..." begin="id(B)(begin)" dur="5s" /> <!--Clip-1-->
<img id="B" src="..." begin="id(A)(begin)" dur="5s" />  <!--Clip-2-->
<img id="C" src="..." begin="1s" end="9s" />            <!--Clip-3-->
</par>
...
```

Figure IV.16 : Représentation graphique (1) d'une Incohérence causale.

Exemple 2:

Plus compliqué que l'exemple précédant, puisque dans ce cas le *Clip-1* devrai se déclanché *2s* après le *Clip-2*, or celui-ci avec le *Clip-3* s'attendent mutuellement, donc aucun de ces Clips ne va débuter sa présentation, donc la présence de cycle dans les graphe génère une indépendance des fils de leur composite père de ce fait ils perdent les repères de temps, puisque comme on a déjà évoqué précédemment ce dernier[6] a sa propre horloge de temps.




```
...
<par>
<ref id="A" src="..." begin="id(B)(2s)" dur="8s" /> <!--Clip-1-->
<ref id="B" src="..." begin="id(C)(end)" dur="6s" />  <!--Clip-2-->
<ref id="C" src="..." begin="id(B)(end)" dur="4s" />  <!--Clip-3-->
</par>
...
```

Figure IV.17 : Représentation graphique (2) d'une Incohérence causale.

[6] Élément composite (par, seq).

IV.7. Application

Nous allons essayer de présenter dans cette section du chapitre quelques exemples modèles qui expriment la majorité des comportements qui peuvent avoir les simple et composite d'un code SMIL1.0.

a. Situations possible avec le composite seq

Soit la portion d'un document SMIL, présentant deux *Clips* qui se jouent en séquence, qu'on va modifier à chaque fois ses attributs temporels relatifs au composite *seq*, puis apercevoir son impact sur la présentation.

Remarque :

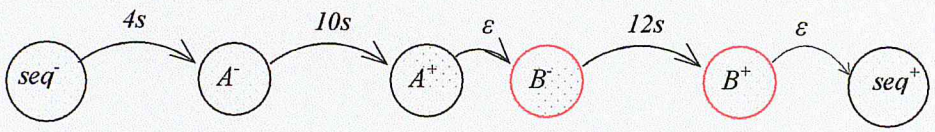
Les schémas hachuré en rouge dans les graphes qui suivent, déterminent une exécution cohérente, mais elles ne seront pas et diffusée par le média Player.

```
...
<seq [att-temporel]>
<audio id="A" src="..." begin="4s" dur="10s" />            <!--Clip-1-->
<animation id="B" src="..." region="reg1" dur="12s" />     <!--Clip-2-->
</seq>
...
```

Figure IV.18 : Exemple-1 d'une partie du code SMIL.

1^er^ cas : [*att-temporel*] ≡ *vide* → <*seq*>

Dans ce premier cas d'exemple, l'élément composite *seq* s'achève quand le dernier *Clip* du bloc, aura accompli son exécution, donc c'est ce dernier qui va mettre fin à l'exécution du composite *seq*.



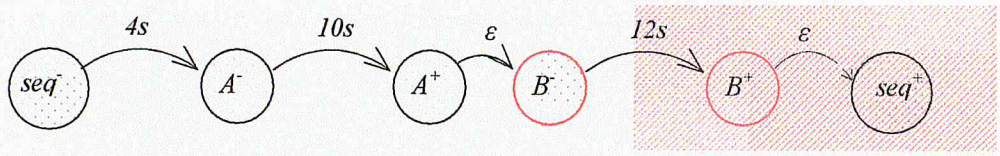
La fin du Clip-2 termine la séquence.

Figure IV.19 : Représentation graphique de l'exemple-1 avec <seq>.

2^ième^ cas : [*att-temporel*] ≡ *dur="valeur"* ou *end="valeur"* → <*seq dur="valeur"*> ou <*seq end="valeur"*>.

Ce cas se présente sous deux autres situations distinctes :

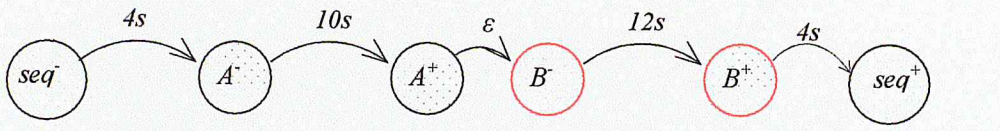
1. Soit le *seq* provoque la fin de son élément fils en cours d'exécution, et ça si la durée du composite *seq* est moins importante vis-à-vis de son fils.



La fin du composite `seq` *provoque la fin du Clip-2.*

Figure IV.20: Représentation graphique de l'exemple-1 avec <seq end="20s">.

2. Soit tout les *Clips* s'exécutent sans aucune interruption de la part du composite *seq*, et ça si la fin du composite *seq* est la dernière à être exécutée.



La fin du composite `seq` *aura lieu après un délai de 4s après la fin du Clip-2.*

Figure IV.21 : Représentation graphique de l'exemple-1 avec <seq dur= "30s ">.

b. Situations possible avec le composite par

De même avec le composite *seq*, soit la portion d'un document SMIL, présentant trois *Clips* qui se jouent en parallèle, qu'on va modifier à chaque fois ses attributs temporels relatifs au composite *par*, puis apercevoir son impacte sur la présentation.

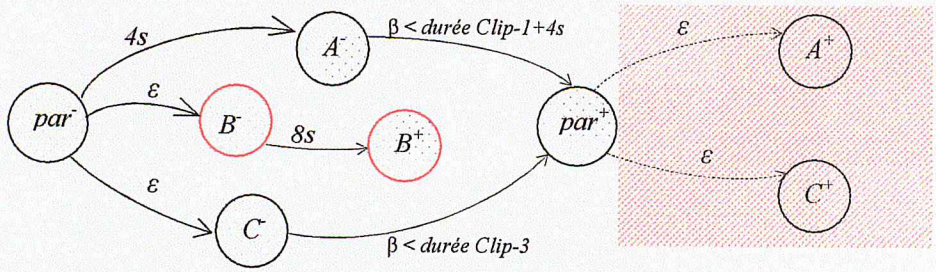
```
...
<par [att-temporel]>
<textstream id="A" src="..." begin="4s" dur="10s" />   <!--Clip-1-->
<img id="B" src="..." region="reg1" dur="8s" />        <!--Clip-2-->
<audio id="C" src="..." region="reg1" dur="15s" />     <!--Clip-2-->
</par>
...
```

Figure IV.22 : Exemple-2 d'une partie du code SMIL.

1er cas : [*att-temporel*] ≡ *dur="valeur"* ou *end= "valeur "* → *<par dur="valeur">* ou *<par end="valeur" >*.

De même que dans le *seq*, on peut trouvé deux situations distinctes, et ça selon la durée ou l'instant fin du *par*, bien sûr par rapport à ses éléments fils.

Donc soit le *par* met fin à la présentation de ses fils, ou tout les *Clips* finissent leurs présentations sans interruption.

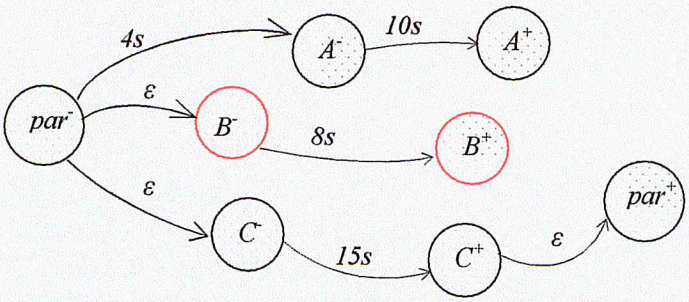


La fin du composite `par` *termine les Clips 1 et 3.*

Figure IV.23 : Représentation graphique de l'exemple-2 avec <par dur= "13s ">.

2ième cas : [*att-temporel*] ≡ *endsync="last"* → *<par endsync="last">*.

L'écriture *<par endsync= "last ">* est semblable à *<par>*[7], donc le composite *par* finit lorsque son dernier *Clip* finit son exécution.

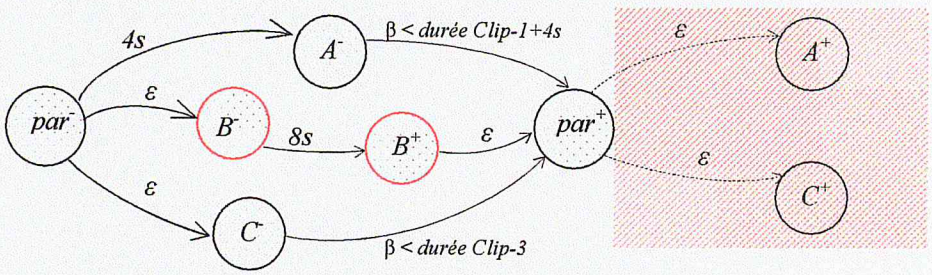


La fin du dernier Clip met fin au composite par.

Figure IV.24 : Représentation graphique de l'exemple-2 avec <par endsync="last">.

3ième cas : [*att-temporel*] ≡ *endsync="first"* → *<par endsync="first">*.

On retrouve que la présentation prendra fin lors de l'achèvement du premier *Clip* en cours, du bloc, de ce fait tout les *Clips* en cours d'exécution vont être arrêté à cet instant.



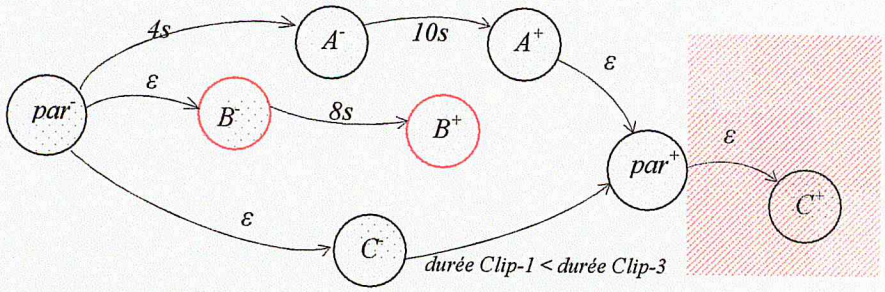
L'achèvement du Clip-2 provoque la fin du composite `par` *et les Clips 1 et 3.*

Figure IV.25 : Représentation graphique de l'exemple-2 avec <par endsync="first">.

[7] [att-temporel] ≡ vide.

4ième cas : [*att-temporel*] ≡ *endsync="id(id-Clip)"* → *<par endsync="id(id-Clip)">*.

La fin du composite *par* aura lieu quand le *Clip* avec l'identifiant spécifié dans les paramètres temporels du *par*, aura terminé sans exécution.



La fin du Clip désigner met fin au composite par et aux autres médias dans le bloc.

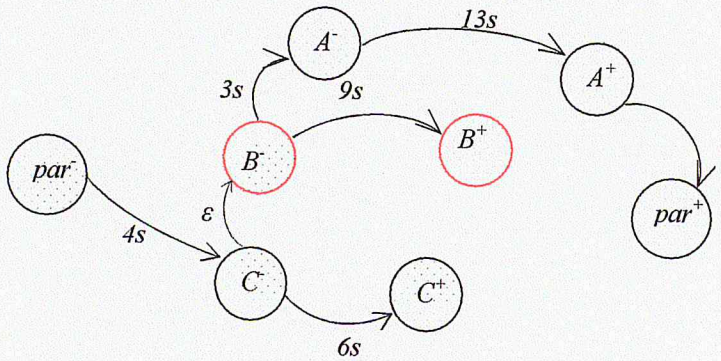
Figure IV.26 : Représentation graphique de l'exemple-2 avec < par endsync="id(A)">.

c. Situations possible avec des relations causales

Soient les deux exemples suivants :

Exemple 1 :

Le *Clip-3* commence *4s* après le début du *par*, or son début cause le déclanchement immédiat du *Clip-2*, ainsi le début de ce dernier cause le déclanchement du *Clip-1* avec un retard de 3s du début de *Clip-2*.

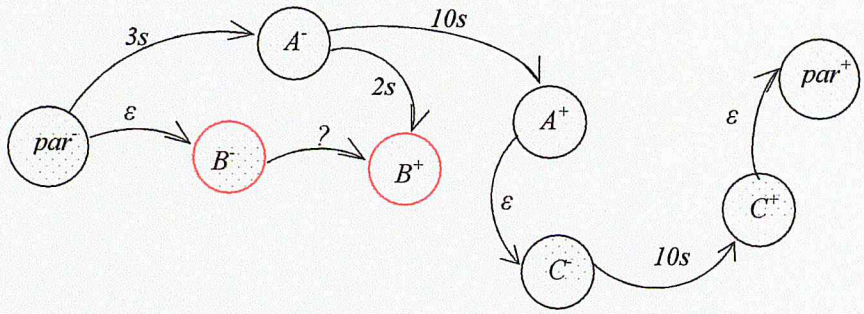



```
...
<par>
<audio id="A" src="..." begin="id(B)(3s)" dur="13s"/>    <!--Clip-1-->
<audio id="B" src="..." begin="id(C)(begin)" dur="9s"/> <!--Clip-2-->
<video id="C" src="..." begin="4s" end="10s" />          <!--Clip-3-->
</par>
...
```

Figure IV.27 : Représentation graphique de l'exemple-1 avec des relations causales.

Exemple 2 :

Dans l'exemple qui suit, on remarque que le *Clip-1* commence *3s* après le début du composite *par*, et cause à l'aide de la relation *end="id(A)(2s)"* la fin du *Clip-2 2s* après le début du *Clip-1* et cause à l'aide de la relation *begin="id(A)(end)"* le début du *Clip-3* immédiatement après la fin du *Clip-1*.

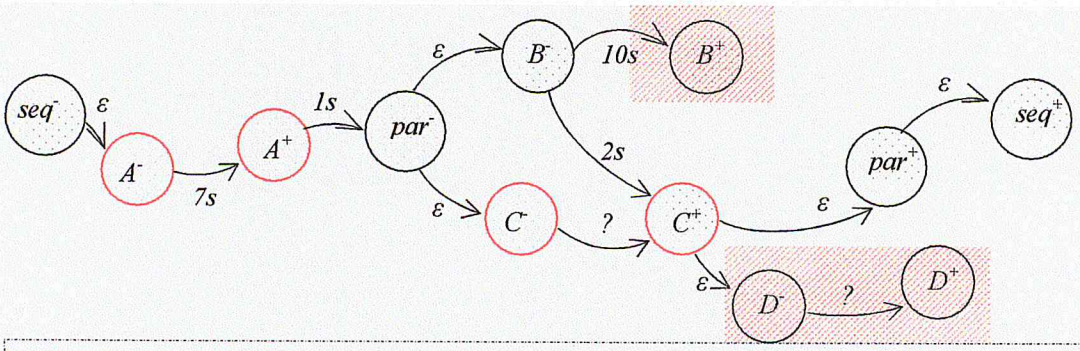



```
...
<par>
<audio id="A" src="..." begin="3s" dur="10s"/>          <!--Clip-1-->
<audio id="B" src="..." end="id(A)(2s)" />              <!--Clip-2-->
<video id="C" src="..." begin="id(A)(end)" dur="10s" /> <!--Clip-3-->
</par>
...
```

Figure IV.28 : Représentation graphique de l'exemple-2 avec des relations causales.

d. Situations possible avec des blocs imbriqués

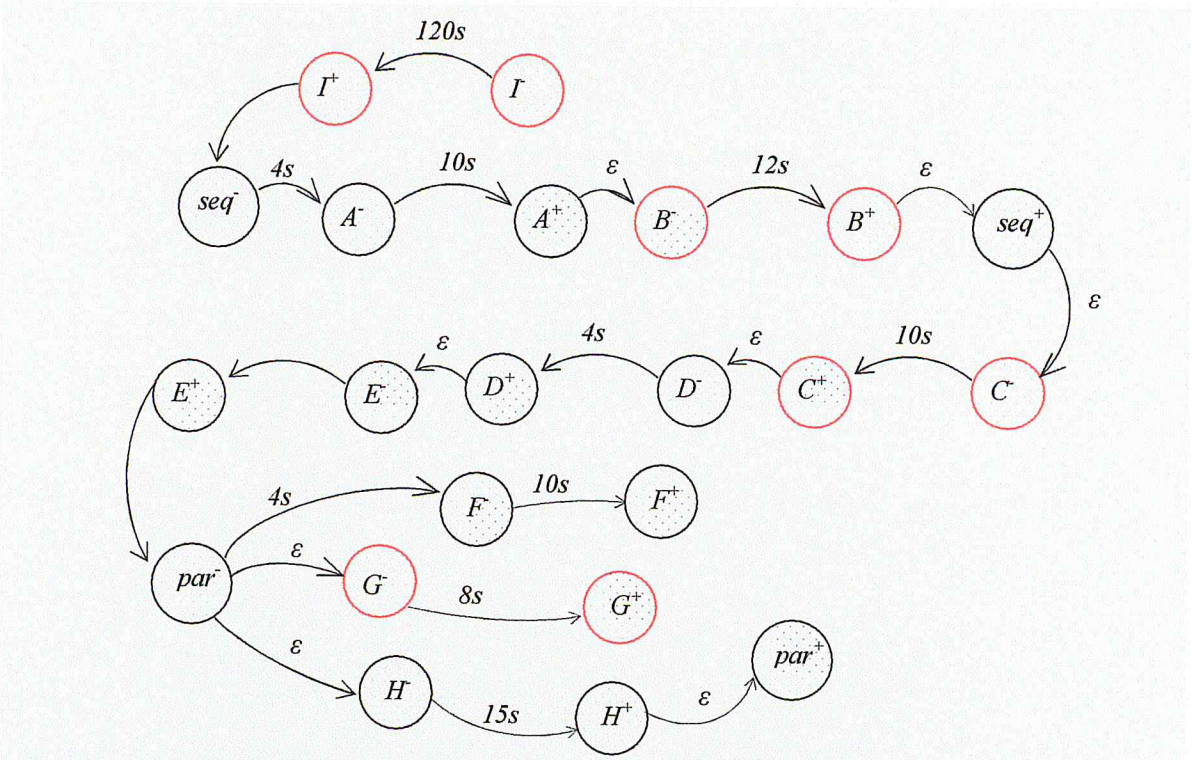
Dans cette portion d'exemple aussi, nous avons un composite *par* dans un autre de type *seq*, or le bloc *par* se comporte comme un élément simple dans une séquence de médias.




```
...
<seq>
  <img id="A" src="..." dur="7s" />                    <!--Clip-1-->
  <par begin="1s" endsync="first">
    <audio id="B" src="..." dur="10s"/>                <!--Clip-2-->
    <ref id="C" src="..." end="id(B)(2s)" />           <!--Clip-3-->
    <img id="D" src="..." begin="id(C)(end)" />        <!--Clip-4-->
  </par>
</seq>
...
```

Figure IV.29 : Représentation graphique d'un exemple un composite imbriqué.

e. Exemple général




```
<smil>
<head>
<layout>
<root-layout width="320" height="240" background-color="black" />
<region id="reg_1" left="10" top="10" width="300" height="200"/>
<region id="reg_2" left="10" top="210" width="300" height="25"/>
</layout>
</head>
<body>
<audio id="I" src="..." dur="120s" />
<seq>
  <audio id="A" src="..." begin="4s" dur="10s" />
  <animation id="B" src="..." region="reg1" dur="12s" />
</seq>
<img id="C" src="1.jpg" region="reg_1" dur="4s" />
<img id="D" src="2.jpg" region="reg_1" begin ="7s" dur="4s"
end="10s"/>
<audio id="E" src="moliendo.mp3" />
<par>
  <audio id="F" src="..." begin="3s" dur="10s"/>
  <audio id="G" src="..." end="id(A)(2s)" />
  <video id="H" src="..." begin="id(A)(end)" dur="10s" />
</par>
</body>
</smil>
```

Figure IV.30 : Représentation graphique d'un code complet.

Conclusion

Nous avons vu dans ce chapitre comment traduire le code SMIL en un graphe temporel. Ainsi le graphe obtenu sera analysé dans le but de détecter la présence d'incohérences dans le document multimédia.

De ce fait nous retenons les points essentiels à la réalisation de notre modèle qui sont :

- La définition de la cohérence.
- La construction d'un système de contrainte de type STP, puis traduction de ce système en un graphe.
- La représentation graphique des contraintes et la détection d'incohérences.

Chapitre V

DESCRIPTION DE SMIL_GEN

Chapitre V

DESCRIPTION DE SMIL_GEN

Au cours du chapitre précédent, nous avons pu voir les différents outils conçus pour construire des documents multimédias au format SMIL ; Nous avons vu comment les besoins liés aux auteurs étaient plus ou moins couverts par ces environnements.

À partir de ces informations, nous avons essayés de concevoir un outil qui se rapprochera le plus de ce que nous considérons être l'environnement d'édition idéal pour les documents multimédias au format SMIL ; Pour cela, nous nous inspirerons des différents outils vus dans les chapitres, I et II.

Nous élargirons notre domaine de référence pour chercher des idées et des solutions que présente chaque outil dans plusieurs domaines.

Dans l'éditeur, la vue temporelle ainsi que la spécification du scénario prendront une place doyenne dans notre conception. SMIL étant un langage décrivant des documents multimédias qui se distinguent par le fait que la dimension temporelle y joue un rôle important ; C'est pourquoi, nous nous devons d'assurer un environnement dont le modèle d'édition soit proche de la technique du WYSIWYG.

Dans ce chapitre, nous allons largement nous intéresser aux fonctionnalités fournies par notre environnement. Nous donnerons une description détaillée de notre outil *SMIL_GEN*[1], et nous présenterons l'architecture générale de notre environnement d'édition. Ensuite, nous nous intéresserons aux modes de visualisation de notre outil, et enfin nous étofferons les différents services d'aide que devra fournir notre environnement pour faciliter la tâche d'édition aux auteurs.

[1] Smil_Generator, générateur de code SMIL.

V.1. Spécification de l'architecture de *Smil_Gen*

V.1.1. Architecture générale

L'architecture générale de notre environnement comporte trois fonctions :

- Fonctions liées aux opérations d'édition.
- Fonctions liées à la visualisation de l'information.
- Fonctions d'aide aux l'auteurs.

V.1.2. Formalisme d'édition de *Smil_Gen*

L'édition du document multimédia peut se faire selon deux niveaux différents vis-à-vis les capacités de l'auteur, or la première solution, celle de notre recherche qui consiste à l'édition via un formalisme environnement auteur, c'est-à-dire par l'intermédiaire d'interface graphique, ou alors, avec un formalisme de langage de sauvegarde.

Dans notre cas, l'outil conçu devra se conjuguer avec le formalisme attaché au langage SMIL. Nous serons automatiquement appelé à utiliser un formalisme *auteur* avec une approche relationnel, ce qui représente un sérieux avantage.

En effet, des formalismes comme le formalisme absolu ou événementiel sont trop complexes et rendent le document difficilement maîtrisable pour les auteurs. De plus, lors de notre étude dans les chapitre précédant nous avons aboutie que l'approche basée sur les groupes et les relations était plus facilement compréhensible, et offrait une édition plus souple.

V.2. Les règles exigées pour *Smil_Gen*

Dans cette section, nous allons étoffer les différentes règles liées à *Smil_Gen*, ces règles sont issues des observations et conclusions faites à travers plusieurs outils vus dans le chapitre I. Elles sont aussi le fruit de l'étude faites sur le langage SMIL dans le chapitre II. Le but de ces règles n'est pas de limiter les auteurs dans leur expression, mais ont pour but d'améliorer l'édition de *Smil_Gen*.

V.2.1. Description des médias

Les médias doivent être caractérisés par un ensemble d'attributs qui sont :

- Le type du média.
- La source du média.
- Les attributs de présentation du média : couleur, taille, ordre des positions des régions.
- Les attributs temporels : début, fin, durée.

V.2.2. Description du scénario

Le formalisme lié à SMIL permet d'associer à chaque groupe de média un élément *par* ou *seq*, ainsi qu'un ensemble de relations et des attributs. Cet ensemble d'informations permet de définir le placement relatif de chaque objet d'un groupe par rapport au début de ce même groupe. Le placement relatif d'un objet est donc le résultat d'un calcul entre ces différentes informations qui peuvent être :

- *Cohérentes* : Dans ce cas la solution trouvée satisfera toutes les informations données par l'auteur.
- *Incohérentes* : Dans ce cas, le système indiquera à l'auteur qu'il y a une incohérence avec les valeurs des attributs.

V.2.3. Simplicité d'édition

L'environnement auteur doit offrir une édition à manipulation simple et direct, c'est-à-dire, que l'auteur doit pouvoir éditer les attributs de ses objets dans les vues spatiale et temporelle.

De plus, le système doit maintenir les relations existantes entre les objets tout le long du cycle d'édition, y compris lorsque l'auteur déplace graphiquement les objets :

- A l'aide de la vue de présentation pour les relations spatiales.
- A l'aide de la vue temporelle pour les relations temporelles.

V.2.4. Spécification multi vues

Le formalisme essentiellement relationnel lié à SMIL implique une grande complexité de la tâche de visualisation, et nécessite la mise en place de services de cohérence et de Calcul.

Nous allons nous intéresser maintenant aux services de visualisation que doit fournir notre environnement auteur *Smil_Gen* ; Ces services de visualisation ne doivent pas être indépendants de l'édition et se feront au travers de *vues*[2], or offrir une seule vue à l'auteur serait insuffisant. Le système auteur doit donc offrir plusieurs vues adaptées à la visualisation des informations. Les vues indispensables pour *Smil_Gen* sont :

- **La vue spatiale :** Elle permet de présenter le document, et d'éditer ses informations spatiales, D'un point de vue édition, celle-ci doit permettre à l'auteur d'éditer les attributs spatiaux par des manipulations simple.

[2] Entité graphique qui permet de visualiser à l'écran une information liée à un document. Cette information peut être d'ordre temporelle, spatiale ou hypertexte.

- **La vue temporelle :** Cette vue présente la dimension temporelle du document et offre des mécanismes permettant de visualiser les relations ainsi que les informations liées à la hiérarchie temporelle. D'un point de vue édition, cette vue doit permettre la manipulation des différents attributs temporels.
- **La vue hiérarchique :** Elle sert à visualiser les structures temporelle et spatiale du document. Cette vue doit aussi servir de support pour l'édition, en facilitant les opérations de restructurations tout en permettant à l'auteur de visualiser les objets associés à chaque groupe.
- **La vue attributs :** C'est les différentes commandes graphique qui permettent d'accéder aux différents attributs et objets du document.
- **La vue source :** C'est le but et la phase finale de notre éditeur puisque elle permet de visualiser la source du fichier de sauvegarde.

V.2.4.1. Dépendance entre les vues

Pour faciliter le travail de l'auteur dans les différentes vues, on doit mettre en oeuvre une synchronisation sur l'objet sélectionné, qui impose au système, lorsque l'auteur change de vue, de ne pas perdre les informations éditionnées

L'utilisation d'un système multi vues nécessite une synchronisation des informations visualisées dans chacune des vues, de manière à garantir une certaine cohérence à l'auteur, c'est-à-dire que la vue source c'est la traduction de la vue spatiale, temporelle et hiérarchique.

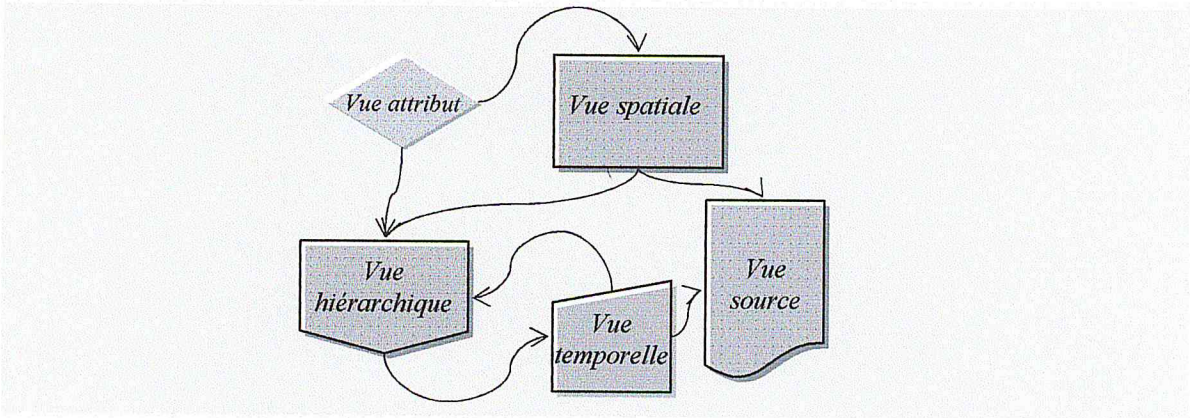


Figure V-1 : Relations entre les vues.

V.2.5. Les services d'aide

L'outil *Smil_Gen* doit assurer des services complets de vérification de cohérence et d'aide lors de la spécification, et ceci, dans les cas d'erreur ou d'incohérence du document. Ces services se concrétisent à travers une visualisation et un diagnostic élaboré des erreurs, par exemple des alertes d'erreur à l'aide d'une fenêtre ou des messages d'aide.

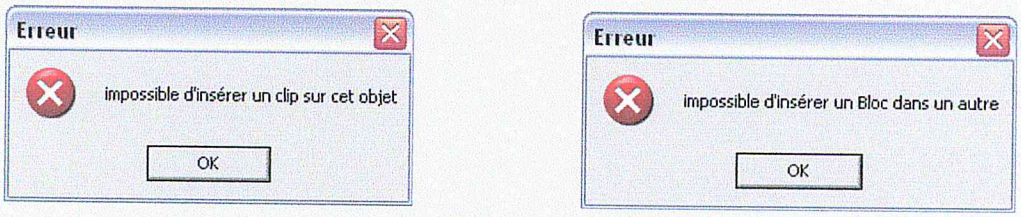


Figure V-2 : Fenêtre de message d'erreur.

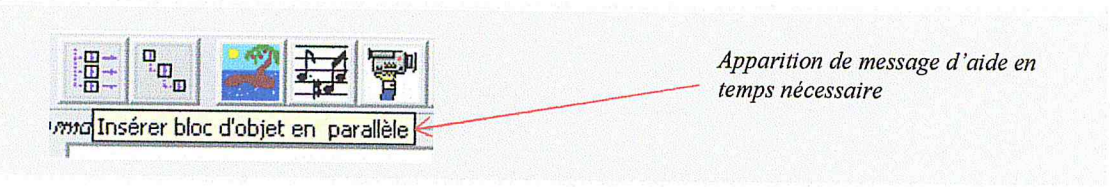


Figure V-3 : Message d'aide.

V.2.5.1. Vérification de la cohérence

Comme nous l'avons vus dans le chapitre II, le langage SMIL peut présenter certaines incohérences. Donc, nous serons appelé à faire face aux problèmes liés à ces incohérences, tout en essayant d'apporter des solutions concrètes. Ces solutions, auront pour but de signaler à l'auteur l'incohérence, et éventuellement de la corriger par lui puisque le système de vérification ne peut pas effectuer cette tâche, car c'est une préférence de l'utilisateur pour la création de son scénario, donc la machine ne peut pas connaître son choix de présentation.

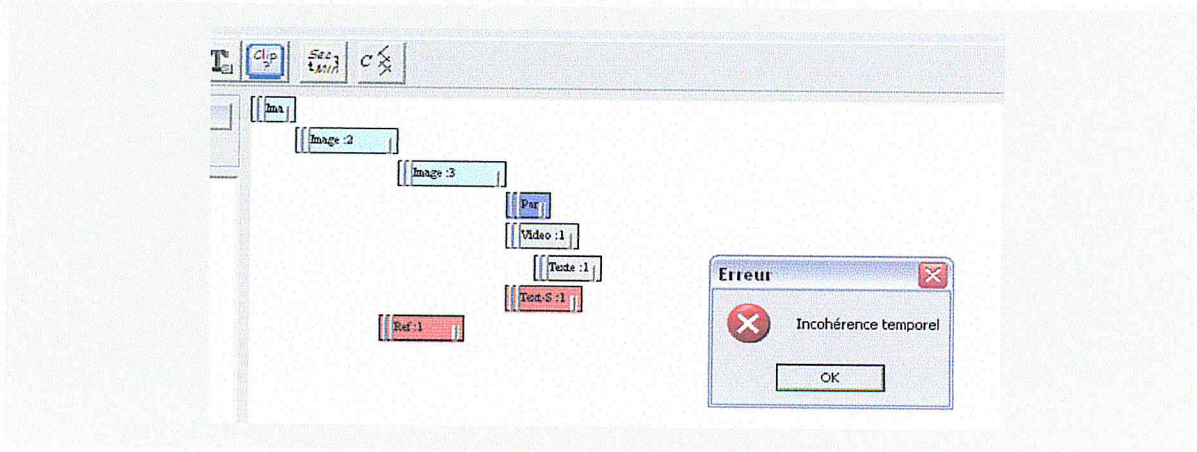


Figure V-4 : Message d'aide.

Dans la *figure V-4*, nous remarquerons qu'il y a deux objets colorés en rouge désignant une incohérence temporelle, pour l'illuminer on doit déplacer ces deux objets et cela pour modifier leurs attributs temporels, et puis revérifier s'il ne reste pas encore d'incohérence temporelle, à l'aide de commande dans ***Smil_Gen***.

V.2.5.2. Protection contre les données erronées

Smil_Gen assure une bonne protection contre les données erronées que peut éventuellement introduire un auteur dans la vue attributs, ou par manipulation des objet ou de ces attributs dans les autres vues. Par exemple, si l'auteur entre une lettre dans un champ numérique, *Smil_Gen* ne l'accepte pas, en supprimant tout caractère différent d'un chiffre. Par ailleurs, il existe aussi une autre protection concernant les attributs "*begin*" et "*end*" qui assure la cohérence du document. Par exemple, si la valeur de l'attribut "*end*" est inférieure à celle de l'attribut "*begin*", le système signale l'erreur et ne prend pas en compte la donnée erronée.

V.3. Fonctionnement interne de Smil_Gen

On peut distingué facilement dans notre logiciel deux parties complémentaires la première attaché à la partie spatiale et l'autre temporelle.

V.3.1. Le fonctionnement spatial

C'est l'étape initiale que l'utilisateur doit suivre, puisque on doit concevoir notre interface de présentation, mais pour cela nous devons connaître quelques paramètres du système où l'exécution de notre scénario va se dérouler, pour cela avant toute action d'édition nous devons spécifier la résolution de notre moniteur pour que *Smil_Gen* prend en charge toutes les modifications nécessaire.

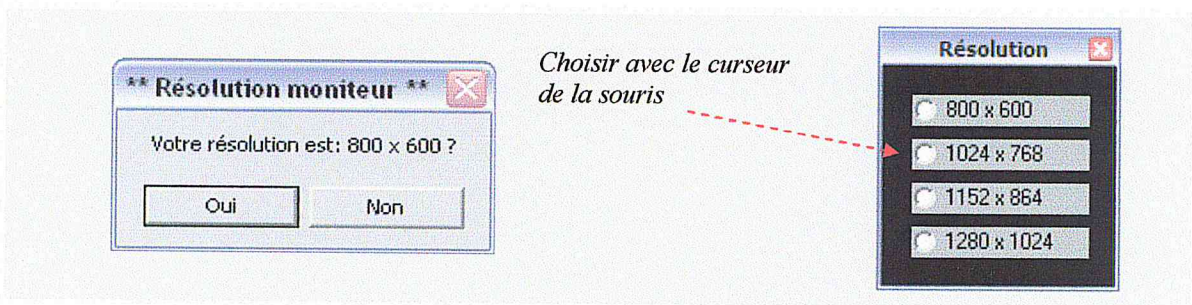


Figure V-5 : Fenêtre pour le choix de résolution du moniteur.

Remarques :

- Les valeurs de mesure spécifier dans *Smil_Gen*, sont exprimées en pixel.
- L'interface graphique qui représente notre écran d'affichage n'est pas de taille concret mais, les proportions par rapport aux régions crées sont réelles avec une échelle d'affichage réduite.

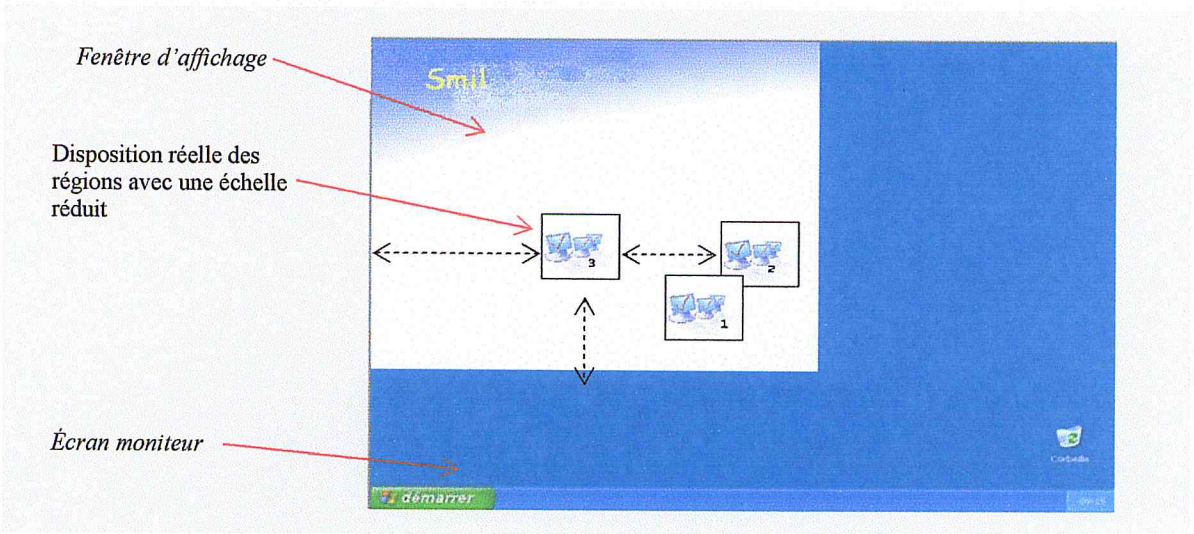


Figure V-6 : Disposition des régions par rapport au layout.

Création des régions

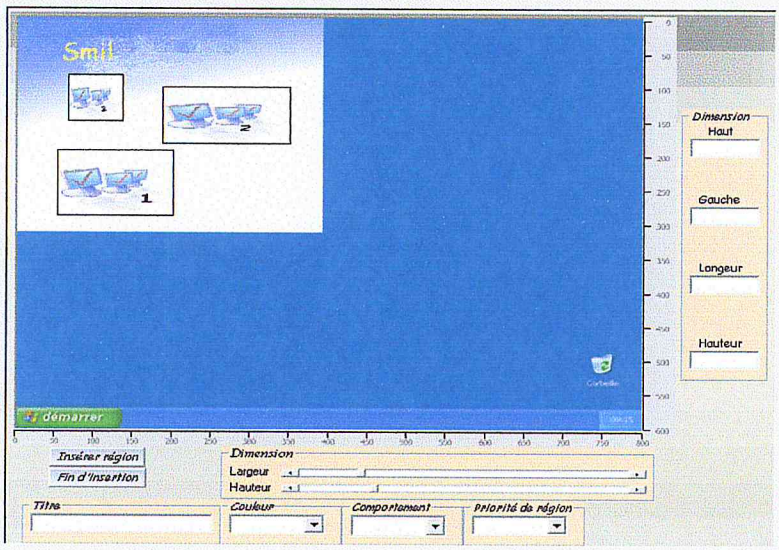


Figure V-7 : La vue spatiale de la présentation du document.

Nous pouvons décrire le processus de création d'une région comme suit :

- Insertion de région à l'aide de la commande "*Insérer région*",
- le déplacement du rectangle se fait on maintenant le bouton gauche de la souris enfoncé et le curseur sur le rectangle,

- réglage de la taille du rectangle à l'aide des deux "*scrool_bar*", désignant largeur et hauteur,
- récupération automatique des coordonnées de la région,
- prise en compte de la région dans la vue hiérarchique.

V.3.2. Le fonctionnement dynamique

C'est la partie la plus sensible de notre travail puisqu'on touche le concept de temps, dans cette étape on intègre les médias et éléments nécessaires à notre présentation du scénario à l'aide de la barre illustrée dans la *Figure V-8*, à chaque insertion d'objet on a une vue hiérarchique de la structuration des objets, illustrée par la *Figure V-9*, et une vue de présentation temporelle illustrée dans la *Figure V-11*, ainsi par un simple double clique sur le nœud de l'objet requis,ou sur sa représentation dans la vue temporelle (time line), nous donne la possibilité d'affecté à l'objet les différents propriétés nécessaires, illustrées par la *Figure V-10,* pour les différents besoins du scénario.

Figure V-8 : Commande d'intégration de médias et éléments.

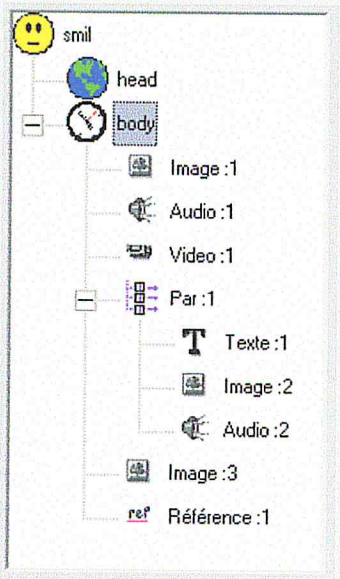


Figure V-9 : La vue hiérarchique de la structuration du document.

Figure V-10 : La vue Attribut des propriétés d'un Clip.

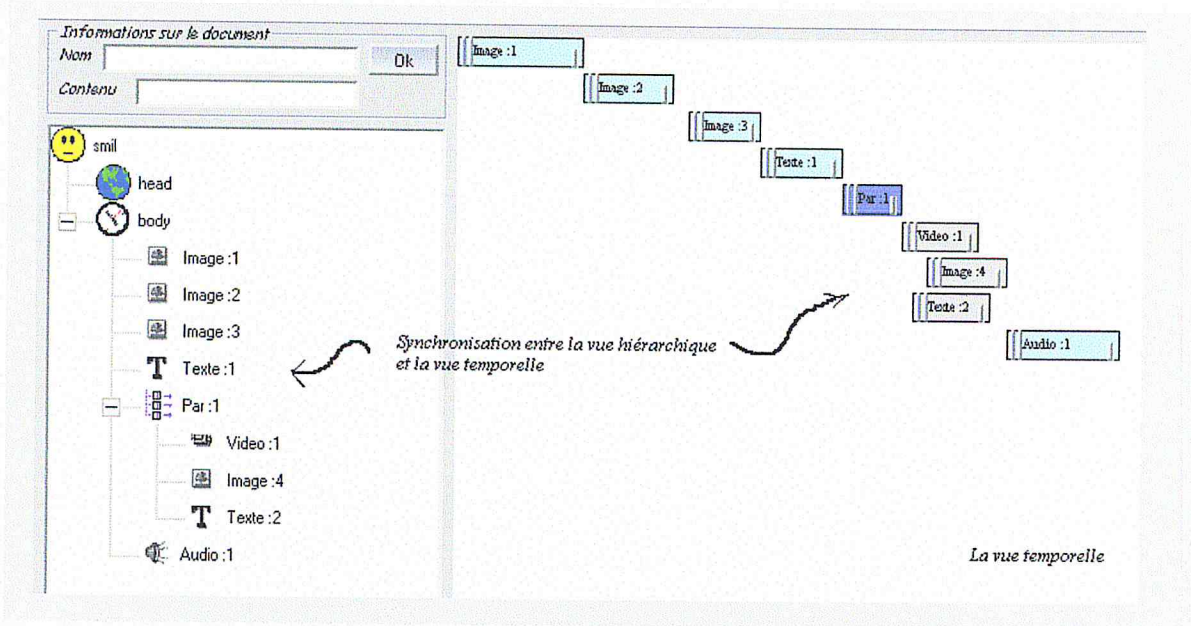


Figure V-11 : La vue temporelle du document.

V.4. Calcul et traitement dans *Smil_Gen*

Chaque information introduite par manipulation directe ou indirecte dans *Smil_Gen*, va être mise dans des matrices de sauvegarde, relative au type de données.

- Dans le cas de la vue spatiale les données résultant de la création des régions vont êtres sauvegardés dans une matrice pour faire des traitements et calculs, illustrées par la *Figure V-12.*

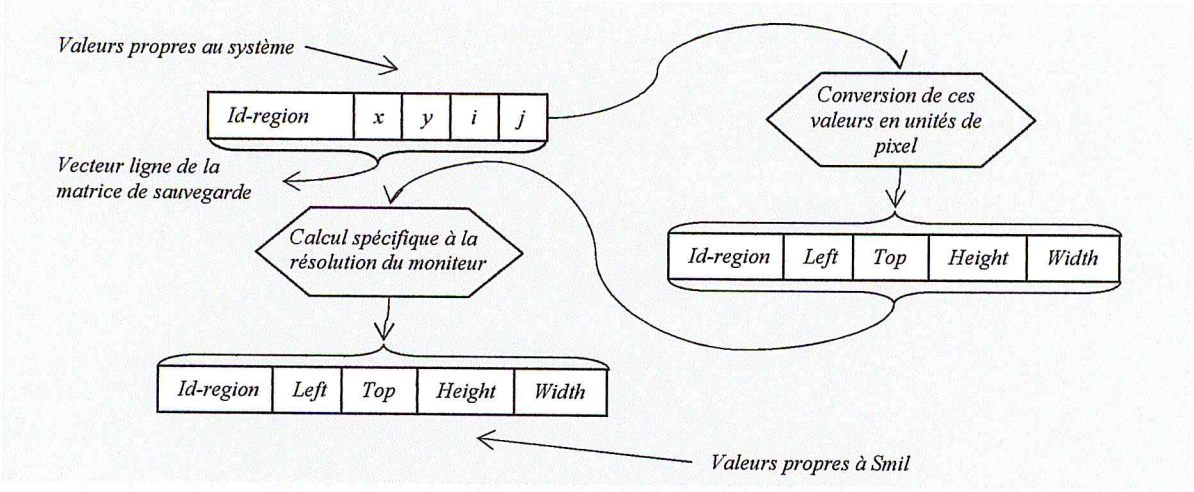


Figure V-12 : Calcul de valeur spatiale.

- Pour la cohérence du document SMIL, la sauvegarde des attributs du temps[3], se fait par bloc de trois *Clips,* c'est-à-dire on extrait toutes les données temporelles de trois *Clips* éditionnées, puis on fait des opérations de vérification de la cohérence comme un début négatif ou autres comme vue dans le chapitre précédant, puisque le langage SMIL lors de son exécution, ne vérifie pas la cohérence de son contenue entièrement car il n'a pas de compilateur, or il ne peut tenir en mémoire que les attribut de trois *Clips,* l'exemple suivant illustrera mieux la procédure d'exécution d'un document SMIL.

Exemple :

Le code *Smil* illustré dans la *Figure V-13,* est incohérent, même s'il est syntaxiquement correct, dans *tab-4* de la *Figure V-13*, le début du *Clip-A4* est référencié par la fin du *Clip-A1*, or lors de l'exécution du Clip-A4 il n'a aucune donnée concernant le *Clip-A1*dans la matrice de sauvegarde, de ce fait le Clip-A4 sera impossible à jouer.

[3] Début, fin, durée.

```
<smil>
<head>
<layout>
<root-layout width="320" height="240" background-color="black" />
<region id="reg1" left="10" top="10" width="300" height="200"/>
<region id="reg2" left="10" top="210" width="300" height="25"/>
</layout>
</head>
<body>
<img id="A1" src="1.jpg" region="reg1" begin ="1s" dur="4s" />
<img id="A2" src="2.jpg" region="reg1" begin ="7s" dur="4s" end="10s"/>
<img id="A3" src="3.jpg" region="reg1" begin="-2s" dur="4s" />
<img id="A4" src="4.jpg" region="reg1" begin ="id(A)(end)" dur="4s"/>
<img id="A5" src="5.jpg" region="reg1" dur="4s" />
<textstream id="A6" src="text.rt" region="reg2"/>
<audio id="A7" src="moliendo.mp3" />
</body>
</smil>
```

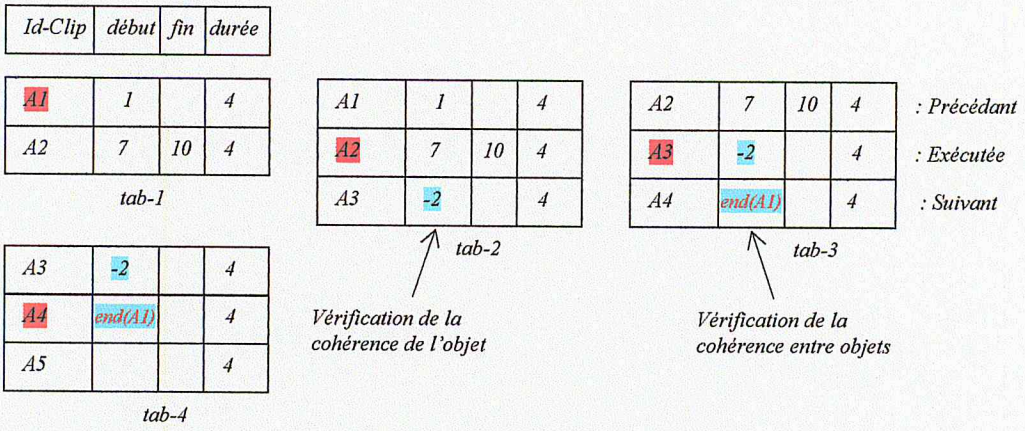


Figure V-13 : Cycle d'exécution d'un code SMIL.

V.5. Vue textuelle

La vue textuelle est accessible facilement par l'interface de *Smil_Gen*, nous aurions pu se contenter de la vue hiérarchique et temporelle qui offrent tout de même un bon aperçu sur le code SMIL. Mais il n'en demeure pas moins, qu'une vue proprement textuelle illustrée par la *Figure V-14*, est plus adaptée à nos besoins et ça pour d'éventuel utilisateur expérimenté qui peut éditer directement le code *Smil* dans la vue textuelle, ainsi on obtiendra par la suite un fichier qui contient un document multimédia cohérent au format *Smil*, qui peut être joué sur n'importe quelle *Player*, et même portable d'une machine à une autre car il n'est pas paramétré avec le système ou il a été éditer.

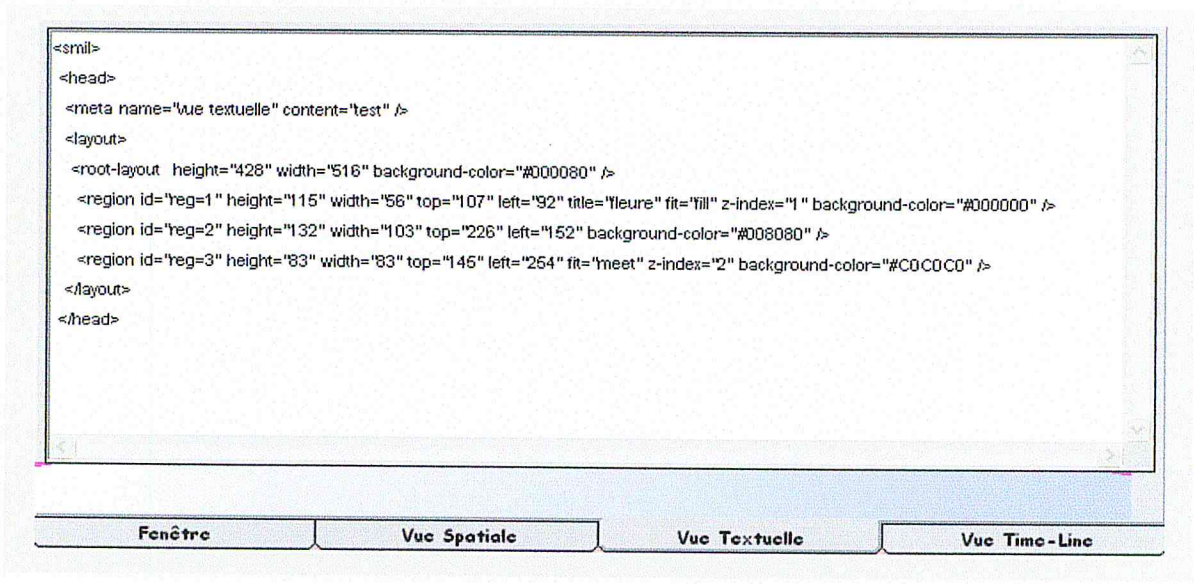


Figure V-14 : La vue textuelle.

Conclusion

Au cours de ce chapitre, nous venons de proposer un environnement auteur pour la spécification de documents multimédias. Les différentes propositions qui ont été faites, permettent aussi de répondre aux différents besoins de visualisation et de présentation définis au début de ce chapitre. L'environnement résultant permet donc d'apporter à la fois une réponse concernant la simplicité d'édition, ainsi que sur les services offerts à l'auteur, tel que la visualisation des vues à tout moment, et la vérification de cohérence.

En ce qui concerne l'édition, nous avons mis en relief les interfaces graphiques utilisées dans ***Smil_Gen***, mais surtout, nous avons pu enrichir notre éditeur avec des nouveautés qui sont venues donner plus de poids à notre réalisation. Ces interfaces ont été conçues dans le souci d'offrir une bonne vision à l'auteur sur la présentation du document (*point de vue spatial*), et une bonne perception du scénario, en représentant les relations dans la vue hiérarchique, et en respectant les proportionnalités des médias dans la vue présentation temporelle.

De plus, un effort considérable a été concédé aux détections des erreurs et aux diagnostics, qu'il n'est pas visible par l'utilisateur, mais c'est la partie la plus importante qui rend le document cohérent. Nous espérons ainsi, avoir répondu à toutes les contraintes possibles qui peuvent être liées à la conception de ***Smil_Gen.***

Chapitre VI

RÉALISATION DE L'ÉDITEUR SMIL_GEN

Chapitre VI

RÉALISATION DE L'ÉDITEUR *SMIL_GEN*

Introduction

Nous approchons maintenant l'aspect de réalisation de notre but. Qu'il s'agit d'éditeur graphique pour les documents multimédias cohérents au format SMIL, appelé ***Smil_Gen***. Ce dernier permet de présenter le code SMIL dans un format cohérent.

VI.1. Contexte matériel et logiciel

L'implémentation de ***Smil_Gen*** s'est effectuée sur un PC sous le système d'exploitation Microsoft Windows, L'outil choisi pour le développement de notre application est le langage Visual Basic version 6.0.

Notre choix du langage Visual Basic de Microsoft s'est fait parce que c'est un outil rapide et facile à utiliser pour créer des applications Microsoft Windows de type graphique comme dans notre cas la réalisation d'un éditeur graphique qui génère le code ***Smil***. Ainsi il permet de développer des logiciels complets, et offre une gamme complète d'outils qui simplifient et accélèrent le développement d'applications.

VI.2. L'éditeur graphique pour les documents au format SMIL : *Smil_Gen*

Le but de l'éditeur ***Smil_Gen*** est de générer un code ***Smil*** syntaxiquement correct, et temporellement cohérent, de telle façon que ce code ***Smil*** soit transportable sur d'autres PC et permettra de jouer la même présentation, même s'il ont différentes configurations.

VI.2.1. Description de l'interface utilisateur

Dès le lancement de l'application, la fenêtre principale, illustrée par la *Figure VI-1* s'affiche contenant essentiellement une barre d'outils pour les différents besoins d'édition comme l'insertion d'objet média ou de bloc parallèle ou séquentielle et même l'analyse du code *Smil*.

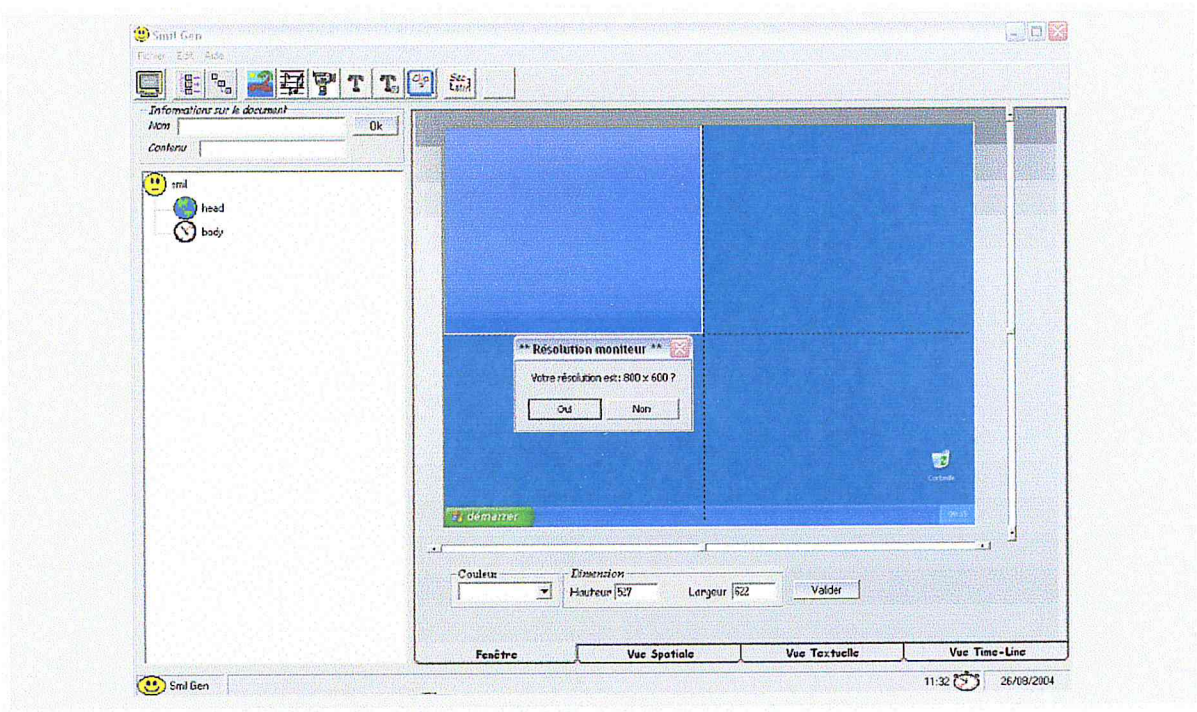


Figure VI-1 : Fenêtre principale de l'application.

Et après le choix de la résolution moniteur, on peut passer d'une vue à l'autre, d'un simple clic sur les "*onglets*".

VI.2.2. Fonctionnement de *Smil_Gen*

a. Démonstration 1 :

Nous illustrons cette partie par un exemple qui démontre l'enchaînement des étapes d'édition jusqu'à la génération du code Smil. L'édition d'un document multimédia se divise essentiellement en deux étapes, la spécification des régions d'affichages des médias visuels, et le paramétrage temporel de ces derniers.

L'idée, est de réaliser un minuteur numérique de type ***ss***, de 10 jusqu'à 00, qui fait déclancher un *Clip* audio. Pour cela il faut créer une zone d'affichage pour le minuteur et faire les réglages nécessaires pour l'enchaînement temporel.

Étape 1 :

En premier temps, nous créons la fenêtre principale d'affichage puis une région dedans comme suit :

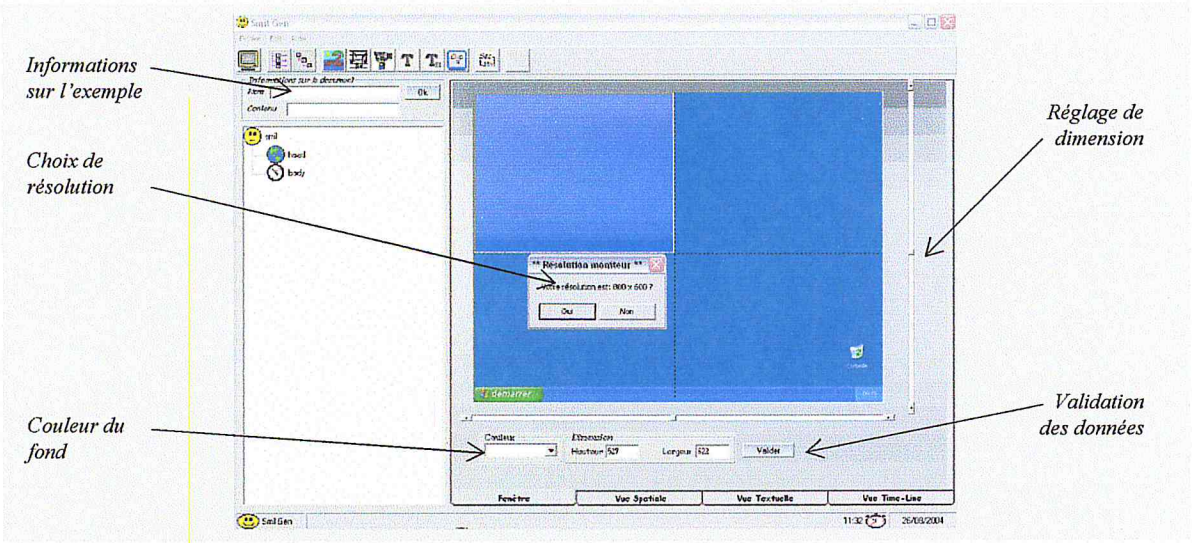


Figure VI-2 : Création de la fenêtre d'affichage.

À la fin des différents régalages nous validons nos préférences à l'aide de la commande *Valider*.

Deuxièmement nous définissons les régions nécessaires à nos besoins

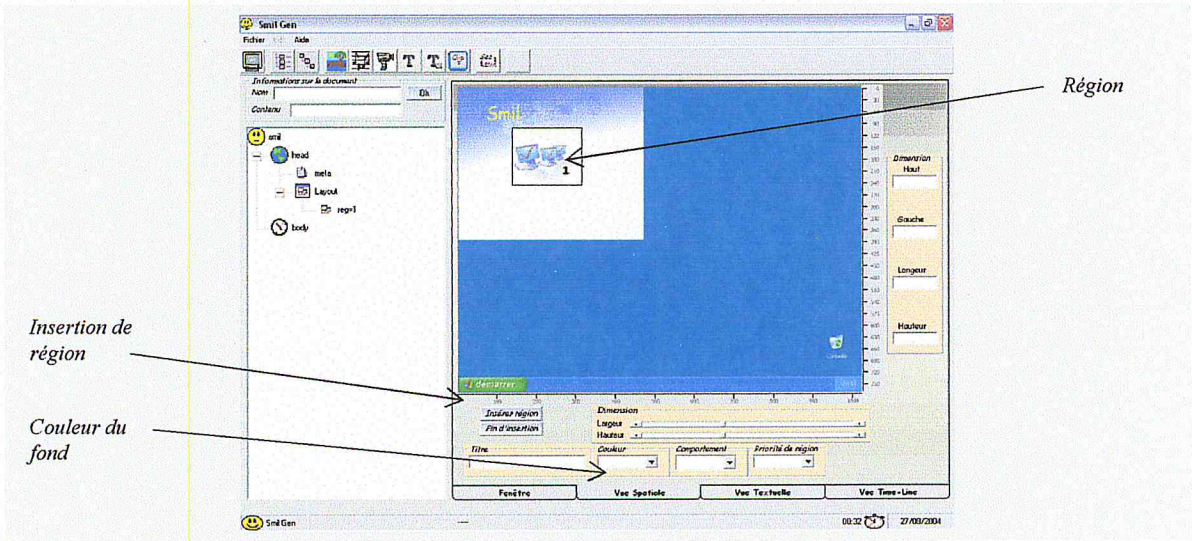


Figure VI-3 : Insertion des régions.

Étape 2 :

À ce niveau, nous insérons tous les médias nécessaires pour la réalisation du scénario ; puis par un clic sur chaque objet dans la vue Time-line, nous obtiendrons la fenêtre illustrée par la *Figure VI-5* qui nous servira pour les différents réglages.

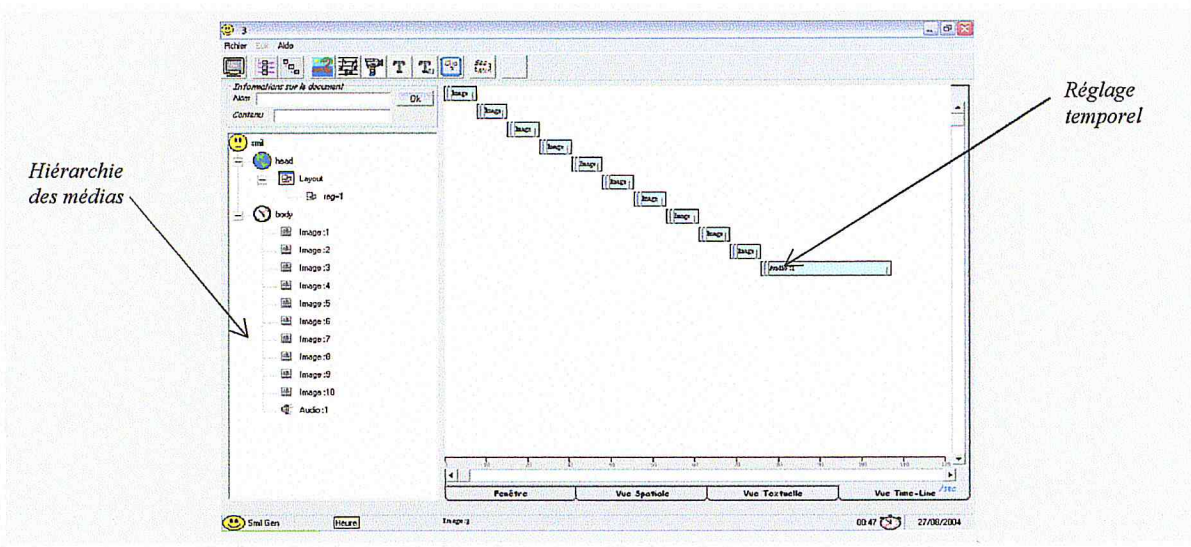


Figure VI-4 : Insertion des régions.

Figure VI-5 : Fenêtre de réglage.

À la fin de tous les réglages nous arrivons à la fin de notre objectif qui est la génération du code *Smil,* illustrée par la *Figure VI-6.*

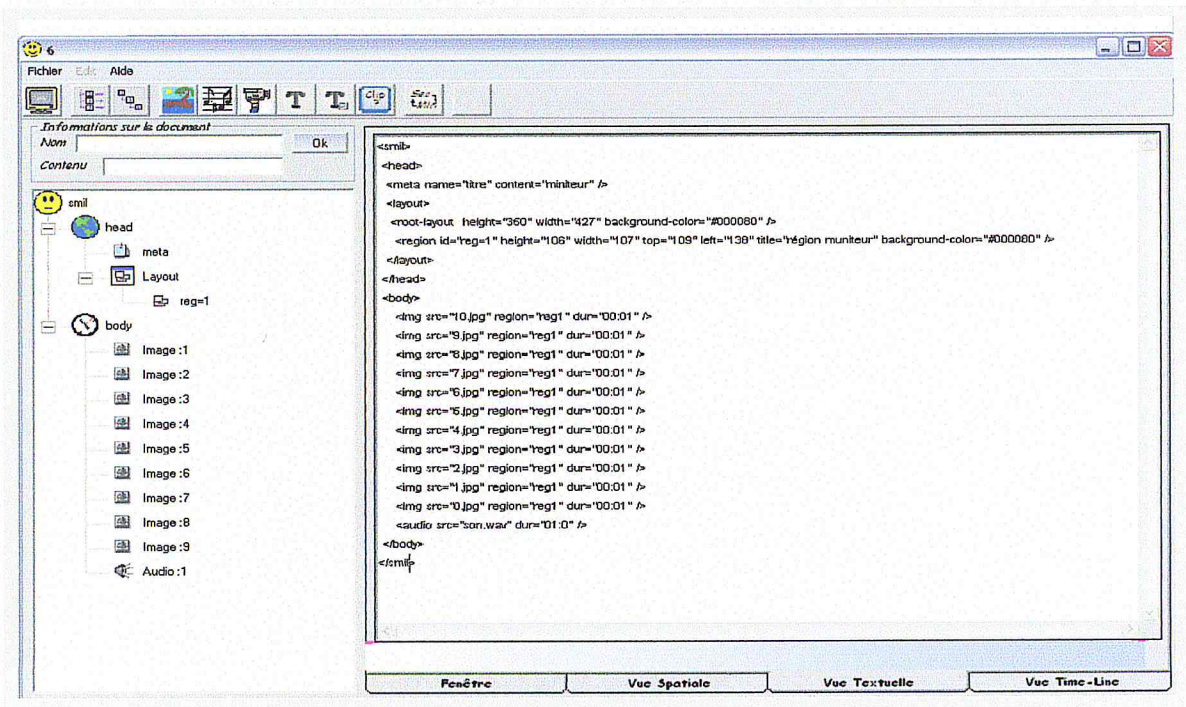


Figure VI-6 : Code Smil.

b. Démonstration 2 :

Dans cette deuxième démonstration nous allons faire quelques modifications temporelles, puis analyser notre document pour détecter si il y a des incohérences temporelles, puis de les corriger ; nous reprenons l'exemple précédant de l'étape 2, puis nous paramétrons les objets média comme il est illustré dans la *Figure VI-7*, avec la commande qui nous permet d'analyse le document, nous vérifions la cohérence du document puis les objets colorés en rouge doivent être paramétrer une deuxième fois.

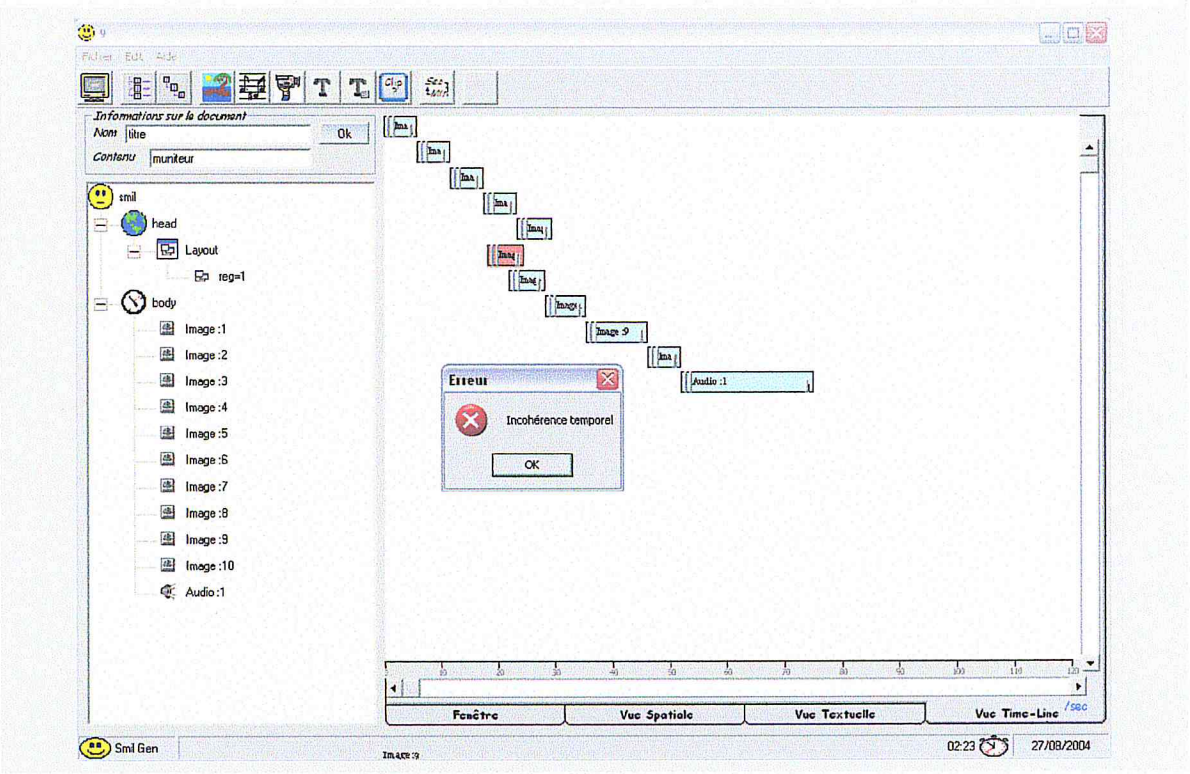


Figure VI-7 : Incohérence temporelle.

Conclusion

Nous avons présenté dans ce chapitre l'interface de notre éditeur *Smil_Gen*, ainsi que son fonctionnement à travers une démonstration d'un exemple.

Smil_Gen permet de générer le code SMIL cohérent pour un document multimédia au format SMIL. L'exécution du document SMIL peut se faire par un lecteur qui accepte le format "smi" ou "smil", Real One Player par exemple, notre éditeur ne couvre pas bien sûr la totalité de la description du langage SMIL 1.0 présentée dans le chapitre II, mais il permet la réalisation de la plus part des scénarios imaginer par un utilisateur non expérimenté, pour la réalisation d'une présentation simple.

CONCLUSION GÉNÉRALE

CONCLUSION GÉNÉRALE

L'objectif du projet présenté dans ce mémoire portait principalement sur l'étude d'un modèle temporel pour les documents multimédias au format SMIL. Notre contribution est l'élaboration d'un système que nous avons baptisé ***Smil_Gen***.

Pour la réalisation de ***Smil_Gen,*** nous avons proposé une méthodologie guidée par l'approche basée sur les STP, qui est une classe particulière des CSP binaires. L'auteur conçoit son document en utilisant le langage SMIL, ainsi le comportement temporel exprimé dans le document est traduit en un graphe temporel. Le passage par le graphe nous a permis de cerner le problème qui est la présence d'incohérences temporelles dans le document.

Bien entendu, cette réalisation ne couvre pas toute la spécification du langage SMIL présenté dans le chapitre II. En effet, il manque le traitement de certains attributs de ce langage tel que le *switch* et les éléments de lien.

Cependant, le travail réalisé a été très fructueux pour nous, vu la richesse de connaissance acquise. L'apprentissage du langage SMIL occupe une très grande partie de cette connaissance, ainsi que les techniques d'analyse des documents multimédias. L'élaboration de ce projet nous a permis, entre autre, d'ouvrir une porte sur le monde du multimédia.

Les travaux qui restent à accomplir peuvent s'orienter sur :

- Le traitement des alternatives de présentations (*switch*) qui dépendent du choix de l'utilisateur et de la configuration de son système.
- l'analyse des scénarios indéterministes, vu que notre système ne prend en charge que les scénarios déterministes.

Nous espérons, enfin, que notre travail servira de base à d'autre mémoires ou études relatives au domaine multimédia au futur.

Bibliographie et références

Bibliographie et références

[COU 98] COURTAUD D., *"SMIL (Synchronized Multimedia Integration Language) "*, DESS Ingénierie Documentaire et Multimédia, Module M4A, 1998.

[COU 96] COURTIAT J.P., DE OLIVERA R. C., *"Proving Temporal Consistency in a New Multimedia Synchronisation Model"*, Proceedings of the Fourth ACM Conference on Multimedia, ACM Press, Boston, novembre 1996.

[COU 00] COURTIAT J.P., SANTOS C.A.S, LOHR C, OUTTAJ B., *"Exprience with RT-LOTOS, a temporel extension of the LOTOS formal description technique"*, july 2000.

[COU 03] COURTIAT JP., *"Conception formelle de documents multimédia interactifs : une approche s'appuyant sur RT-LOTOS"*, rapport LAAS No03189, doctorat, Université Paul Sabatier, Toulouse, 18 Avril 2003, 172p.

[DAN 62] DANTZIG G. B., *"Linear Programming and Extensions"*, Princeton University Press, Princeton, NJ, 1962.

[DEC 91] DECHTER R., MEIRI I. et PEARL J., *"Temporal Constraint Networks"*, Artificial Intelligence, 49, pp. 61-95, 1991.

[DHT 98] DHTML, *"Dynamic HTML"*, disponible à : http://www.dhtml-zone.com/

[DIS 98] W3C DISCUSSION Services, *"Public lists SMIL Mail Archives"*, disponible à : http://Lists.w3.org/Archives/Public/www-smil.

[ENC 02] Encyclopédie Microsoft Encarta 2002.

[HAR 93] HARDMAN L., VAN ROSUSUM G. et BULTERMAN D.C.A., *"Structured Multimedia Authoring"*, Proceedings of the first ACM International Conference on Multimedia, 99. 283-289, Anaheim, Août 1993.

[ISO 89] INTERNATIONAL STANDARD ORGANIZATION, *"International Standard ISO/IEC 8613: Information Processing - Text and Office Systems - Office Document Architecture (ODA) and Interchange Format"*, Genève, New York, 1989.

[JMF 00] Sun Microsystems, *"The Java Media Framework 2.0"*, disponible à: http://java.sun.com/products/java-media/jmf/index.html.

[KIM 95] KIM M. et Song J., *"Multimedia Documents with elastic time"*, Proceedings of the third ACM International conference on Multimedia, édité par ACM Press, pp. 143-154, Polle Zellweger, San Francisco, novembre 1995.

[LAY 97] LAYAÏDA N., *" Madeus : Système d'édition et de présentation de documents structurés multimédia"*, Thèse de Doctorat, Université Joseph Fourier, Grenoble, France, 1997.

[LAY 99] LAYAÏDA N., *"Adaptabilité : Pistes d'étude pour la définition d'une infrastructure d'accès au contenu multimédia pour des machines hétérogènes"*, Rapport de contrat, INRIA Rhône-Alpes, Octobre 1999.

[LEI 93] LEISERSON C. E., SAXE J. B., *"A mixed-integer linear programming problem which is efficiently solvable"*, Proceedings of the 21st Annual Allerton Conference on Communication, Control, and Computing, pp. 204-213, 1983.

[LIA 93] LIAO Y. Z., WONG C. K., *"An Algorithm to compact a vlsi symbolic layout with mixed constraints"*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. CAD, num. 2, pp. 62-69, 1993.

[MAC-DIR 00] MACROMEDIA Director, *"Using Director 8"*, Manuel de référence, disponible à : http://www.macromedia.com/software/director,2000.

[MAC-FLA 00] Macromedia, *"Flash"*, disponible à : http://www.macromedia.com/software/flash/

[MAC-LIN 00] MACROMEDIA Lingo, *"Learning Lingo"*, Dictionnaire Lingo, disponible à : http://www. macromedia.com/software/lingo,2000.

[MHE 93] ISO/IEC. JTC 1/SC 29, *"Codec Representation of Multimedia and Hypermedia Information Objects (MHEG)"*, Part 1, Comitee Draft 13522-&, juin 1993.

[MHM 98] Alcatel, "MHML : Langage de description de document multimédia", rapport interne, 1998.

[PAP 82] PAPADIMITRIOU C. H., STEIGLITZ K., *"Combinatorial Optimization: Algorithms and Complexity"*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[REA 98] REAL Networks, *"Real Networks G2,* disponible à : http://www.real.com/g2/index.html.

[REA 99] REAL Networks, *"Smil Wizard in G2Authoring Kit"*, disponible à : http://www.real.com/products/tools/authkit/index.html.

[SAM 03] MAIA SAMPAIO P.N., *"Conception formelle des documents multimédias interactifs : Une approche s'appuyant sur RT-LOTOS"*, thèse de doctorat, Université Paul Sabatier Toulouse, Avril 2003.

[SHO 81] SHOSTAK R., *"Deciding linear inequalities by computing loop residues"*, *JACM,* num. 28, pp. 769-779, 1981.

[SMI 98] W3C, *"Synchronized Multimedia Integration Language (SMIL) 1.0"*, P. Hoschka. W3C Recommandation, 15 juin 1998, disponible à : http://www.w3.org/TR/REC-smil.

[SMI 00] *"Just smil"*, 2000 disponible à : http://www.allhtml.com/smil/justsmil.com.

[SMI-20] W3C, *"Spécification du langage d'intégration multimédias synchronisés (SMIL 2.0) "*, recommandation du W3C, disponible à : http://www.w3.org/TR/smil20/.

[TAR 97] TARDIF L., *"Visualisation du scénario temporel d'un document multimédia"*, Rapport de DEA Informatique, INPG-Universite Joseph Fourrier, juin 1997.

[TEC 98] Muriel J., Layaïda N., Roisin C., *"Le temps dans les documents"*, disponible à : http://opera.inrialpes.fr/people/Nabil.Layaida/publi/techIng98.pdf.

[WIR 95] WIRAG S., WAHL T. et ROTHERMEL K., *"Tiempo: An Authoring and Presentation System for Interactive Multimedia"*, Fakultätsbericht 5/95, rapport technique, Université d'informatique, Université de Stuttgart, Germany, 1995, disponible à : http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/tiempo.engl.html.

[W3C 98] W3C, *"Recommendation Synchronized Multimedia Integration Language (SMIL) 1.0 Specification"*, 15 June 1998, disponible à : http://www.w3.org/TR/REC-smil.

[XML 98] BRAY T., PAOLI J. et SPERBERG-MQ C.M., MALER E , *"Le langage de balisage extensible (XML) 1.0 - Seconde édition"*, Recommandation du W3C, 10 février 1998,
disponible à : http://www.w3.org/TR/REC-XML.

[XML 99] W3C, *"Extensible Markup Language (XML)"*, disponible à : http://www.W3C.org/XML/

NB: Vu les mises à jour effectuées sur les listes d'Internet, leur contenu risque de changer.