Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Les aspects temporels dans la présentation multimédia implémentation de Madeus en Java

Kayitana, Alain

Award date:
1999

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 02. May. 2026

Les aspects Temporels dans la présentation multimédia :

Implèmantation de Madeus en JAVA.

Mémoire réalisé par Alain Kayitana

en vus de l'obtention du diplôme de « Licence en Informatique ».

Promoteur : Monsieur Jean Paul **Leclercq**

UBS 8333607

A Vous Eloi, Laïssa, Juliette et Jeanne,
pour tous les efforts consentis et pour vos encouragements,
je dédie ce travail.

Remerciements

Au terme de ce travail, je tiens à exprimer ma profonde gratitude à Monsieur Laurent Kempenners pour ses conseils et sa grande disponibilité malgré ses nombreuses occupations. Je remercie également tous ceux qui de près ou de loin ont contribué à la réalisation de ce travail.

Table des matières

TABLE DES MATIÈRES 1

INTRODUCTION 3

CHAPITRE 1. LA PRÉSENTATION MULTIMÉDIA TEMPORELLE. 5

1. NOTIONS PRÉLIMINAIRES. 6

1.1 NOTION DE SYSTÈME MULTIMÉDIA. 6
1.2 NOTION D'UNITÉS DE PRÉSENTATION. 7
1.3 NOTION DE SYNCHRONISATION TEMPOREL. 7
1.4 NOTION DE SCÉNARIO TEMPOREL 10

2. MODÈLES DES DOCUMENTS MULTIMÉDIAS. 11

2.1 DÉFINITION. 11
2.2 RELATIONS INTER-MÉDIAS D'UN DOCUMENT 11
2.3 EDITION MULTIMÉDIA. 13

3. MODÈLES TEMPORELS. 17

3.1 DÉFINITION 17
3.2 REPRÉSENTATION DE L'INFORMATION TEMPORELLE 17
3.3 EXPRESSIONS DES RELATIONS TEMPORELLES 19
CONCLUSION. 20

CHAPITRE 2. LE LANGAGE SMIL 21

2.1 ORIGINE DE SMIL 22
2.2 DESCRIPTION DE LA SYNTAXE SMIL 24
2.2.1 Structure générale *25*
2.2.2 L'en-tête d'un document SMIL *25*
2.2.3 Le corps d'un document SMIL *28*
2.3 MÉCANISME D'ÉVÉNEMENT 35
2.4 NOTIONS AVANCÉES DE SMIL 36
2.4.1 Le contenu alternatif *36*
2.4.2 Les liens *37*
3. APPORT DE SMIL PAR RAPPORT AUX PRÉSENTATIONS MULTIMÉDIAS TEMPORELLES 39
3.1 SMIL PAR RAPPORT AUX TYPES DE MÉDIAS 39
3.2 SMIL PAR RAPPORT À LA SYNCHRONISATION TEMPORELLE. 39
3.3 SMIL PAR RAPPORT AUX SCÉNARIOS TEMPORELS 39
3.4 SMIL PAR RAPPORT AUX MODÈLES DE DOCUMENTS 39
3.5 SMIL PAR RAPPORT À L'ORGANISATION D'UN DOCUMENT 40
3.6 SMIL PAR RAPPORTS AUX TYPES D'ÉDITION 40

CHAPITRE 3. ETAT DE L'ART DES OUTILS 41

3.1 REALSYSTEM G2 (HTTP://WWW.REAL.COM) 42
3.1.1 RealPlayer *42*
3.1.2 RealSystem™ G2 Production *43*
3.2. CWI(CENTROUM VOOR WISKUNDE EN INFORMATICA HTTP://WWW.CWI.NL/GRINS/) 45
3.2.1. GRINS Player *45*
3.2.1. GRINS Editor *46*
3.3. SOJA HELIO (HTTP://WWW.HELIO.ORG) 48
3.4. S2M2(HTTP:/ /SMIL.NIST.GOV/PLAYER/S2M2.HTML) 48
3.5. LPPLAYER(HTTP://WWW.LABYRINTEN.SE) 49
3.6 EVALUATIONS 49

CHAPITRE 4. ARCHITECTURE ET CONCEPTION DE L'APPLICATION 50

4.1 ANALYSE DES BESOINS 51

4.2 ARCHITECTURE DE L'APPLICATION. 51
4.3 DÉCOUPE EN PACKAGES 54
4.3.1 Package serveur *55*
4.3.2 Package traitementListe *59*
4.3.3 Package graphe *62*
4.3.4 Package traitementFichier *77*
4.3.5 Package coordinateur *78*
4.3.6 Classe SmilApplication *80*
4.3.7 Réseau d'échanges. *81*
4.4 CODAGE 82
A. JDK 1.1.8 (JAVA Development Kit) *82*
B. JMF(JAVA Média Framework) *82*
C. JFC(JAVA Fondation Class) *83*
D. JFlex *83*
E. CUP *84*

CONCLUSION GÉNÉRALE **85**

Introduction

Actuellement le terme «multimédia » est à la mode, mais nous ne savons pas ce que multimédia veut dire exactement. Certains essaient de définir le multimédia comme un ensemble de techniques qui permettent de manipuler les informations d'origine diverses comme l'image, le texte, le son ou la vidéo. En analysant cette définition, nous nous rendons compte qu'elle ne suffit pas. Si ce n'était que l'intégration de plusieurs médias, nous n'aurons pas besoin des compétences particulières.

Nous pouvons étendre la définition du multimédia en insistant sur la manière dont ces médias sont mis ensemble et c'est cela qui fait du multimédia une discipline à part entier. Le multimédia est donc à la fois le contenu et la technologie mise en œuvre.

Nous parlerons du multimédia, quand les critères suivants sont remplis :

- quand différents éléments informationnels d'origines diverses sont mis ensemble à la disposition de l'utilisateur,
- quand le produit final ou le service qui en résulte forme un tout original,
- quand les informations présentées sont structurées suivant un lien logique pour permettre à l'utilisateur la possibilité de dialoguer ou se déplacer dans le produit ou le service de manière interactive,
- enfin, quand le tout est mis à la disposition de l'utilisateur en utilisant la technologie et la puissance des machines actuelles.

Ce dernier point explique pourquoi le multimédia ne s'est pas développé un peu plutôt. La technologie n'était pas encore disponible.
Nous pouvons nous poser aussi la question de savoir comment le multimédia est utilisé aujourd'hui. Le multimédia est rangé en deux catégories. La première catégorie utilise les supports classiques telles que les disques compacts(CD-Rom, CD-I, DVD. Dans ce cas nous parlerons de multimédia «*hors ligne* ». La deuxième catégorie est l'utilisation des réseaux de télécommunication, des réseaux informatiques tels qu'Internet. Cette approche permet un dialogue entre l'utilisateur et la source. Dans ce cas nous parlerons alors de multimédia «*en ligne* ».

Une autre question qu'on peut se poser est l'origine du multimédia. Nous pouvons répondre en disant que le multimédia est le résultat de la convergence de plusieurs domaines telles que l'informatique, les télécommunications, l'audiovisuel et les arts graphiques. C'est normal car le multimédia est base sur la communication et il a profité de tout ce qui existe en terme de médias et de technologie de la

communication. Le point important est que cela a été possible grâce aux progrès technologiques. Ces progrès sont la numérisation et la compression des données. La numérisation permet de ressembler et de faire interagir dans un seul produit ou service des médias aussi divers que le son, l'image, le texte, la vidéo ou autres.

La dernière question qu'on peut se pose est l'avenir du multimédia. Le multimédia a permis des liens entre différents domaines, qui au départ n'interagissaient pas. Cela a donné naissance à une série de métiers tels qu'éditeurs, auteurs, créateurs, graphistes, réalisateurs etc.. Le multimédia est donc porteur de beaucoup d'espoir pour tous. Les applications multimédias sont donc nombreuses et variées.

Cette petite introduction sur le multimédia nous montre l'importance croissante de ce domaine et nous nous proposons d'étudier certains aspects du multimédia dans ce travail de fin d'études.

Dans le premier chapitre nous allons découvrir ce qu'est une présentation multimédia, ce qu'est un scénario temporel, certains méthodes d'éditions des documents multimédias et les notions connexes telles que la synchronisation entre médias.

Dans le deuxième chapitre, nous allons décrire un langage d'édition(SMIL) d'une présentation multimédia et son apport par rapport aux aspects du multimédia vus au premier chapitre.

Dans le troisième chapitre nous verrons l'état d'art de ce qui se fait dans l'édition et la présentation des documents multimédias et la comparaison de ces différents outils.

Dans le dernier chapitre, nous décrirons l'architecture et la conception d'une application de visualisation d'un document multimédia. Cette application sera implémentée en JAVA.

Chapitre 1. La Présentation Multimédia Temporelle.

Le but de ce chapitre est d'introduire les notions de base sur les systèmes multimédias. Nous insisterons sur le problème de synchronisation des objets multimédias. Nous décrirons la nature des problèmes liés à la synchronisation temporelle et son apport dans l'édition et la présentation des documents multimédias. Nous discuterons aussi des différentes structures des documents multimédias. Pour de plus amples explications, nous recommandons au lecteur de se référer à la thèse de doctorat de Nabil Layaida[1].

[1] http://opera.inrialpes.fr/Infos/Personnes/Nabil.Layaida/

1. Notions préliminaires.

1.1 Notion de système multimédia.

La notion de «système multimédia » ou simplement «d'application multimédia » n'est pas facile à définir. Il existe dans la littérature plusieurs critères pour définir un système multimédia.

Le premier critère est le nombre de médias manipulés. Ce critère nous renseigne sur la capacité d'une application à manipuler les différents médias comme l'audio, la vidéo, l'image, le texte. Mais à lui seul, ce critère n'est pas suffisant. Par exemple une application telle que Office 97 qui intègre un traitement de texte, un éditeur de graphiques, peut être considérée comme un système multimédia.

Le second critère est la nature temporelle des médias manipulés. Il y a des médias dépendants du temps(*médias temporisés*) comme l'audio et la vidéo et des médias indépendants du temps(*médias non temporisés*) comme l'image et le texte. Ce critère est aussi insuffisant. Par exemple l'encyclopédie sur CD-Rom qui manipule le texte et l'image avec des annotations sonores, peut être considéré comme un système multimédia, mais elle n'a pas un traitement intégré pour les médias temporisés et non temporisés.

Le dernier critère est le niveau d'intégration de différents médias au sein de l'application. Ce critère nous renseigne sur la capacité d'une application à traiter de façon homogène plusieurs médias de nature différents. Le traitement des médias porte sur la manière de faire des modifications, des mises à jour et de stockage de différents médias.

Nous pouvons dire qu'un bon système multimédia doit au moins remplir les critères cités ci dessus, c'est à dire :
- le nombre de médias que manipule cette application,
- la nature temporelle des médias,
- le niveau d'intégration des différents médias.

Le schéma suivant nous montre les différents types d'applications par rapport au qualifiant "multimédia" selon les trois critères.

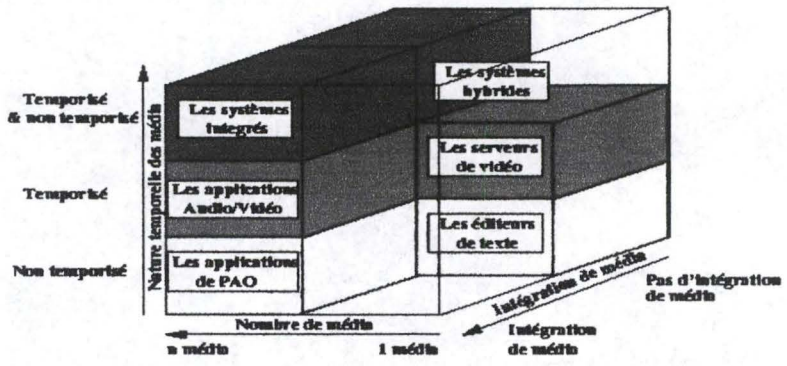


Figure 1 : Exemples d'applications

Un système multimédia peut avoir plusieurs fonctionnalités. Les principaux fonctionnalités sont :

- l'édition qui réalise les opérations de création, de modification, de composition des documents multimédias,
- la présentation qui permet de visualiser ou restituer à un utilisateur le contenu d'un document multimédia. Les opérations fournis à l'utilisateur permettent de naviguer à travers le document dans l'espace et dans le temps.

1.2 Notion d'unités de présentation.

La notion d'unités de présentation est intimement liée aux médias temporisés. Ces médias sont formés d'une suite d'unités plus petites de présentation, appelées ***unités d'information.*** Ces unités d'information sont soumises à des contraintes temporelles. Par exemple une vidéo est composée d'une suite de scènes, chacune correspond à un passage particulier de la vidéo. Ces scènes peuvent à leur tour être découpées en sous scènes. Les sous scènes peuvent être composées d'une succession d'images. Le fait d'isoler ou d'identifier les unités d'informations pour un média est important. En effet on peut utiliser la succession de ces unités pour réduire la taille des données d'un média. L'exemple courant est la compression des images en exploitant les unités d'information redondantes.

1.3 Notion de synchronisation temporelle

La notion de synchronisation est difficile à définir. Intuitivement le terme synchronisation fait référence au temps et c'est ce lien avec la dimension temporelle qui rend la synchronisation difficile à expliquer. L'autre problème qui rend la synchronisation difficile à expliquer, est que le domaine du multimédia est récent. Les connaissances qui s'y rapportent ne sont pas encore bien maîtrisées. Nous avons vu que le traitement intégré des médias dans un système multimédia est l'un des critères important, mais avant de faire ce traitement il faut d'abord prendre en considération les médias temporisés et les médias non temporisés, où le facteur temps ne joue le même rôle.
Cette différence de rôle du temps nous montre deux aspects de synchronisation :

- le support des médias temporises dans une application,
- l'intégration des traitements de l'ensemble de ces médias.

Ces deux aspects nous permettent de comprendre le problème de synchronisation car ils traitent les médias temporisés, leur intégration, leur combinaison avec des médias non temporisés.

Il existe plusieurs types de synchronisation. Ces types de synchronisation permettent de comprendre la nature des relations entre les différents médias et d'isoler les mécanismes de résolution de chacune d'entre elles. Certains types de synchronisation sont effectués au niveau du média alors que les autres sont effectués entre les différents médias.

A. Synchronisation intra-élément.

La synchronisation intra-élément se base sur les relations temporelles entre les unités d'information d'un même média. L'exemple typique est la relation entre les images d'une séquence vidéo. Si la présentation affiche 25 images par seconde, chaque image est affichée pendant 40 millisecondes(Figure 2 a).

B. Synchronisation inter-élément.

La synchronisation inter-élément se base sur les relations temporelles de plusieurs médias. Un exemple de ce type de synchronisation est une présentation simultanée d'une vidéo, d'un audio et d'un texte(Figure2 b).

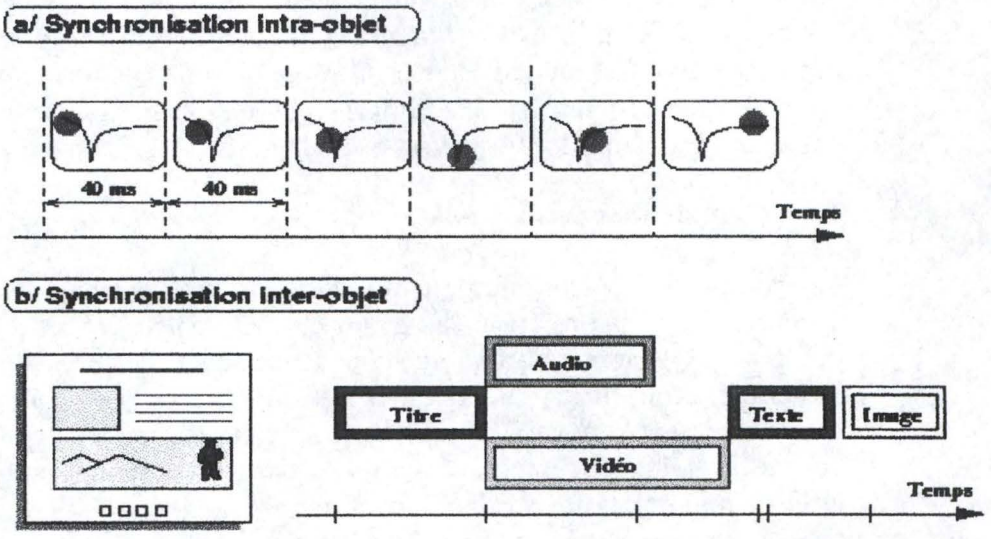


Figure 2 : Synchronisation inter et intra élément.

C. Synchronisation des lèvres.

La synchronisation des lèvres est une combinaison des deux synchronisations citées ci-dessus. Il impose un couplage temporel entre la progression des flux de plusieurs médias. Ce couplage exprime le décalage temporel entre les flux de ces médias. L'exemple courant est la présentation simultanée d'une séquence audio et d'une séquence vidéo associée. Il faut que le son de la séquence audio soit en accord avec le mouvement des lèvres dans la séquence vidéo (Figure 3).

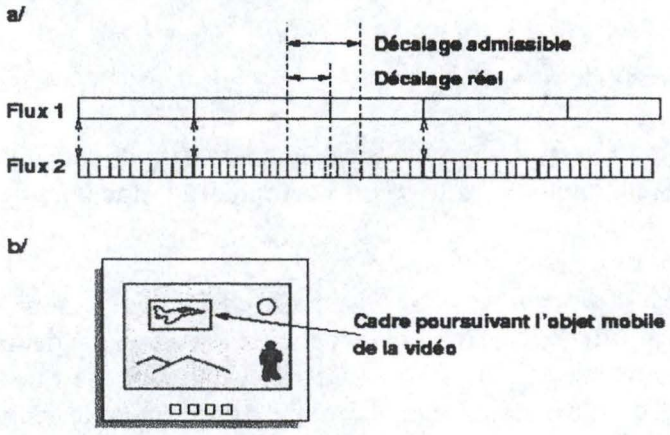


Figure 3 : Synchronisation des lèvres

La compréhension de ces différents types de synchronisations est capitale. Elle permet de cerner la nature des relations entre médias et d'identifier les mécanismes adaptés pour la résolution de chacune d'entre elles.

En ce qui concerne les applications multimédias, la synchronisation présente deux types d'aspects selon les contraintes temporelles appliquées aux différents médias :

- le premier type d'aspect se réfère aux contraintes de temps du monde réel. Dans ce cas nous parlerons de «*synchronisation naturelle*»,
- le second type d'aspect se réfère aux contraintes du constructeur de l'application. Dans ce cas nous parlerons de «*synchronisation artificielle*». Les contraintes temporelles ne respectent pas nécessairement les contraintes naturelles des médias.

Dans le cas de la synchronisation naturelle, les contraintes temporelles sont intrinsèques aux médias considérés et font généralement partie de la présentation de ces médias. Par exemple pour un média temporisé, les contraintes traduisent les conditions de reproduction des effets temporels liés au média lors de sa numérisation(débit fixe) et font partie de la représentation de ce média.
Le traitement de la synchronisation naturelle ressemble fort au modèle de type producteur/consommateur qui fait intervenir trois composantes. Le premier composant est la «source » qui permet de numériser des données du média. Le second composant est la «connexion » qui permet le transfert des données numérisées vers le dernier composant qui est la «destination ». La destination permet de visualiser ou restituer les données du média. L'illustration de la synchronisation naturelle est la transmission d'une vidéo depuis un serveur sur une machine distante vers une application utilisatrice. Cette application doit respecte les contraintes temporelles de la source.

Dans le cas de la synchronisation artificielle, les contraintes correspondent aux conditions de présentation dans l'application. Il s'agit d'une opération pour regrouper les différents médias afin de produire une nouvelle présentation multimédia plus complexe. L'illustration de la synchronisation artificielle est la création par exemple

d'animations. L'utilisateur doit spécifier une durée artificielle de l'animation avant son utilisation.

1.4 Notion de scénario temporel.

Un scénario temporel définit comment les éléments d'un document s'enchaînent dans le temps. C'est en quelque sorte la projection d'un document sur la dimension temporelle.
Il existe deux types de scénario temporel :

- les scénarios « *déterministes* » où les enchaînements sont définis avant la présentation d'un document. Le calcul des instants de début et de fin pour différents médias se fait de façon statique. Cela suppose que l'on connaisse la durée exacte de ces médias. Donc il n'existe qu'un seul trace d'exécution du scénario.
- les scénarios « *indéterministes* » où les enchaînements sont définis en fonction des données connues au moment de la présentation. Un scénario indéterministe peut avoir un ensemble de traces qui lui correspondent(Figure 4).

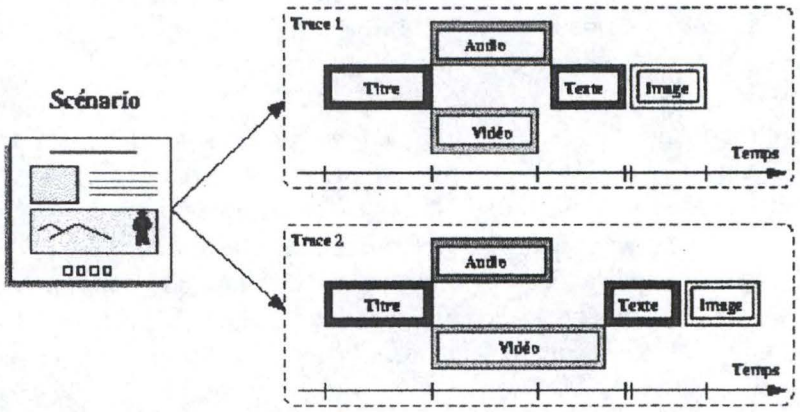


Figure 4 : Exemple de scénario temporel indéterministe.

Sur cet exemple, la non connaissance du temps de présentation de la vidéo et de l'audio situé sur un site distant, implique qu'il peut exister plusieurs traces d'exécution.

2. Modèles des documents multimédias.

2.1 Définition.

Dans la section précédente, nous avons vu qu'un système multimédia doit offrir les fonctionnalités d'édition et de présentation d'un document multimédia. Ces fonctionnalités sont dépendantes du type d'un document multimédia.
Avant de donner la définition d'un modèle de document multimédia, nous allons décrire certains propriétés qui sont souhaitables pour faciliter l'édition et la présentation d'un document multimédia.

- il faut garantir l'indépendance de la représentation par rapport à l'édition d'un document multimédia. Pour cela le document doit être structuré. Cela veut dire qu'il faut séparer les aspects liés à l'organisation logique et les aspects liés à sa présentation,
- il faut que le processus d'édition soit incrémental. Cela veut dire que chaque opération d'édition doit permettre de voir le résultat à la présentation,
- il faut maintenir l'intégrité du document. Chaque opération d'édition effectue par l'utilisateur, le système multimédia doit vérifier la cohérence du document,
- il faut rendre homogène le traitement des différents médias dans un document multimédia,
- enfin, il faut garantir la résolution automatique des contraintes temporelles afin de permettre à l'utilisateur des opérations de synchronisation.

Ces différentes propriétés permettent de construire des systèmes multimédias interactifs, une représentation des documents multimédias riches au point de vue sémantique et des documents interchangeables entre applications. C'est cette représentation qu'on appelle «modèle de document multimédia ».

2.2 Relations inter-médias d'un document.

Le modèle d'un document permet de représenter toutes les relations qui existent entre les éléments du document multimédia. Ces relations sont appelées des «relations multimédias » et portent sur :

- l'organisation logique,
- l'organisation spatiale,
- l'organisation temporelle,
- l'organisation hypermédia.

A. Organisation logique.

L'organisation logique exprime le regroupement des éléments qui sont liés au niveau de la sémantique. Ce regroupement donne naissance à un élément qui constitue une nouvelle entité. Plusieurs entités peuvent à leur tour être groupées pour former d'autres entités logiques plus complexes. Prenons l'exemple d'un travail de fin d'études.
L'organisation logique peut être la suivante :

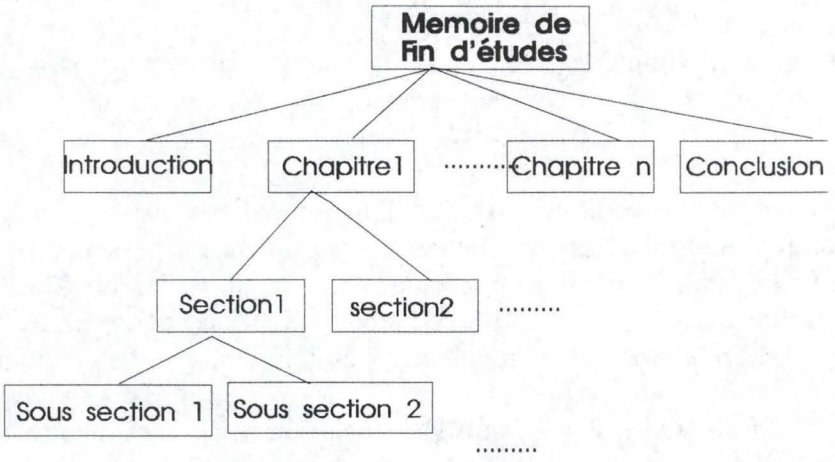


Figure 5 : Structure logique d'un document

Ce type de document décrit des relations d'inclusions entre les éléments et définit une classe de documents.
Nous remarquons qu'on peut éditer facilement une grammaire de ce type de document à l'aide de l'opération «Liste de » :

Mémoire de fin d'études :
Introduction,
Liste de Chapitres,
Conclusion.
Chapitre :
Liste de Sections.
Section :
Liste de Sous Section.
.....

B. Organisation spatiale.

L'organisation spatiale concerne la disposition des éléments selon différents canaux à travers le temps. Par exemple si on présente une vidéo, il faut disposer un espace géométrique sur l'écran pendant toute la durée de la vidéo. Cette opération s'appelle le *formatage spatio-temporel*. Ce type d'opération permet de définir des règles génériques de présentation du document et le principal avantage est la réduction du nombre de spécifications pour la construction d'un document.

C. Organisation temporelle.

L'organisation temporelle concerne la disposition des éléments dans le temps. Les médias d'un document sont reliés temporellement et définissent un ordre de présentation. Par exemple un document multimédia qui comprend une animation d'images, il faut préciser un ordre de présentation des images. La synchronisation des images au moyen des relations temporelles permet de construire la structure temporelle de ce document.

D. Organisation hypermédia.

L'organisation hypermédia permet de décrire les relations entre les éléments d'un document et entre documents différents. Ces relations sont des renvois ou des références et sont utilisées pour naviguer entre différents documents ou entre les parties d'un document.
L'exemple courant est un document HTML[2]. Un lien est défini par une balise de départ et une balise d'arrivée.

2.3 Edition multimédia.

Il existe différentes approches dans l'édition des documents multimédias. Dans chaque approche, il y a des opérations spécifiques offertes à l'utilisateur, ses avantages et ses inconvénients.

A. L'édition basée sur les axes de temps ou "timelines"

L'édition basée sur les axes de temps d'un document est décrite par un flot de données par rapport à un axe de temps. C'est cet axe qui représente la dimension temporelle du document. C'est le type d'édition le plus utilisé dans les systèmes multimédias commerciales.L'activité d'édition consiste à placer des icônes qui représente les médias sur l'axe du temps en précisant l'instant de début et l'instant de fin par rapport au début du document. Les activités concurrentes sont représentées au moyen de plusieurs axes du temps.

---

[2] HTML :langage pour l'édition des documents sur Word Wide Web.

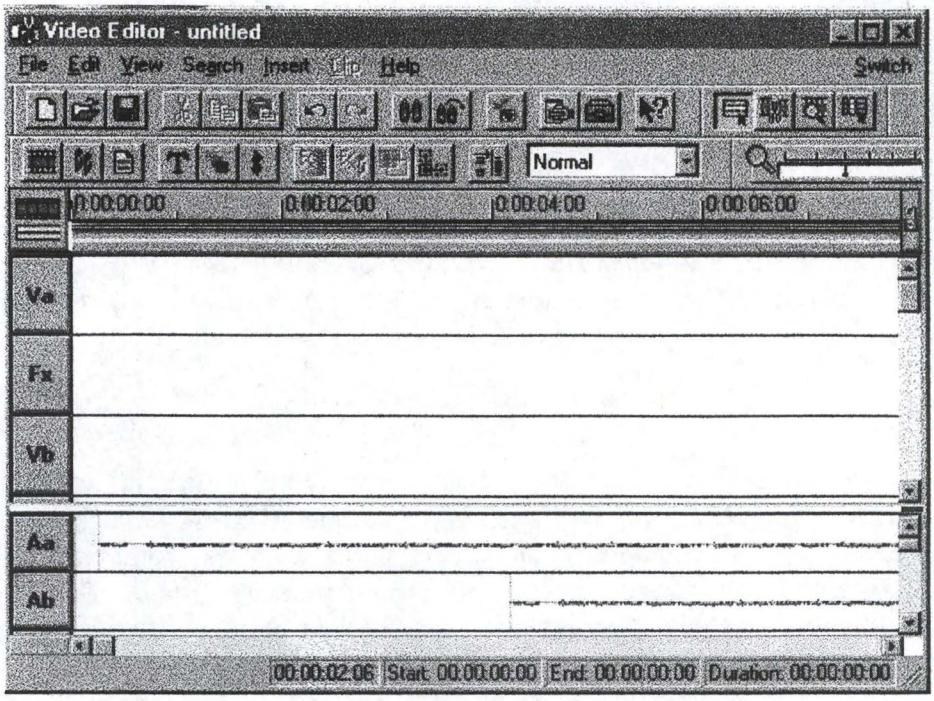


Figure 6 : Edition basée sur les timelines. Interface de Média Studio.

Les éditeurs à base des « timelines » sont bien adaptés aux présentations multimédia car leur interface rend bien compte, de façon intuitive le placement temporel des différents événements de la présentation.
Toutefois les inconvénients sont multiples :

- la modification de durée(c'est à dire la date de début ou la date de fin) nécessite de repositionner les autres éléments sur l'axe du temps,
- la datation explicite des différents événements de présentation rend impossible la représentation des éléments dont la durée est imprévisible,
- la représentation(l'organisation) logique est presque inexistante ce qui empêche la réutilisation du document dans la plupart des cas.

B. Edition basée sur les graphes.

L'édition basée sur les graphes permet de représenter un document sous la forme d'un diagramme de flot de contrôle et décrit l'interaction entre différents éléments à l'exécution. Sur le graphe d'exécution, on précise l'instant de début et l'instant de fin d'un élément par rapport au début ou à la fin de l'élément précédant ou suivant dans le scénario temporel document. Cela permet de se passer de la connaissance des instants de début et de fins absolues des différents éléments.
Par exemple, si on a un document qui a cinq médias :

- une introduction sonore de 5 secondes,
- une vidéo de 10 secondes,
- une image qu'on doit affiche 6 secondes,
- un texte qui accompagne l'image de 6 secondes,
- un conclusion sonore de 15 secondes.

Nous voulons un scénario temporel qui commence par l'introduction puis joue la vidéo suivi de l'affichage simultané de l'image et du texte et se termine par la conclusion.

Les graphes fournissent un support mieux adapté pour les opérations d'insertion, de suppression des médias dans un scénario temporel. Comme on ne précise pas l'instant de début et l'instant de fin par rapport au début d'un document.

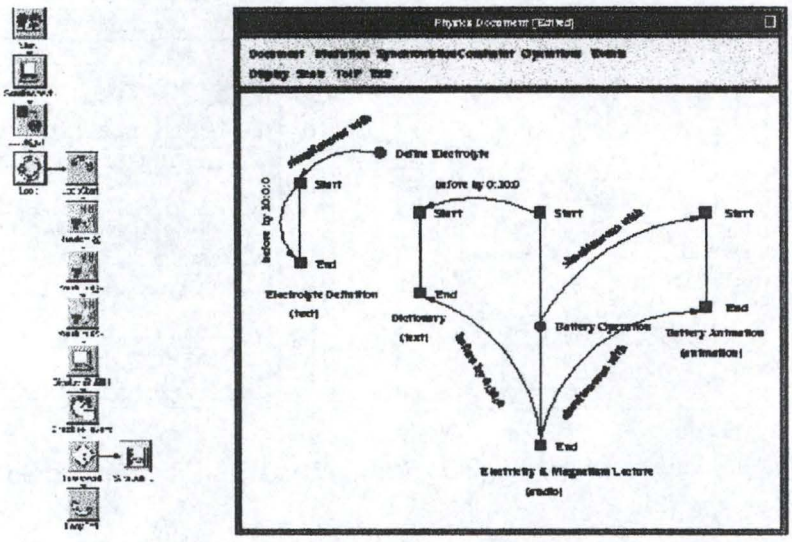


Figure 7. Edition basée sur les graphes : Interface de Firefly

Les relations de synchronisations sont mieux exprimées et il est possible de représenter beaucoup plus de combinaisons de scénarios possibles que dans le cas des timelines.

Les inconvénients majeurs sont que :

- la description temporelle devient rapidement complexe et peu lisible. Cela est du à une représentation des événements de bas niveau du scénario et que chaque média fait intervenir beaucoup d'événements temporels (son début, sa fin et ses événements internes),
- le document est peu structuré selon la dimension temporelle, ce qui empêche la réutilisation de ses parties.

Ces deux types d'éditions définissent un document multimédia en utilisant des primitives de synchronisation temporelles qu'on applique sur les médias.

C. Edition basée sur la structure

L'édition fondée sur la structure exploite l'organisation logique du document pour permettre sa synchronisation temporelle. Cette approche permet d'organiser le contenue d'un document en modules isolés sur lesquelles on peut appliquer les méthodes de synchronisation. Ces méthodes sont la mise en parallèle ou en séquence des éléments appartenant à une même entité logique.
L'outil commercial qui exploite ce type d'édition est le CWI Multimédia Interchange Format Editor. Cet outil permet de voir le document soit d'une vue structurelle(hiérarchie d'un document sous forme de boites encapsulées) ou soit d'une vue temporelle(vue canaux).

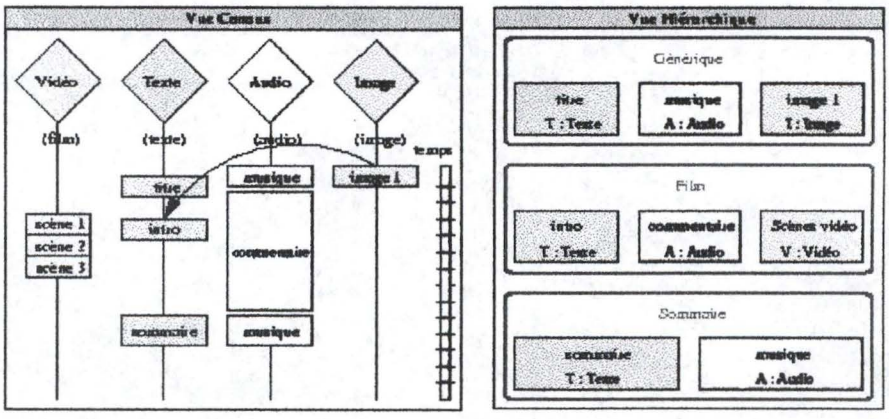


Figure 8 : Edition basée sur la structure.Interface de CMFed).

D. Edition basée sur la programmation.

L'édition basée sur la programmation sert pour pallier aux limités d'autres types d'édition. Cette approche permet une description plus fine des primitives de synchronisation au moyen d'un langage de programmation. Ce langage permet de décrire les relations temporelles et spatiales de façon plus élaborée. L'inconvénient de ce type d'édition est le changement du style d'édition de documents. La description d'un document se transforme en activité procédurale que déclarative. En plus la programmation n'est pas à la portée d'un auteur non-spécialiste.

3. Modèles temporels.

Dans les sections précédentes, nous avons passé en revue certains aspects des problèmes liés à la synchronisation dans les systèmes multimédias. Nous avons vu les différentes organisations d'un document multimédia notamment l'organisation temporelle. Dans la suite nous allons décrire la modélisation de la structure temporelle d'un document.

3.1 Définition

Un modèle temporel pour l 'édition et la présentation multimédia est composé :
- d'un langage temporel qui permet la spécification d'un scénario temporel,
- de mécanismes d'analyse pour prévenir les incohérences et de les détecter à l'édition par exemple,
- enfin, de mécanismes de synthèse qui permettent de produire une présentation conforme aux spécifications.

3.2 Représentation de l'information temporelle

Tout élément multimédia peut être manipulé à travers les informations temporelles suivantes :
- son instant de début,
- sa durée de présentation,
- son instant de fin.

Il est possible d'obtenir l'une en fonction des deux autres car une des trois informations est redondante. En effet, il est possible de calculer l'une en fonction de deux autres.

Il existe deux façons de représenter un scénario à travers les changements qui surviennent(exemple : démarrage ou terminaison d'un média) ou en reliant les activités entre elles(exemple : une séquence audio est représentée pendant la séquence vidéo).

A. La représentation basée sur les instants

La représentation basée sur les instants permet de présenter un média par son instant de début et son instant de fin.

B. La représentation basée sur les intervalles

La représentation basée sur les intervalles permet de présenter un média comme une entité temporelle décrite par sa durée. Les unités temporelles de base peuvent être classées alors, dans trois catégories en fonctions des caractéristiques attachées à leurs durées :
- les *intervalles discrets*. Ce sont des unités dont la présentation ne dépend pas du temps. Ces unités peuvent être affectées d'une durée de présentation explicite ou implicite dans un scénario donné. Le texte et l'image fixe sont de telles unités temporelles,

- les *intervalles déterministes*. Ces unités dépendent du temps et leurs valeurs sont connus. La vidéo et l'audio sont des exemples de telles unités temporelles.
- les *intervalles indéterministes*. Ces unités n'ont pas de durée connue à priori.

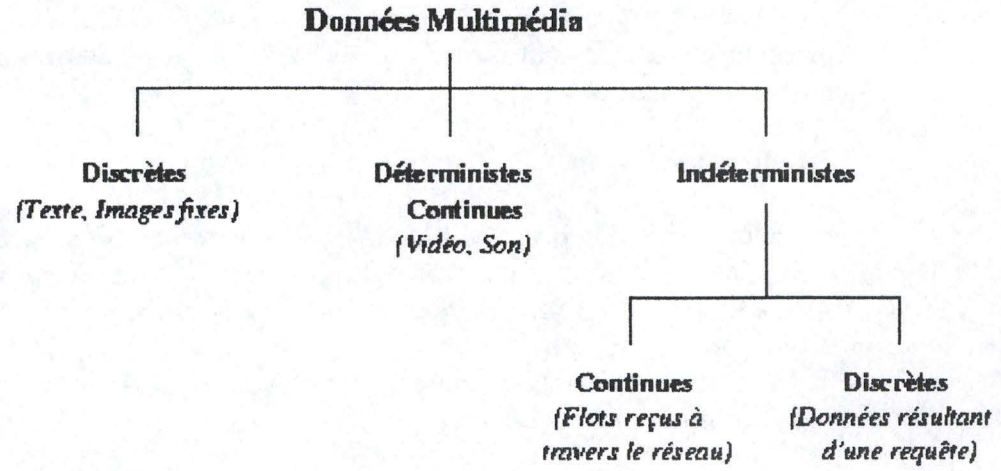


Figure 9. Les unités temporelles de base.

Le domaine de validité des unités de présentation à base d'intervalles est modélisé par un ensemble de valeurs possibles de durée de présentation. Ce domaine peut être défini par un triplet de valeurs :

- la durée minimale «**min**» qui représente la borne inférieure pour la durée de présentation d'un élément multimédia(la date correspondant à cette durée est la date au plut tôt) ,
- la durée nominale «**nom**» qui correspond aux contraintes de synchronisation intrinsèques d'un élément multimédia ,
- la durée maximale «**max**» qui représente la borne supérieur pour la durée de présentation(la date correspondant à cette durée est la date au plus tard).

Le comportement temporel de tous les éléments multimédias peut être représenté par ces domaines de validité. Si on se restreint à la manipulation des éléments à travers ces domaines, il devient possible de rendre homogène leur représentation et leurs traitements.
Delà, les contraintes de la synchronisation se traduisent alors par un choix adapté de ces valeurs.

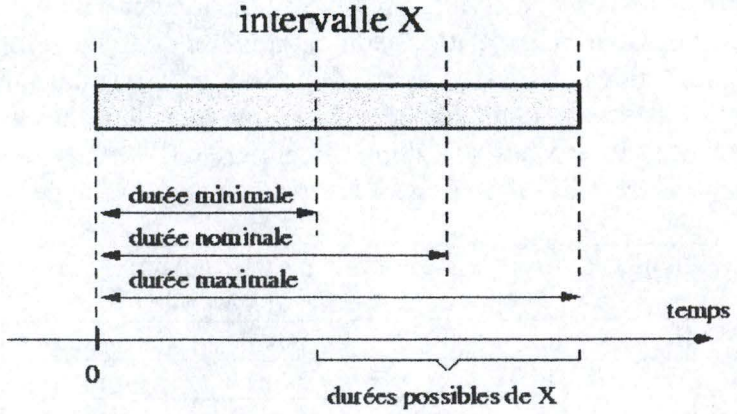


Figure 10. Domaine de validité d'un intervalle temporel

3.3 Expressions des relations temporelles.

Les expressions des relations permettent de décrire la manière dont les éléments doivent être liés temporellement pour réaliser un scénario d'un document. Les relations temporelles servent comme entrée de la partie analyse d'un modèle et comme une mémoire des intentions de l'auteur du document.

Il existe deux classes de relations selon les modes de présentation des unités temporelles(par instants ou par intervalles). Dans les deux cas, les scénarios sont représentés à partir d'un ensemble B de relations dites primitives. Les relations sont exclusives et permettent d'exprimer comment les unités temporelles d'un document se situent les unes par rapport aux autres.

A. Les relations à base d'instants.

Les relations à base d'instants font parties de l'Algèbre d'instants noté PA(Point Algebra). Les unités considérées sont les instants de début et de fin des médias.
Un instant peut :

- précéder un autre (<),
- succéder un autre(>),
- être égal à un autre(=).

L'ensemble des relations de primitives de l'algèbre d'instant est noté B={<, >, =}.
Les relations primitives interviennent souvent dans la composition multimédia car elles interviennent dans la construction des scénarios multimédia les plus simples.
On peut composer ces relations primitives (ou disjonctives) qui forment les parties de B. Cet ensemble est noté 2^B. Par exemple soit les instants e1 et e2, la relation e1<=e2 signifie que « e1 a eu lieu avant ou en même temps que e2 ».

B. Les relations à base d'intervalles.

Les relations à base d'intervalles font patries de l'Algèbre d'intervalles noté IA(Interval Algebra). Ces relations se réduisent à des combinaisons de

positionnement de deux intervalles sur la droite orientée. Le modèle général a été proposé par J.F Allen[3]. Ces relations sont au nombre de treize dont sept relations de base et leurs relations inverses. Ces relations sont reparties dans deux classes. Il y a les relations de séquentialité et celles qui introduisent le parallélisme de présentation. Le tableau qui suit donne quelques relations et leurs équivalences à base d'instants où x+ et x- représentent les instants de début et de fin.

Relation	Symbole	Inverse	Relation à base d'instants équivalente au symbole	Classe
X before y	B	bi	$x^- < x^+ < y^- < y^+$	Seq
x meets y	M	mi	$x^- < x^+ = y^- < y^+$	Seq
x overlaps y	O	oi	$x^- < y^- < x^+ < y^+$	Par
Y finishes x	F	fi	$x^- < y^- < x^+ = y^+$	Par
Y during x	D	di	$x^- < y^- < y^+ < x^+$	Par
x starts y	S	si	$x^- = y^- < x^+ < y^+$	Par
X equals y	E	ei	$x^- = y^- < x^+ = y^+$	Par

Conclusion.

Dans ce chapitre, nous avons introduit les différentes notions sur l'édition des documents multimédias, la synchronisation temporelle et les scénarios temporelles. Dans le chapitre suivant, nous allons décrire un langage d'édition de documents multimédias et comment ce langage peut implémenter certains aspects de la synchronisation temporelle.

---

[3] ALLEN J. F., ``Maintaining Knowledge about Temporal Intervals", *Communications of the ACM,* vol. 26, num. 11, pp. 832-843, novembre 1983.

Chapitre 2. Le langage SMIL[4]

Dans ce chapitre, nous allons décrire le langage SMIL, nous commencerons par l'origine de ce langage, puis nous décrirons la syntaxe de ce langage et la signification des différentes structures. Nous terminerons ce chapitre en analysant l'apport de ce langage à la présentation multimédia temporelle.

[4] SMIL: Synchronized Multimedia Integration Language

2.1 Origine de SMIL

SMIL est un langage défini par rapport à XML[5]. Le langage XML, lui-même étant défini par rapport à SGML[6], le langage naturel pour l'édition des documents. SGML tient son origine au langage GML[7] développé en 1969 par une équipe dirigée par Goldfard, Mosher et Lorie de la société IBM. Ce langage a été normalisé par l'Institut de normalisation américain ANSI et est devenu SGML.

Au début des années quatre-vingt-dix, lorsque le WWW[8] commence à se développer, il fallait un système de description plus simple, accessible à tout le monde, c'est ainsi qu'est né le langage HTML[9] qui est un type de document spécial de SGML. Ce langage est le plus utilisé aujourd'hui pour la publication des documents multimédias sur le WWW.

A la fin des années quatre-vingt-dix, il existait des tendances qui réclamaient la suppression des limites de HTML. Ces limites empêchaient toute l'utilisation de la puissance de SGML. Le fait que SGML ne s'est pas imposé comme un langage d'édition de documents sur le WWW, est du surtout à sa trop grande complexité et en plus les programmes capables de traiter les documents SGML étaient difficiles à mettre au point.

Devant ce problème le W3C[10] a formé un groupe de travail pour créer une variante simplifiée de SGML. Le résultat fut le langage XML. Le langage XML a conservé tous les éléments les plus utiles de SGML et tout ce qui est secondaire ou trop compliqué a été abandonné.

Tous ces langages ont un point commun, ce sont des langages de marquage. Le principe d'un langage de marquage est l'utilisation des balises pour exécuter une certaine action. Ces balises peuvent être imbriquées pour exprimer des relations plus complexes.

La syntaxe générale des balises d'un document est :
- <balise>....</balise> qui forment une paire de balises d'ouverture et de fermeture.
- <balise> qui n'a pas de balise pour la fermeture.

La sémantique des balises est différente selon le langage utilisé.

Dans le cas du langage HTML, les balises sont utilisées pour la mise en page d'un document multimédia[11], son formatage pour la visualisation ou soit pour référence

---

[5] XML: Extensible Markup Language.
[6] SGML :Standard Generalized Markup Language.
[7] GML :Generalized Markup Language.
[8] WWW: Word Wide Web.
[9] HTML: HyperText Markup Langage.
[10] W3C: Wold Wide Web Consortium
[11] Au début du HTML c'était un document texte.

d'autres documents. Les parties d'un document à mettre en valeur sont encadrées par des balises qui définissent l'action à exécuter.
Si par exemple on veut mettre en gras un mot, on le met entre des balises

<GRAS> mot en gras </GRAS>.

L'affichage sera **mot en gras.** Le programme utilisé pour l'affichage doit donc connaître le jeu des balises en vigueur. Le texte d'un document HTML repose sur des données non structurées. Il n'est pas possible de traiter de façon automatique par un programme les données d'un tel document. Tous au plus le programme pourra lire la suite de caractère sans signification pour lui.

Dans le cas du langage XML, les balises sont utilisées pour structure des données d'un document. Ces balises indiquent la place d'une donnée dans la structure d'un document.

Exemple

```
<LIVRE>
        <TITRE> JAVA Facile</TITRE>
        <AUTEUR> Alexandre Maret </AUTEUR>
</LIVRE>
```

Cet exemple nous indique qu'il s'agit en question d'un livre, son titre est « JAVA Facile » et l'auteur est Alexandre Maret. Un programme qui traite un document XML pourra agir à bon escient, à cause de cette structuration de données.

L'avantage du langage XML par rapport au langage HTML est la possibilité de structure des documents de manière à simplifier leur échange et en préservant la possibilité de retraitement par une autre application.

L'exemple du livre nous montre qu'ils existent plusieurs formats de données, c'est à dire l'ensemble des règles d'organisation auxquelles sont soumises les données d'un document. Ces règles sont décrites dans un fichier qu'on appelle DTD[12]. Le fichier DTD décrit la syntaxe du document en question. Avec XML l'utilisateur peut donc créer une description adaptée à ses documents.

L'évolution rapide du WWW lors de ces dernières années a soulevé plusieurs problèmes notamment :

- l'incorporation d'autres médias que le texte comme l'image, le son et la vidéo. Il faut préciser les relations temporelles(comme la séquence,..) et la position de ces médias au moment de la présentation,
- l'inexistence de formats d'échanges de ses médias pose des problèmes car il existe une diversité de plates formes matérielles et logicielles.

Pour résoudre ces problèmes d'intégration sur le WWW, le W3C a créé un autre groupe de travail qui a défini un DTD spécial pour l'édition de documents

---

[12] DTD : Document Type Definition.

multimédias. La normalisation a donné naissance à SMIL. La description des règles d'un document SMIL est faite dans un DTD SMIL, écrit dans le langage XML. Ce langage permet d'éditer des documents multimédias pour le WWW, tout en étant déclaratif et sous format textuel.
Nous terminons cette brève historique de SMIL en précisant la différence entre un langage et un métalangage. Un métalangage est un langage qui décrit les règles utilisées par d'autres langages dérivés du métalangage. Ainsi SGML est métalangage pour les langages XML et HTML et XML est à son tour un métalangage pour SMIL ainsi que d'autres types de documents.

2.2 Description de la syntaxe SMIL.

Précisons d'abord ce qu'est un **objet multimédia** dans le cadre de ce chapitre. Un objet multimédia est soit un média image, un média texte, un média son, un média vidéo ou un scénario temporel de ces médias. Dans la suite nous utiliserons soit un objet multimédia ou un média pour dire la même chose.

SMIL est un langage déclaratif qui permet d'améliorer la présentation des documents multimédias sur le WWW, en ajoutant les fonctions de synchronisation temporelle entre objets multimédias. C'est un langage d'intégration où les objets multimédias sont référencés et non pas inclus. SMIL est défini par un DTD XML et il se base sur des pairs Attributs/valeurs.
Nous allons décrire dans la suite, l'essentiel de la structure d'un document SMIL au moyen de petits exemples. La description complète peut être consultée sur le site du W3C.

SMIL permet de spécifier :
- le contenu des objets multimédias,
- la répartition spatiale des objets multimédias,
- la répartition temporelle des objets multimédias,
- les liens(navigations inter objets),
- les contenus alternatifs(bande passante, langue,...),
- les annotations.

Les objets multimédias utilisés dans SMIL ont une dimension spatiale, temporelle ou les deux à la fois.
Par exemple :
- le texte est un objet à deux dimensions,
- la vidéo est un objet à deux dimensions avec une dépendance temporelle,
- l'audio a seulement une dépendance temporelle.

2.2.1 Structure générale

La structure générale d'un document SMIL est toujours :

```
<smil>
    <l'en-tête>
    <le corps>
</smil>
```

Un document SMIL commence par la balise <smil> et se termine par la balise </smil>. Ce document est composé de deux parties : L'en-tête et le corps.
L'en-tête peut être absent, tandis que le corps est obligatoire.

Dans l'en-tête, on définit la disposition spatiale des objets multimédias tandis que dans le corps on définit la disposition temporelle des objets multimédias.

```
<smil>
    <head>
    [...]
    </head>
        <body>
        [...]
        </body>
</smil>
```

L'en-tête d'un document SMIL est délimité par les balises <head> et </head> et le corps d'un document SMIL est compris entre les balises <body> et </body>.

2.2.2 L'en-tête d'un document SMIL

L'en-tête peut avoir deux parties distinctes :
- une partie définie par la balise <meta ...> spécifie les informations où on peut trouver les objets multimédias,
- l'autre partie définie entre les balises <layout> et </layout> spécifie les informations sur la disposition spatiale des objets multimédias.

A. La balise <meta>

Les informations de la balise <meta> définit les propriétés générales d'un document SMIL. Il s'agit par exemple de l'autheur du document, une liste de mots clés à utiliser dans le document ainsi que leurs valeurs. Chaque propriète du document est encapsulé dans un balise <meta>

Les principaux attributs et leurs valeuts de la balise <meta> sont :

- attribut "**name**". Cette attribut identifie une propriété définie. La liste des propriétés est ouverte, seulement deux propriétés sont importantes à savoir name="title" et name="base" qui informe l'utilisateur sur le titre du document et le chemin de base d'accès aux objets multimédias,
- attribut "**content**". Il s'agit de la valeur de la propriété name.

Exemple:

```
<meta name="title" content="Présentation de SMIL" />
<meta name="base" content="http://www.info.fundp.ac.be" />
<meta name="author" content="Monsieur Toto" />
```

Si cette partie est absente, le chemin par défaut est le répertoire où se trouve le document SMIL.

B. Les balises <layout> et </layout>

Les informations contenues entre les balises <layout> et </layout> spécifient la disposition générale des différents objets multimédias du document SMIL. Il existe plusieurs types de disposition spatiale.
La première manière de définir une disposition spatiale est :

```
<layout>
        <root-layout ....>
        (<region> ...</region>)*[13]
</layout>
```

Il s'agit de définir une fenêtre principale pour la visualisation du document SMIL. Cette fenêtre est spécifiée à l'intérieure de la balise <root_layout...>.
Les principaux attributs de cette fenêtre sont :

- l'attribut **width** qui indique la largeur de cette fenêtre en pixels,
- l'attribut **height** qui indique la hauteur de la fenêtre en pixels,
- l'attribut **background-color** qui indique la couleur de fond de cette fenêtre.

A l'intérieur de la fenêtre principale, on définit une suite de régions au moyen des balises <region>...</region>.
Dans ces régions, seront affichées les objets multimédias du document.
Les attributs principaux de chaque sous régions sont :

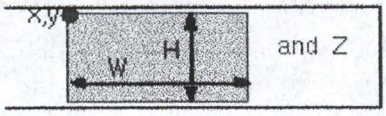
- l'attribut **width** qui indique la largeur de cette région en pixels ou en pourcentage par rapport à la largeur de la fenêtre principale,
- l'attribut **height** qui indique la hauteur de la fenêtre en pixels ou en pourcentage par rapport à la largeur de la fenêtre principale,
- les attributs **top** et **left** qui indiquent l'espacement par rapport au coin supérieur gauche de la fenêtre principale,

---

[13] * signifie zero ou plusieurs fois

- l'attribut **z-index** qui indique l'ordre d'empilement quand il y a superposition de plusieurs régions,
- l'attribut **id** qui est un identificateur de la région,
- l'attribut **fit** qui indique la manière dont un média sera affiché dans cette région si la taille diffère de la largeur et la hauteur de la région.

Chaque objet multimédia contient une référence à une région où il doit apparaître. Le schéma suivant montre les attributs d'une région par rapport à la fenêtre principale c'est à dire left(X), top(Y), width(W),height(H) et z-index(Z).



Voici un exemple de ce qu'on vient de décrire ci dessus.
Le code dans la partie head est :

```
<smil>
  <head>
    <layout>
      <root-layout width="300" height="200"
                   background-color="white" />
        <region id="region1" left="75" top="50"
        width="32" height="32" />
    </layout>
  </head>
[...]
```

Dans cette exemple, on définit un fenêtre principale de 300*200 pixels et un région de 32*32 pixels et son coin supérieur gauche se trouve à 75 pixels selon l'abscisse et 50 pixels selon l'ordonnée du coin supérieur gauche de la fenêtre principale.

La deuxième manière de définir une disposition spatiale est :

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

Cette diposition spatiale est par défaut pour tous les objets multimédias, c'est à l'utilisateur de définir comment les différents objets seront affichés dans une fenêtre de visualisation. En plus si la fénêtre principle n'est spécifiée alors la taille de la fenêtre de visualisation est calculée par le logiciel de visualisation de façon à ce qu'elle soit aussi large que l'objet multimédia le plus large.

La troisième manière de définir une disposition spatiale est :

<layout> <region>...</region></layout>
Cette disposition definit une region où tous les objets seront affichés.

2.2.3 Le corps d'un document SMIL

Dans cette partie, on précise quels sont les objets multimédias du document SMIL et des scénarios temporels du document SMIL. Il s'agit de définir comment les différents objets multimédias seront présentes les uns par rapport aux autres.

A. Les objets multimédias[14].

Les objets multimédias sont représentés par des balises suivantes :

- <text .... /> pour un média texte,
- <audio.../> pour un média audio,
- <video.../> pour un média vidéo,
- < img..../> pour un média image,
- <animation.../> pour une animation.

Les principaux attributs de ces médias sont :

- l'attribut **alt** qui montre un texte de remplacement à afficher à la place du média au cas où ce média est introuvable,
- l'attribut **author** qui indique le nom de l'auteur du média en question,
- l'attribut **begin** qui indique l'instant de début pour présente le média,
- l'attribut **copyright** qui indique les informations sur les droits d'utilisations du média,
- l'attribut **dur** qui indique la durée du média. La durée d'un objet multimédia est **intrinsèque** pour l'audio et la vidéo tandis que l'image et un texte ont une durée **explicité** spécifiée par l'utilisateur.
- l'attribut **begin** qui indique l'instant de début pour présente le média,
- l'attribut **id** qui est l'identificateur du média,
- l'attribut **region** qui référence la région ou le média sera affiché,
- l'attribut **src** qui localise le média.

Les attributs **begin, end** ont pour valeur le temps d'horloge dont la syntaxe est la suivante :

Temps ::= Temps_complet | Temps_partiel | Temps_fraction
Temps_complet::= Heures ":" Minutes ":" Secondes ("." Fractions)?
Temps_partiel ::= Minutes ":" Secondes ("." Fractions)?
Temps_fraction ::= Entier ("." Fractions)? ("h" | "min" | "s" | "ms")?
Heures ::= Deuxchiffres
Minutes ::= Deuxchiffres; entre 00 et 59
Secondes ::= Deuxchiffres; entre 00 et 59
Fractions::= Chiffre+
Entier ::= Chiffre+
Deucxchiffres::= Chiffre Chiffre
Chiffre ::= [0-9]

---

[14] la liste compleéte des medias utilisés dans SMIL peut être trouve sur le site http://www.w3c.org/TR/REC-smil/#smil

Ou :
A|B signifie A ou B,
+ signifie zero ou un répétions de plusieurs fois,
A B signifie A et B,
? signifie 0 ou 1 fois.

L'attribut **src** a pour valeur un URL. L'URL(Uniform Ressource Locator) permet d'envoyer ou recevoir des données et des documents à travers le réseau, ainsi que les méthodes de transmission associées.
Dans un document SMIL, la localisation des médias est :

- relatif : l'attribut **src** contient seulement le nom du média et on utilise les informations de la balise <meta> pour trouver le chemin complet,
- absolu : l'attribut **src** contient le chemin complet du média.

Exemples de médias.

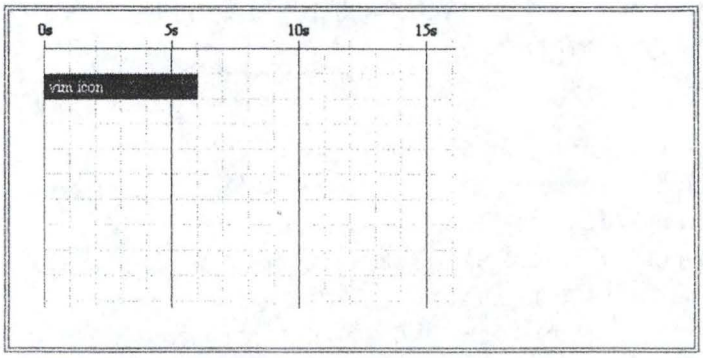
<text src="file.html" dur ="3s" />

<audio src= "http:/ /www.info.fundp.ac.be/audio/essai.wav" dur ="3s" begin="7s"/>
<video src="rtsp://www.w3c.org/SYMM/vid.rm" dur ="3s" begin="6s" />

Dans l'exemple qui suit, on veut que l'image dont la source est **vim32x32.gif** soit visible pour une durée de 6s au moyen de l'attribut **dur** dans la région **vim_icon.** Au cas ou l'image n'est pas disponible, un texte de remplacement sera afficher dans cette région au moyen de l'attribut **alt.**

```
<smil>
 <head>
  <layout>
   <root-layout width="300" height="200"
       background-color="white" />
   <region id="vim_icon" left="75" top="50"
       width="32" height="32" />
  </layout>
 </head>
 <body>
  <img src="vim32x32.gif" alt="The vim icon"
       region="vim_icon" dur="6s" />
 </body>
</smil>
```

La visualisation sera :



On peut aussi préciser le temps en secondes avant de commencer la visualisation de cette image par l'attribut **begin** ="nombre de seconde" et le temps en seconde pour terminer la visualisation par l'attribut **end** = "nombre de seconde".

```
<smil>
 <head>
  <layout>
   <root-layout width="300" height="200"
      background-color="white" />
   <region id="vim_icon" left="75" top="50"
      width="32" height="32" />
  </layout>
 </head>
 <body>
  <img src="vim32x32.gif" alt="The vim icon"
      region="vim_icon" dur="6s" begin="2s" />
 </body>
</smil>
```

B. Scénarios temporels

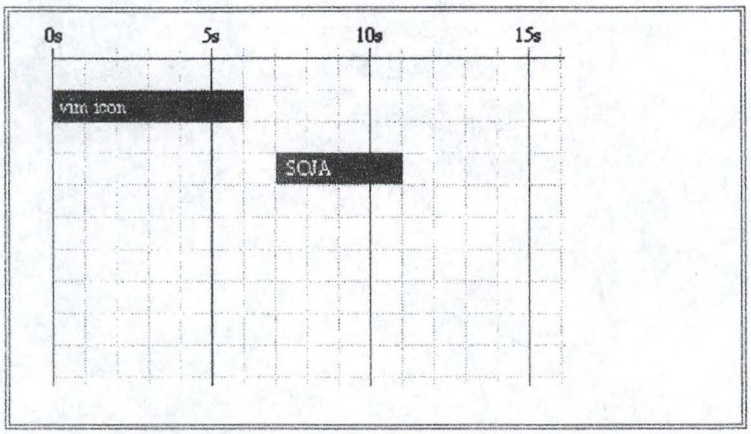
Il existe deux types de scénarios temporels dans SMIL :
- Un scénario séquentiel décrit par les balises <seq> et </seq>
- Un scénario parallèle décrit par les balises <par> et </par>

B.1 Les balises <seq> et </seq>

Les balises <seq> et </seq> groupent les objets multimédias pour qu'ils soient présentés en séquence. Les objets multimédias sont joués les uns après les autres dans l'ordre où ils apparaissent à l'intérieur de ces balises.
Les attributs de ces balises sont les mêmes que celles des médias sauf que :
- l'attribut **begin** spécifie l'instant de début de la séquence. S'il n'est pas spécifié, il correspond implicitement à l'instant de début du premier média,
- l'attribut **end** spécifie l'instant de fin de la séquence. S'il n'est pas spécifié, il correspond implicitement à l'instant de fin du dernier média,

- les attributs **begin** de chaque média de la séquence, sauf le premier média, signifient qu'il faut commencer la présentation après x temps de l'instant de fin du média précédant dans la séquence.

Supposons qu'on veut présenter l'icône vim-icon pendant 6s, attendre 1s, puis l'icône soja pendant 4s.

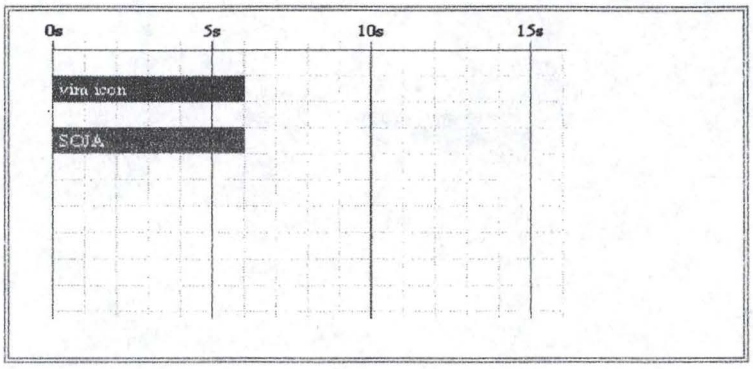


Le code est dans ce cas :

```
 smil>
<head>
 <layout>
  <root-layout width="300" height="200"
     background-color="white" />
  <region id="vim_icon" left="75" top="50"
     width="32" height="32" />
  <region id="soja_icon" left="150" top="50"
     width="100" height="30" />
 </layout>
</head>
<body>
 <seq>
 <img src="vim32x32.gif"
     alt="The vim icon"
     region="vim_icon" dur="6s" />
 <img src="madewithsoja.gif"
     alt=”Image SOJA"
     region="soja_icon" dur="4s" begin="1s" />
 </seq>
</body>
</smil>
```

B.2 Les balises <par> et </par>

Les balises <par> et </par> groupent les éléments qui sont joués en parallèle. Les objets multimédias qui se trouve à l'intérieur de cette balise démarrent en même temps. Par exemple on peut présenter les deux images en même temps.




```
<smil>
 <head>
  <layout>
   <root-layout width="300" height="200"
      background-color="white" />
   <region id="vim_icon" left="75" top="50"
      width="32" height="32" />
   <region id="soja_icon" left="150" top="50"
      width="100" height="30" />
  </layout>
 </head>
 <body>
  <par>
   <img src="vim32x32.gif" alt="The vim icon"
      region="vim_icon" dur="6s" />
   <img src="madewithsoja.gif" alt="Made with SOJA"
      region="soja_icon" dur="6s" />
  </par>
 </body>
</smil>
```

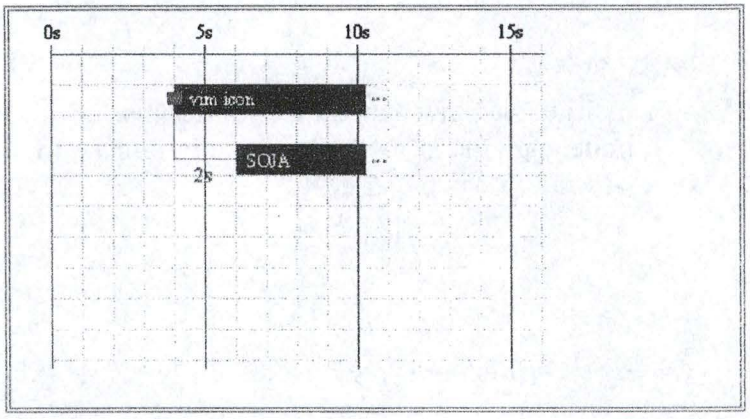
L'instant de début des objets multimédias de la balise « par» est égal au temps de début de la balise lui-même.
Les attributs de ces balises sont les mêmes que celles des médias sauf que :

- l'attribut **begin** spécifie l'instant de début de tous les médias,
- l'attribut **end** spécifie l'instant de fin de la présentation et correspond à l'instant de fin du média qui finit en dernier lieu ou le média qui finit en premier lieu,

- Les médias à l'intérieur de la balise « par » peuvent avoir leurs propres attributs de début et de fin par rapport aux autres.

Par exemple dans l'exemple suivant, on veut que la première et la deuxième image soient retardées respectivement de 4s et 6s par rapport au début de la balise par.

```
<smil>
<body>
 <par>
  <img src="vim32x32.gif" region="vim_icon"
     begin="4s" />
  <img src="madewithsoja.gif" region="soja_icon"
     begin="6s" />
 </par>
</body>
</smil>
```

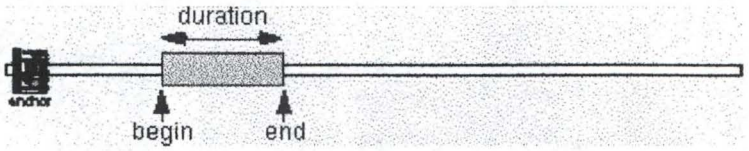


Ces deux mécanismes que nous venons de voir, sont à la base de la synchronisation entre médias. On peut les combiner pour obtenir une présentation complexe par une suite d'imbrications successives.

C. Remarques sur la durée des médias

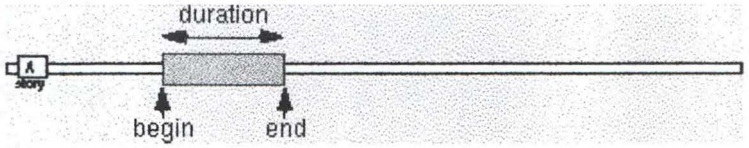
Un objet multimédia possédant une durée implicite ou explicite et un temps de début, a un temps de fin qui vaut la somme du temps de début et la durée de cet objet.

<video src="video.mov" region="V-main" begin="4s" />



Un objet multimédia possédant un temps de début explicite et un temps de fin explicite a une durée egal à la somme du temps de fin moins le temps de début.

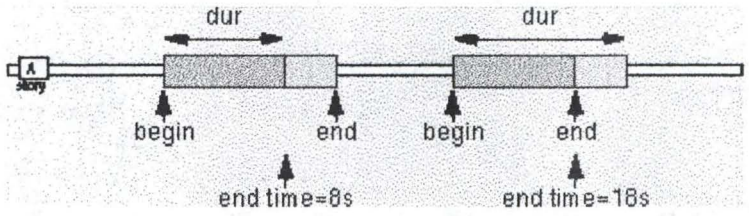
<text src="title.html" region="m_title" begin="4s" end="8s" />



Un objet multimédia possédant un temps de début explicite, une durée et un temps de fin explicite, a un temps de fin égal au minimum entre le temps de fin explicite et la somme du temps de début et de durée

Exemple

<seq>
<text scr="title.html" region="m_title" begin="4s" dur="4s" end="10s" />
<text scr="title.html" region="m_title" begin="14s" dur="6s" end="18s" />
</seq>



Si le temps de fin est antérieur au temps de début alors l'objet multimédia n'est pas joué. Le temps de fin d'un élément multimédia possédant une durée indéfinie est déterminé par le temps de fin de son parent.

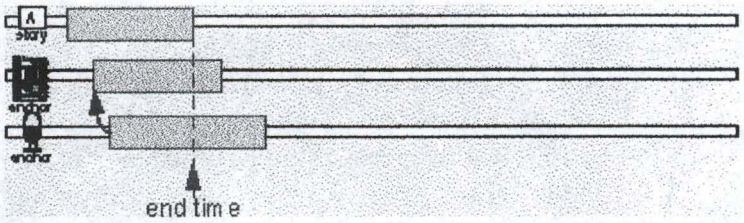
Si l'objet n'a pas de temps de fin défini (ou n'existe pas) alors il sera joué indéfiniment.

Un objet « 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.

L'objet « par » peut se terminer :
- soit quand le premier élément multimédia se termine,
- soit par rapport à un objet référencé,
- soit quand le dernier élément multimédia se termine.

On précise par l'attribut **endsync** = « first|id= id-value |last». Cette attribut peut avoir comme valeurs first, last ou l'identificateur d'un autre média.
Exemple:

```
<par endsync="first">
<text src="leader_title.html" region="m_title" dur="5s" />
<video id="v1" src="cnn.mpv" region="V-main" begin="1.4s" />
<audio src="cnn.aiff" region="music" begin="2s" />
</par>
```



2.3 Mécanisme d'événement

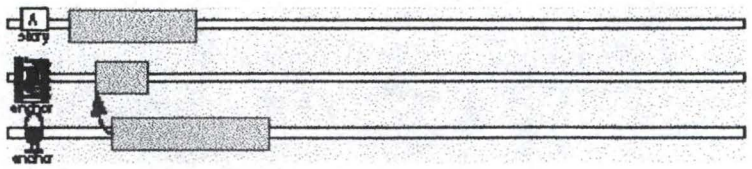
Il existe un mécanisme appelé *modèle d'événement*, qui permet de synchroniser les différents médias. Il s'agit d'exprimer l'instant de début d'un média relatif à un l'instant de début ou de fin d'un autre média. Si un média commence, il envoie un événement "begin" à un autre média qui l'attend. Pour qu'un média attende un événement qui doit se produire, on modifie l'une de ces attributs(**begin** ou **end**) de la manière suivante :

```
<balise begin="id(identificateur)(begin)" />
<balise begin="id(identificateur)(temps d'horloge)" />
<balise begin="id(identificateur)(end)" />
```

Exemple :

```
<par>
<text src="leader_title.html" region="m_title" dur="5s" />
<video id="v1" src="cnn.mpv" region="V-main" begin="1.4s" />
<audio src="cnn.aiff" region="music" begin="id(v1)(0.5s)" />
</par>
```

L'audio est retardée de 0,5 seconde par rapport au début du média vidéo "v1"



2.4 Notions avancées de SMIL

Dans les sections précédentes, nous avons décrit les principaux mécanismes de fonctionnement de SMIL à savoir la disposition spatiale et la disposition temporelle. Les mécanismes que nous allons décrire utilisés les caractéristiques de la machine sur laquelle se déroule la présentation et les caractéristiques du réseau qui sert de support pour le transfert des médias.

2.4.1 Le contenu alternatif

Le contenu alternatif permet de présenter une partie d'un document SMIL selon les caractéristiques de la machine et du réseau. En effet les stations de travail peuvent avoir des capacités différentes. Si les médias d'un document SMIL sont éparpillés sur le réseau, les transmissions de ces médias s'effectuent à des vitesses différentes. L'autre point qui a motivé l'introduction d'un contenu alternatif, est que des utilisateurs différents peuvent avoir des tâches différentes ou des préférences différentes sur les médias.
SMIL permet de prendre en compte les caractéristiques cités ci dessus par l'utilisation des balises <switch> et </switch>.
Les attributs de ces balises sont :

- l'attribut **system-bitrate** spécifie la bande passante disponible sur le réseau,
- l'attribut **system-caption** autorise les auteurs à fournir des sous-titres, pour les personnes malentendantes ou qui apprennent une langue,
- l'attribut **system-language** spécifie le groupe de langue désiré,
- l'attribut **system-overdub-or-caption** sélectionne entre le doublage et les sous-titres,
- l'attribut **system-screen-size** donne les caractéristiques de l'écran,
- enfin, l'attribut **system-screen-depth** donne la profondeur de la palette de couleurs.

Pour que ces attributs soient pris en compte, les médias d'un document SMIL doivent spécifier la valeur de ces attributs.

Exemple.

Supposons qu'on veut jouer un média audio qui se trouve sur une machine distante et qu'il faut tenir compte de la bande passante du réseau et la langue de utilisateur.

Le code est :

```
<smil>
  <body>
    <switch>
      <audio system-bitrate="44000 system-language="fr" src="fr-hi-res.aiff" />
      <audio system-bitrate="44000 system-language="en" src="eng-hi-res.aiff" />
      <audio system-bitrate="16000 system-language="fr" src="fr-low-res.aiff" />
      <audio system-bitrate="16000 system-language="en" src="eng-low-res.aiff" />
    </switch>
  </body>
</smil>
```

Au plus un seul média de la balise est joué et c'est le premier média acceptable qui est joué. Il faut donc placer les médias de meilleure qualité en premier lieu.
Si aucun média n'est acceptable alors aucun média de la balise `switch` n'est joué.

2.4.2 Les liens

Le mécanisme des liens permet de naviguer dans le document SMIL ou vers d'autres documents. Il existe plusieurs types de liens.

a. Lien d'un média à une présentation

Le mécanisme d'un lien à une présentation permet d'afficher une nouvelle présentation dans une autre fenêtre de visualisation pointe par un média. Pour spécifier ce mécanisme, les balises <a> et sont utilisées.
Les principaux attributs de la balise <a> sont :

- l'attribut **show** qui monte une nouvelle présentation,
- l'attribut **href** qui indique la localisation du document à présenter.

Exemple.

```
<a show="new" href="document.smil">
<img src="toto.gif" region="main" />
</a>
```

Une fois qu'on clique sur le média image « toto.gif », une nouvelle présentation SMIL commence dans une autre fenêtre et la présentation source n'est pas affectée.

b. Lien d'un média à un autre média

Le mécanisme d'un lien à un autre média permet de référencer un média qui se trouve dans une autre représentation. La présentation destination commence comme si la présentation avait été avancée rapidement jusqu'au début de l'élément désigné par le lien. Les balises <a> et <a/> sont utilisées, mais on ajoute l'identificateur du média référencé au moyen du signe #.

Exemple

```
<a show="new" href="document.smil#média1">
<img src="toto.gif" region="main" />
</a>
```

Une nouvelle présentation démarre dans une autre fenêtre à partir du média qui a l'identificateur « média1 ».

c. Lien d'un média vers une partie d'un autre média

Le mécanisme d'un lien à une partie d'un autre média permet de référencer une partie d'un autre média. On peut spécifier des sous parties spatiales ou temporelles d'un média par la balise <anchor…>
Les principaux attributs de la balise <anchor> sont :

- l'attribut **coords** qui indique les sous parties spatiales du média, l'ordre des valeurs de coords est left, top,widht et height exprimés en pixels ou en pourcentage,
- les attributs **begin** et **end** indiquent les sous parties temporelles du média,
- l'attribut **id donne** l'identificateur du média référencé.

Exemple

```
<video src="zoomin.mpeg" region="V_main" >
 <anchor id="mic" coords="40%, 70%, 55%, 100%" />
</video>
```

Dans cette exemple la vidéo a une référence vers un média identifié par « mic ». Lors de la présentation, seulement la partie délimité par les valeurs de l'attribut « coords » de ce média est affichée.

Nous allons terminer la description d'un document SMIL en donnant la structure hiérarchique et les informations de chaque niveau de la hiérarchie.

Partitionnement de SMIL.

```
<smil>
    <head>
        <meta>
        ....Informations générales sur le document
        </meta>
        <layout>
        .... Définition de mise en page
        </layout>
    </head>
    <body>
     .... Objets multimédias et csénarios temporelles
     .... Liens
     </body>
</smil>
```

3. Apport de SMIL par rapport aux présentations multimédias temporelles.

3.1 SMIL par rapport aux types de médias

SMIL permet de gérer les médias temporisés et non temporisés, mais SMIL n'est pas une application multimédia. C'est seulement un éditeur de document multimédia, il ne s'occupe pas de la restitution de documents multimédias.

3.2 SMIL par rapport à la synchronisation temporelle.

Comme nous venons de la voir, SMIL permet de gérer la synchronisation temporelle dans un document multimédia en utilisant le mécanisme d'événements. SMIL peut gérer la synchronisation naturelle pour les médias temporisés et la synchronisation artificielle pour les médias non temporisés.

3.3 SMIL par rapport aux scénarios temporels

SMIL permet de gérer des scénarios au moyen des balises « par » et « seq ». L'une de type séquentiel et l'autre de type parallèle. Ces scénarios sont tous indéterministes car les enchaînements qui s'effectueront au moment de la présentation.

3.4 SMIL par rapport aux modèles de documents

SMIL respecte certaines propriétés de modèles des documents. SMIL garantit la séparation de l'édition et de la restitution de documents multimédias, le traitement homogène des médias(même syntaxe et mêmes attributs), mais ne garantit pas la résolution automatique des contraintes temporelles.

3.5 SMIL par rapport à l'organisation d'un document

SMIL permet d'organiser un document :
- selon l'organisation logique(un document SMIL est fort structure),
- selon l'organisation(chaque média possède par défaut une région où il sera présenter),
- selon l'organisation hypermédia(SMIL permet la navigation inter et intra documents),
- enfin, selon l'organisation temporelle(SMIL permet de définir des scénarios temporels).

3.6 SMIL par rapports aux types d'édition

SMIL permet de représenter un document sous forme de graphe. Ce document possède les avantages de l'édition basée sur les graphes.

Conclusion

Dans ce chapitre, nous avons décrit une partie du langage SMIL. C'est cette partie qui sera implémentée dans l'application que nous allons développer au chapitre 4. Mais plusieurs modules de SMIL ont été passé sous silence(voir annexe 2).
Pour de plus amples informations, on peut consulter les sites suivants :
- http://www.w3c.org /AudioVideo/
- http://www.helio.org
- http://justsmile.com
- http://www.euroclid.fr/Cours SMIL W3C/

Chapitre 3. Etat de l'art des outils

Dans ce chapitre nous décrirons l'état de l'art des outils d'édition et de présentation des documents multimédias, ainsi que leurs comparaisons.

Les outils qui existent actuellement sont classés dans deux catégories:
- Les « Players » : Les players sont utilisés seulement pour visualiser un document SMIL. Il peut s'agir d'une application autonome ou une applet.
- Les « Authoring Tools » :Les authoring tools sont des applications qui prennent en charge l'édition, la correction et la visualisation d'un document SMIL.

A part les outils de la société Real System qui sont opérationnels, les autres sont à un stade de développement.

3.1 RealSystem G2 (http://www.real.com)

RealNetwork, Inc est l'une des sociétés qui a participé au groupe de travail créé par le W3C à l'élaboration de SMIL. Cette société possède des outils qui peuvent être téléchargés gratuitement sur leur site et les outils commerciaux qu'il faut acheter. Pour ce qui est des outils à accès libres, certaines fonctions ne sont pas implémentées, ou bien certains formats de médias ne sont pas supportés.

3.1.1 RealPlayer

Il s'agit d'un Player gratuit de documents SMIL. Le Real player supporte plusieurs formats de fichiers : les formats standards et les formats propriétaires.

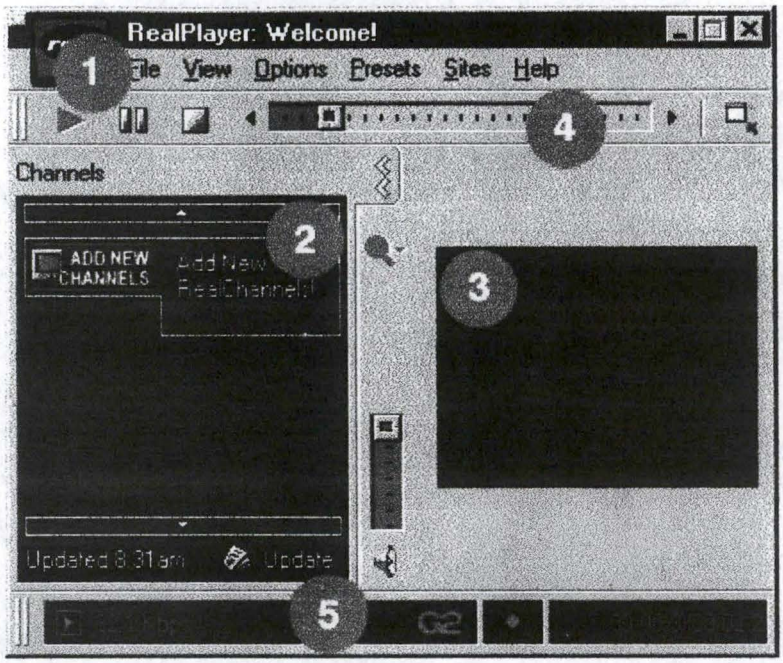
Formats propriétaires Real :
- RealText
- RealVideo(.rm)
- RealAudio(.rm)
- RealPix
- RealFlash

Ces formats sont des fichiers compressés et optimisés pour les présentations via les réseaux où il faut préciser notamment le bande passante.

Formats standards :
- Images : JPEG, GIF,GIF animée
- Audio : AIFF(.aif), AU(.au),WAV(.wav)
- Video : AVI(.avi), QuickTime(.mov),Vivo(.viv),ASF(.asf)
- Text

RealPlayer permet de jouer un document SMIL via un réseau conforme à la version SMIL1.0 (c'est la version actuelle que tout le monde utilisé pour concevoir les applications ou applets).
Il existe une série d'outils permettant de créer les différents médias(RealVideo, RealAudio, Real Text).

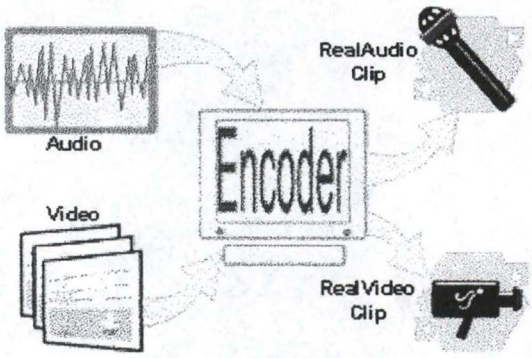
L'interface de Real dans sa version actuelle est la suivante:



1. Boutons Play, Pause et Stop.
2. Canaux.
3. La fenêtre ou se déroule la présentation.
4. Bar de progression qui montre le temps courant.
5. Bar de statut qui donne les informations sur la bande passante du réseau.

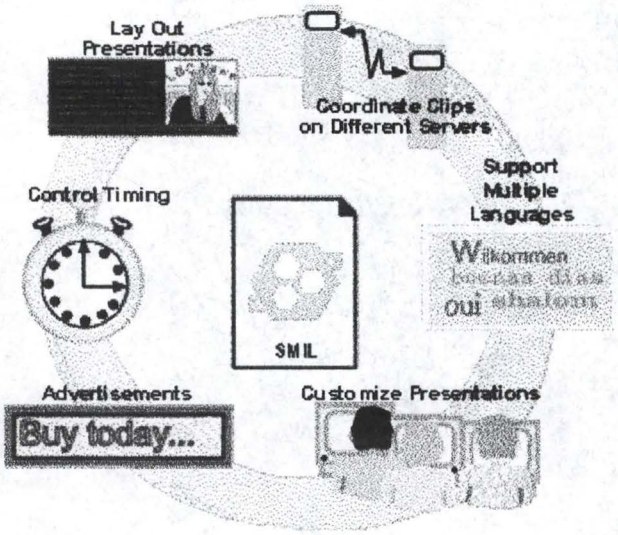
3.1.2 RealSystem™ G2 Production

Real propose aussi les authoring tools gratuits qui permettent d'éditer du début à la fin un document SMIL. Les différents médias sont crées et enregistrés sous le format normale(.wav, .au,.mpg,..) ou sous format real (.rm) pour respecter les caractéristiques du réseau(bande passante,..).



Les outils de création de flux(http://www3.real.com/products/tools/)

Une fois que les différents médias sont créés, on les assemble dans un document SMIL au moyen de l'outil SMIL Presentation Wizard.



Sur ce schéma, on précise les avantages d'utilisation de SMIL Presentation Wizard

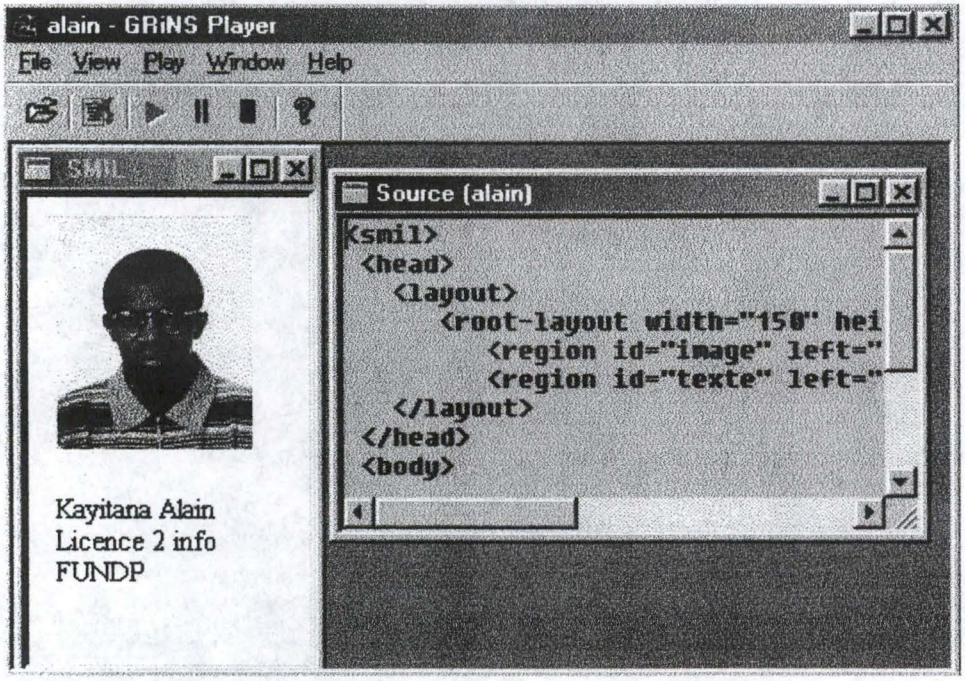
- la non l'utilisation de formats standards non optimisés pour le WWW, en utilisant des formats RealMédia(.rm),
- utilisation des médias se trouvant n'importe où sur le réseau,
- support de différents langages. Un exemple et la création d'une vidéo dont le son est en plusieurs langues(voir description de SMIL dans le chapitre 2),
- support de différents débits dans la présentation du document SMIL,
- support de l'édition par Timeline(voir chapitre 1),
- enfin le contrôle de la disposition spatiale dans la présentation.

3.2. CWI(Centroum voor Wiskunde en Informatica http://www.cwi.nl/GRINS/)

CWI est l'institut national de recherche en mathématique et informatique aux Pays-Bas. Actuellement il a développé un player et un éditeur de document SMIL. Le player est gratuit tandis que l'éditeur est payant.(Commercialisé par la société Oratrix : http://www.oratrix.com/GriNS)

3.2.1. GRINS Player

GRINS(**GR**aphical *i*Nterface to **SMIL**) est une appliaction de visualisation d'un document SMIL et de présentation multimédia sur l'Internet(SMIL).
Le GRINS Player est disponible sur les plates formes Windows et Unix.



L'interface comprend une fenêtre de visualisation qui s'adapte à la taille du « root-layout », et a un menu qui a les fonctions suivantes :
- jouer un document SMIL,
- arrêter un présentation
- montrer le code source du document SMIL.

La particularité de ce Player est qu'il permet de localiser les médias selon les méthodes d'accès définis par les balises suivantes :
- file
- http
- gopher
- ftp
- data

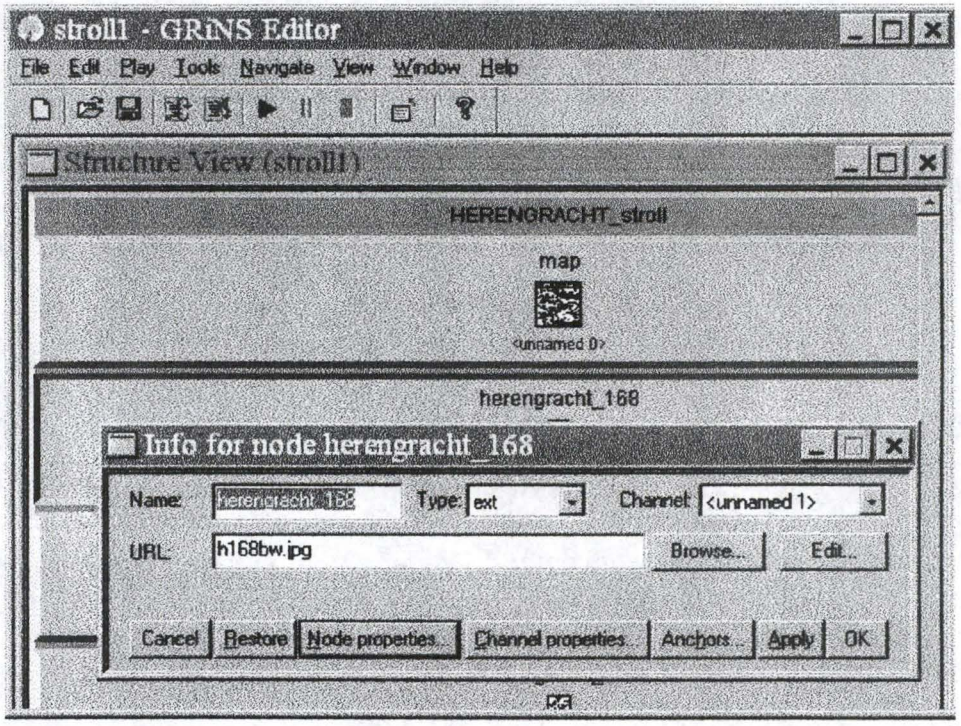
La plupart des formats standards des médias sont supportés par ce player.

Mais il existe certains spécifications SMIL nom supportés comme :

- l'attribut « begin » et « end » dans la balise « anchor » (voir Chapitre2),
- les couple attribus/valeurs fit="fill" et fit="scroll" dans la balise « region » ,
- le couple attribut/valeur name="pics-label" dans la balise meta,
- l'attribut alt dans les balises des objets multimédias.

3.2.1. GRINS Editor

L'editeur de document SMIL ressemble fort au player, mais il permet une édition graphique. On peut préciser les attributs pour chaque média qu'on incorpore dans la présentation, sa position spatiale, au lieu de le faire avec un éditeur de texte. Cela est très avantageux pour l'utilisateur qui ne maîtrise pas la syntaxe de SMIL car le texte source est généré automatiquement à la fin.
L'autre avantage est qu'on peut visualiser sa présentation au fur et à mesure qu'on construit son document SMIL.



Pour chaque objet, on peut consulter et changer ses attributs. Sur la figure ci dessus on a définie une sequence d'images et un texte. Chaque objet multimédia est répresenté par un gros bouton. On peut consulter et modifier les informations sur chaque média en cliquant sur le bouton qui le matérialise.

Le texte source qui est généré :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 1.0//EN"
                "http://www.w3.org/TR/REC-smil/SMIL10.dtd">
<smil>
 <body>

     <seq id="HERENGRACHT_stroll">
      <img id="map" src="map.jpg" dur="3s"/>
      <img id="herengracht_168" src="h168bw.jpg" dur="3s"/>
      <img id="herengracht_218" src="h218bw.jpg" dur="3s"/>
      <img id="herengracht_284" src="h284bw.jpg" dur="3s"/>
      <text type="text/plain" id="Credits" src="data:,Thanks to the
Amsterdam Heritage Site www.amsterdam.nl/bmz/adam" dur="indefinite"/>
     </seq>
 </body>
</smil>
```

3.3. SOJA Helio (http://www.helio.org)

SOJA veut dire « SMIL Output in JAVA Applets ».
C'est un applet JAVA qui permet de rendre des présentations SMIL dans toute page web avec des médias simples comme le texte ,l'image, le son. Il a été developé par HELIO.

Les principales fonctionnalités de SOJA sont:

- une meilleure synchronisation entre différents médias.
- support de fichier audio compressé au format(.auz),
- support des fichiers textes ascii,
- des liens,
- du SMIL-in-SMIL(présentation d'un document SMIL dans un autre document SMIL),
- enfin, des formats GIF, JPEG et AU.

Cet Applet fonctionne actuellement sur les plates formes Unix et Window. Il suffit d'incorporer cet applet dans un document HTML de la manière suivante:

```
<APPLET CODE="org.helio.soja.SojaApplet.class"
     ARCHIVE="soja.jar"
     CODEBASE = "path-to-soja/"
     WIDTH="300" HEIGHT="200">

<PARAM NAME="source" VALUE="path/sample.smil">
<PARAM NAME="display" VALUE="applet">

</APPLET>
```

où le document SMIL est passé comme paramètre.

3.4. S2M2(http:/ /smil.nist.gov/player/S2M2.html)

S2M2(Streaming Synchronized MultiMédia) est aussi un applet JAVA qui est encore à l'état d'un prototype de la NIST (National Institute of Standards and Technology). Cet applet peut être interfacé avec n'importe quel navigateur web pour visualiser un document SMIL.
S2M2 accepte plusieurs formats:

- l'audio(aiff, au, dvl, g723, gsm, ima4, mpeg-1layer1/2, pcm, rmf, wav),
- la vidéo(smc, rle, cinepack, h261, indeo3.2, motion jpeg, mpeg1),
- l'image(gif, jpeg),
- le texte(plain ascii file).

3.5. LpPlayer(http://www.labyrinten.se)

C' est un player de la société Labyrinten Data, qui supporte le langage SMIL pour une présentation multimédia.

3.6 Evaluations

Les outils de Real Networks, Inc paraissent être les mieux adaptés à l'édition et la présentation des documents SMIL sur le réseau, mais leur principal défaut est l'introduction des formats propriétaires qui ne sont pas accessibles par tout le monde. Cela s'explique par des raisons commerciales.

Les outils de CWI sont mieux adaptés pour la visualisation. L'édition paraît un peu difficile pour un novice, mais ces outils supportent la version actuelle de SMIL, en plus ils utilisent seulement des formats standards des objets multimédias.

Quant aux autres éditeurs, leurs produits sont presque semblables, ils ne font que visualiser un document SMIL dans un applet.

Chapitre 4. Architecture et conception de l'application

Dans ce chapitre, nous présentons le logiciel que nous avons implementé en JAVA. Nous allons décrire les différentes classes de l'application de visualisation de documents SMIL. Nous analyserons les composants de chaque classe, les relations de «hiérarchie » et «utilisé » des classes. Nous terminerons en précisant les outils utilisés pour le codage.

Dans les chapitres précédents, nous avions décrit le langage SMIL. Les trois principaux objectifs de ce langage sont :

- le positionnement des différents médias dans la fenêtre de visualisation,
- la synchronisation temporelle des médias,
- l'affichage de ces médias selon les caractéristiques de la machine, de la langue.

L'application a pour but la visualisation de documents SMIL restreints. En effet toutes les fonctionnalités d'un document SMIL ne seront pas présentes dans notre application.
Dans notre architecture nous allons nous limiter à implémenter les fonctions en rapport avec les deux premiers points cités en haut. L'ensemble des fonctionnalités prisés en charge est décrit dans l'annexe 1.

4.1 Analyse des besoins.

Le but principal de notre mémoire est d'implémenter un navigateur SMIL en JAVA. Ce logiciel doit permettre :

- d'analyser d'un document SMIL au moyen d'un parseur SMIL pour obtenir un graphe(arbre) qui représente la structure hiérarchique de ce document,
- d'utiliser le graphe obtenu pour afficher dans une fenêtre de visualisation le document SMIL en respectant les contraintes de positionnement(spécifiés dans l'en-tête du document) et de synchronisation des médias(spécifié par un scénario temporel décrit dans le corps du document).

L'utilisateur du logiciel doit pouvoir :

- ouvrir le fichier SMIL à visualiser,
- fermer le fichier SMIL dans l'application,
- avoir à sa disposition les fonctions pour jouer le fichier SMIL,
- arrêter la visualisation d'un fichier SMIL,
- rejouer ce fichier.

Nous allons nous limiter seulement à visualiser un seul fichier à la fois, une fois que l'application est ouverte.

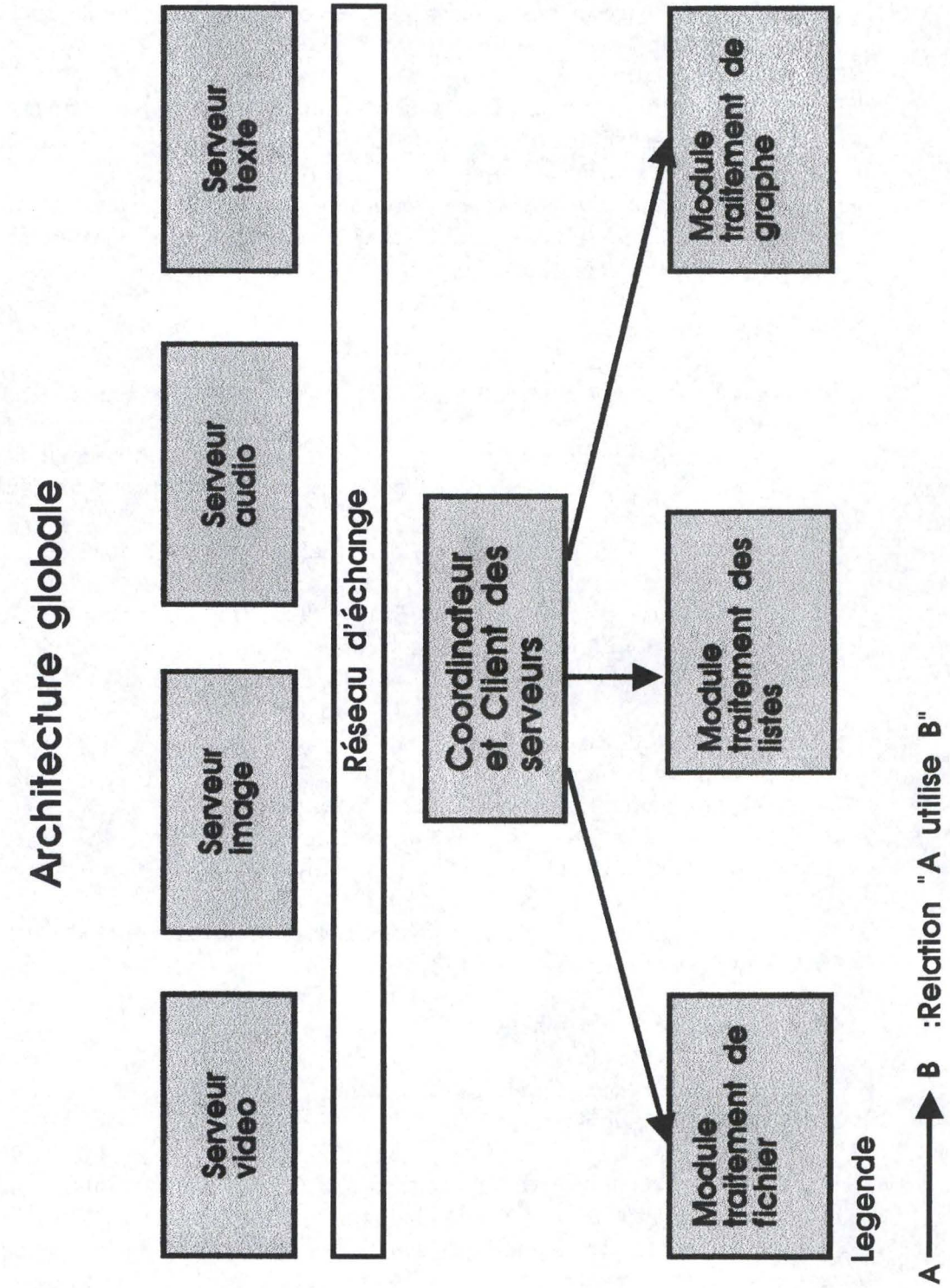
4.2 Architecture de l'application.

L'architecture de notre application présente deux aspects selon la conception architecturale des applications.

Le premier aspect est celui d'architecture client/serveur. Comme un document SMIL utilise plusieurs sortes de médias(vidéo, audio, texte, image), il y aura un serveur pour chaque média. Un module particulier de l'application jouera le rôle de client pour demander à l'un des serveurs de lui fournir les fonctionnalités nécessaires pour la gestion d'un média particulier.

Le second aspect est celui d'architecture basée sur la relation «utilise » Il y a un module principal qui doit utiliser les fonctionnalités des autres modules et faire le rôle de coordinateur entre différents modules.

Le graphe suivant montre l'architecture globale de l'application.



Le module traitement de fichier est chargé de gérer toutes fonctions en rapport avec les fichiers. Il s'agit de :
- l'ouverture,
- la fermeture,
- la lecture d'un fichier.

Ce module est chargé aussi de faire le filtrage des fichiers. Seulement les fichiers avec l'extension «.smil » ou «.SMIL » seront traités par l'application.

Le module traitement graphe est chargé de traiter toutes les fonctions en rapport avec le graphe qui représente la structure d'un document SMIL. La construction du graphe se fait au moment de l'ouverture du fichier SMIL.

Le module traitement de liste est chargé de gérer toutes les fonctions sur les listes d'objets. Il s'agit :
- d'accéder à un objet de la liste,
- d'ajouter un objet à la liste,
- de retire un objet à la liste,
- de donner la longueur de la liste,
- de tester si la liste est vide ou non vide.

Le module serveur est composé :
- d'un serveur vidéo chargé de la gestion d'un média de type vidéo,
- d'un serveur audio chargé de la gestion d'un média de type audio,
- d'un serveur image chargé de la gestion d'un média de type image,
- d'un serveur texte chargé de la gestion d'un média de type texte.

Le module coordinateur et client des serveurs est chargé de la gestion de l'interaction des différents modules, et l'interaction avec l'utilisateur de l'application.

4.3 Découpe en packages

A chaque module de l'application, correspond un package. Chaque package regroupe les classes qui implémentent les fonctionnalités de ce module.

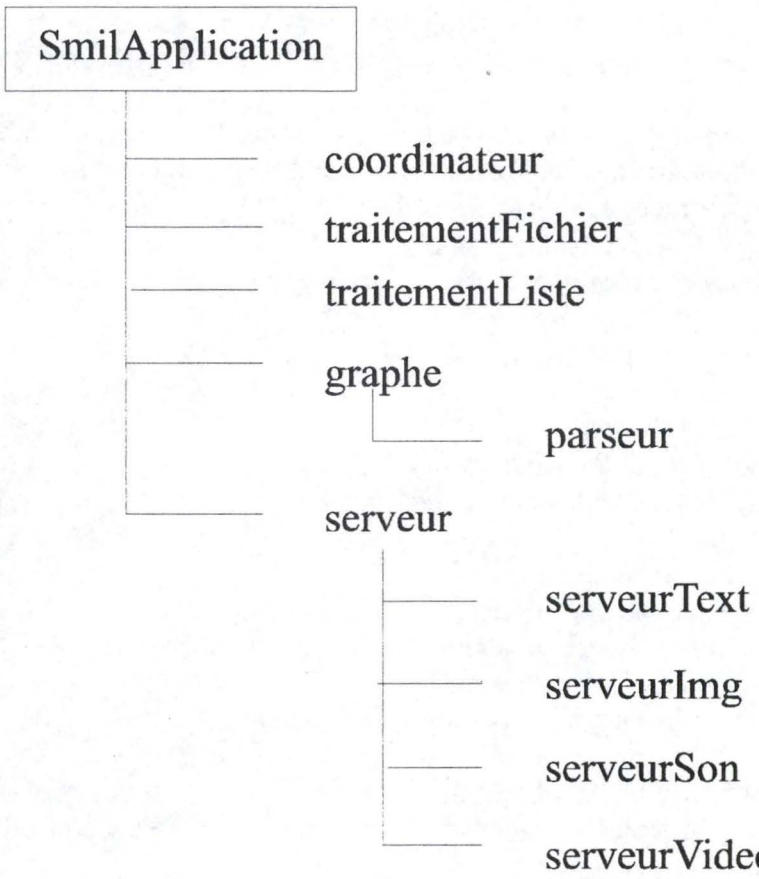


Figure 11 : Hiérarchie des packages

Nous analyserons au fur et à mesure les différents packages. Pour chaque package nous décrirons ses classes. La description de chaque classe se fait en précisant le package, le nom, les données et les services offerts aux autres modules.

La description utilise le schéma suivant :

Non du package : **Nom de la classe :** Explication si nécessaire	
Données de la classe	
Nom de donnée :	Explication si nécessaire
Services offerts	
Nom du service :	Explication si nécessaire

4.3.1 Package serveur

Le package serveur contient les classes des différents serveurs de médias.
Tous les serveurs héritent les fonctions génériques de la classe serveur(voir Figure 12). Les services communs à tous les serveurs sont regroupés dans la classe Serveur. Les services de la classe Serveur permettent de gérer les informations de chaque média sans savoir le type de ce média. Le package serveur possède les sous packages serveurVideo, serveurSon, serveurImg et serveurText qui prennent en charge réspectivement les médias de type vidéo, audio, image et texte.

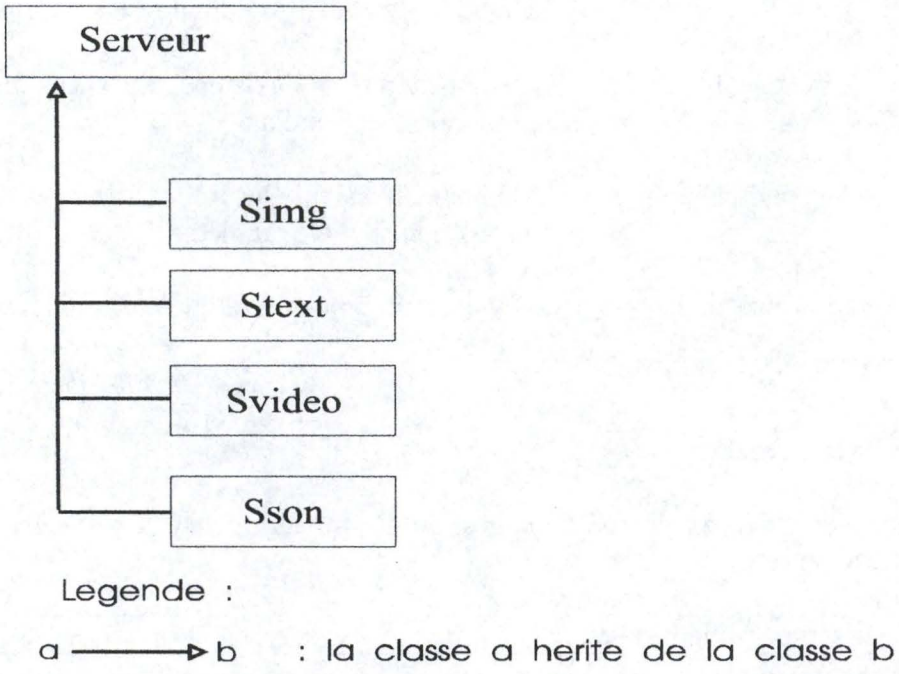


Figure 12 : Hiérachie des classes de serveurs

serveur:**Serveur**	
Données de la classe	
Média m	Explication dans la classe Média
String nomfile	Nom de l'URL du média
SmilApplication f	Explication dans la classe SmilApplication
int left int top int width int height	Les entiers left, top, width et height contiennent les informations sur la disposition spatiale pour afficher le média m
String dur String begin String end	Les chaînes de caractères dur, begin et end contiennent respectivement les informations sur la durée intrinsèque du média, l'instant de début et l'instant de fin du média.

Services offerts	
Serveur(Média média, SmilApplication frame)	Constructeur de la classe serveur et a comme argument un média et la référence à la classe SmilApplication.
getDur()	Méthode qui donne la durée du média.
getBegin()	Méthode qui donne l'instant de début du média.
getEnd()	Méthode qui donne l'instant de fin du média.
getSrc()	Méthode qui donne le nom de l'URL du média
regionAffichage()	Méthode qui teste l'existence de la région où on va afficher le média.
getLeftTopWidthHeight()	Méthode qui calcule les informations sur la disposition spatiale du média.
int convertirmseconde()	Méthode qui convertit la durée d'un média en milliseconde.

A. Serveur vidéo

Le serveur vidéo est chargé de la gestion d'un média vidéo. Il hérite les services de la classe serveur.

serveur.serveurVideo:**Svideo**	
Données de la classe	
Player m	Explication dans l'API JAVA Média Framework
Services offerts	
Svideo(Video video, SmilApplication frame)	Constructeur de la classe média vidéo. Le constructeur a comme arguments un média vidéo et la référence à la classe SmilApplication.
GetVideo()	Méthode qui charge le média vidéo pour la présentation.
Comencer()	Méthode qui commence la présentation du flux du média vidéo.
arreter()	Méthode qui arrête la présentation du flux du média vidéo.
detruire()	Méthode qui détruit le flux du média vidéo.

B. Serveur audio

Le serveur audio est chargé de la gestion d'un média audio. Il hérite les services de la classe serveur.

Serveur.serveurSon:**Sson**	
Données de la classe	
Player m	Explication dans l'API JAVA Média Framework
Services offerts	
Sson(Son son, SmilApplication frame)	Constructeur du serveur son et a comme arguments un média audio et la référence à la classe SmilApplication.
GetSon()	Méthode qui charge le média audio pour la présentation.
Comencer()	Méthode qui commence la présentation du flux du média audio.
Arreter()	Méthode qui arrête la présentation du flux du média audio.
Detruire()	Méthode qui détruit le flux du média audio.

C. Serveur image

Le serveur image est chargé de la gestion d'un média image. Il hérite les services de la classe serveur.

Serveur.serveurImg:**Simg**	
Données de la classe	
Image m	Variable qui contient une image à présenter.
Services offerts	
Simg(Img img, SmilApplication frame)	Constructeur du serveur image et a comme arguments un média image et la référence à la classe SmilApplication.
GetImg()	Méthode qui charge le média image pour la présentation.

ListeRemove(int i)	Méthode qui enlève l'élément de la liste situe à la position i.
int ListeLength()	Méthode qui donne la longueur de la liste.

TraitementListe:**ListeInt**	
Données de la classe	
Services offerts	
ListeInt()	Constructeur de la classe Liste d'entier vide.
ListeIntSet(int i,int elem)	Méthode qui ajoute un entier à une liste à la position i.
LisetIntGet(int i)	Méthode qui donne l'entier de la liste à la position i.

TraitementListe:**ListeMédia**	
Données de la classe	
Services offerts	
ListeMédia()	Constructeur de la classe Liste de Média vide. (Média est définie dans la classe Média)
ListeMédiaSet(int i,Média elem)	Méthode qui ajoute un Média à une liste à la position i.
LisetMédiaGet(int i)	Méthode qui donne le Média de la liste à la position i.

TraitementListe:**ListeMeta**	
Données de la classe	
Services offerts	
ListeMeta()	Constructeur de la classe Liste de Meta vide. (Meta est définie dans la classe Média)
ListeMetaSet(int i,Meta elem)	Méthode qui ajoute un Meta à une liste à la position i.
LisetMetaGet(int i)	Méthode qui donne le Meta de la liste à la position i.

B. Serveur audio

Le serveur audio est chargé de la gestion d'un média audio. Il hérite les services de la classe serveur.

Serveur.serveurSon:**Sson**	
Données de la classe	
Player m	Explication dans l'API JAVA Média Framework
Services offerts	
Sson(Son son, SmilApplication frame)	Constructeur du serveur son et a comme arguments un média audio et la référence à la classe SmilApplication.
GetSon()	Méthode qui charge le média audio pour la présentation.
Comencer()	Méthode qui commence la présentation du flux du média audio.
Arreter()	Méthode qui arrête la présentation du flux du média audio.
Detruire()	Méthode qui détruit le flux du média audio.

C. Serveur image

Le serveur image est chargé de la gestion d'un média image. Il hérite les services de la classe serveur.

Serveur.serveurImg:**Simg**	
Données de la classe	
Image m	Variable qui contient une image à présenter.
Services offerts	
Simg(Img img, SmilApplication frame)	Constructeur du serveur image et a comme arguments un média image et la référence à la classe SmilApplication.
GetImg()	Méthode qui charge le média image pour la présentation.

D. Serveur texte

Le serveur texte est chargé de la gestion d'un média texte. Il hérite les services de la classe serveur.

Serveur.serveurText:**Stext**	
Données de la classe	
String[] text int iligne	Tableau qui contient les lignes du texte à présenter. Le nombre de ligne du texte à présenter.
Services offerts	
Stext(Text text, SmilApplication frame) getText()	Constructeur du serveur Texte et a comme arguments un média texte et la référence à la classe SmilApplication. Méthode qui charge le média texte pour la présentation.

4.3.2 Package traitementListe

Le package traitementListe contient toutes les classes en rapport avec le traitement des listes d'objets. Ce module est chargé de la gestion des fonctions des listes. Ces fonctions sont génériques car on peut avoir une liste d'objets quelconques(Figure13). Il y avait moyen d'utiliser une classe qui est déjà présente dans le Kit de JAVA, mais faute de temps nous n'avons pas pu changer le code de l'application.

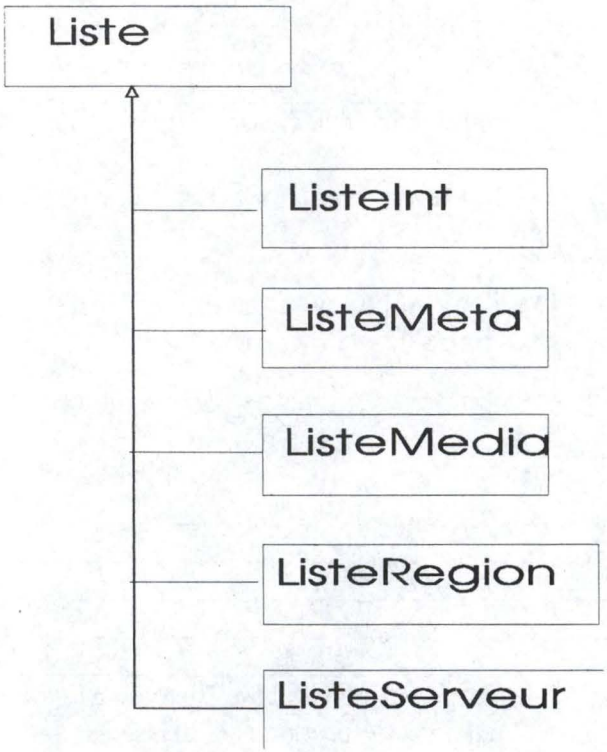


Figure 13 : Hiérarchie des classes de listes

TraitementListe:**Liste**	
Données de la classe	
Object m	Objet quelconque de la liste.
Liste next Liste tete	La tête et l'élément suivant de la liste.
Services offerts	
Liste()	Constructeur de la classe Liste vide.
ListeSet(int i,Object elem)	Méthode qui ajoute un objet à une liste à la position i.
LisetGet(int i)	Méthode qui donne l'élément de la liste à la position i.

ListeRemove(int i)	Méthode qui enlève l'élément de la liste situe à la position i.
int ListeLength()	Méthode qui donne la longueur de la liste.

TraitementListe:**ListeInt**	
Données de la classe	
Services offerts	
ListeInt()	Constructeur de la classe Liste d'entier vide.
ListeIntSet(int i,int elem)	Méthode qui ajoute un entier à une liste à la position i.
LisetIntGet(int i)	Méthode qui donne l'entier de la liste à la position i.

TraitementListe:**ListeMédia**	
Données de la classe	
Services offerts	
ListeMédia()	Constructeur de la classe Liste de Média vide. (Média est définie dans la classe Média)
ListeMédiaSet(int i,Média elem)	Méthode qui ajoute un Média à une liste à la position i.
LisetMédiaGet(int i)	Méthode qui donne le Média de la liste à la position i.

TraitementListe:**ListeMeta**	
Données de la classe	
Services offerts	
ListeMeta()	Constructeur de la classe Liste de Meta vide. (Meta est définie dans la classe Média)
ListeMetaSet(int i,Meta elem)	Méthode qui ajoute un Meta à une liste à la position i.
LisetMetaGet(int i)	Méthode qui donne le Meta de la liste à la position i.

TraitementListe:**ListeRegion**	
Données de la classe	
Services offerts	
ListeRegion()	Constructeur de la classe Liste de Region vide. (Region est définie dans la classe Region)
ListeRegionSet(int i,Region elem)	Méthode qui ajoute une Region à une liste à la position i.
LisetRegionGet(int i)	Méthode qui donne la Region de la liste à la position i.

TraitementListe:**ListeServeur**	
Données de la classe	
Services offerts	
ListeServeur()	Constructeur de la classe Liste de Serveur vide.
ListeServeurSet(int i,Média elem)	(Serveur est défini dans la classe Serveur) Méthode qui ajoute un Serveur à une liste à la position i.
LisetServeurGet(int i)	Méthode qui donne le Serveur de la liste à la position i.

4.3.3 Package graphe

Le package graphe comprend toutes les classes en rapport avec le graphe du document SMIL. Ce graphe représente la structure d'un document SMIL. Le package graphe contient un sous package parseur qui est chargé de l'analyse lexicale et syntaxique d'un document SMIL[15]. La partie du Langage SMIL prise en charge dans le cadre de ce mémoire se trouve dans l'annexe1. La grammaire de ce langage se trouve dans l'annexe 2. Les différentes structures que nous allons décrire sont en rapport avec les productions de la grammaire cité ci dessus.

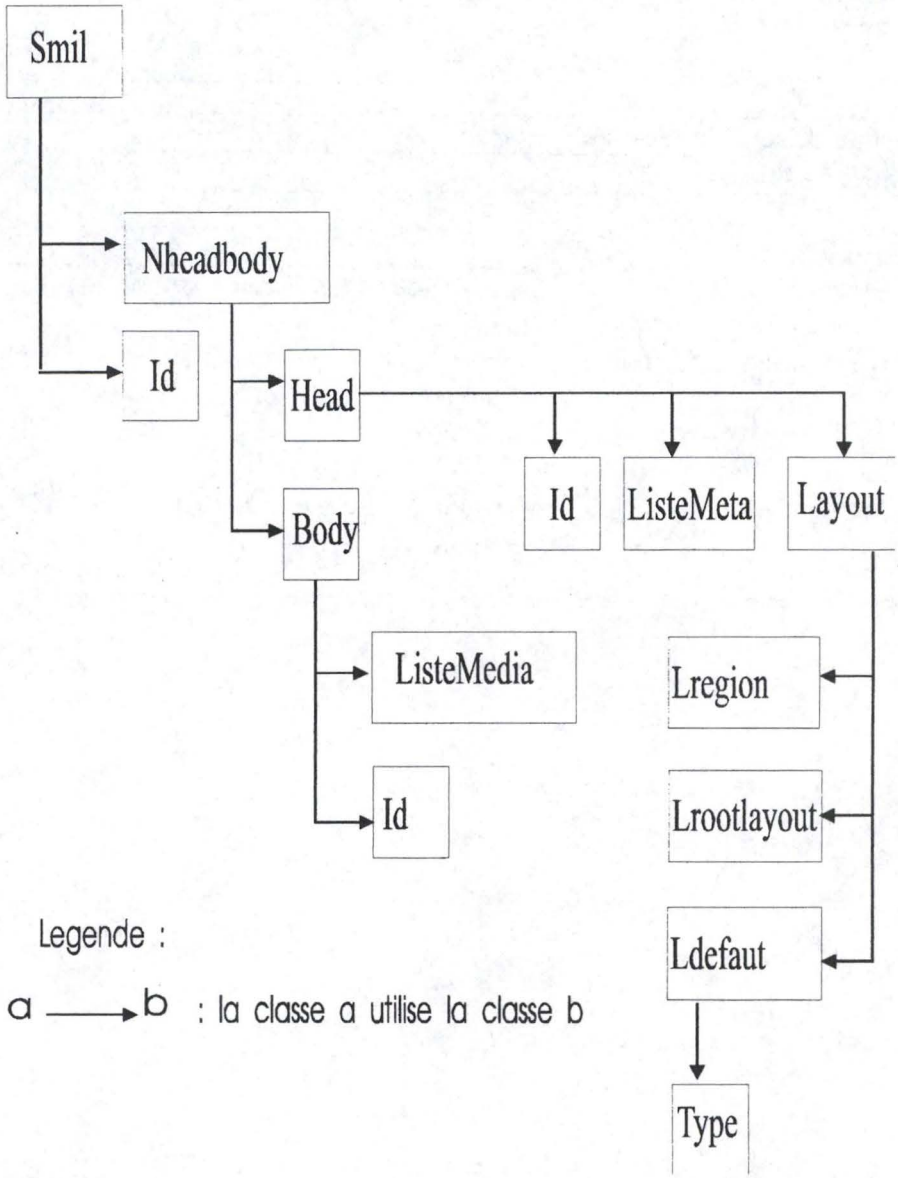


Figure 14 : Relation « utilise » de l'application

---

[15] Pour ce qui regarde l'analyseur lexical et syntaxique, ainsi que les vocabulaires utilisés, nous renvoyons le lecteur à l'utilisation de ces outils : http://www.princeton.edu/appel/modern/java/CUP et http://www.jflex/de

Dans la suite nous allons décrire les différentes classes selon le schéma ci dessus.

graphe:**Smil:** Définit le document Smil.	
Données de la classe	
Id s_id Nheadbody s_hb	Définie dans la classe Id. Définie dans la classe Nheadbody.
Services offerts	
Smil(Id id,Nheadbody hb)	Constructeur de la classe Smil.

graphe:**Id**	
Données de la classe	
String i_id	Donne l'information sur l'identificateur de la classe. (média ou region ou root layout, meta,head, body)
Services offerts	
Id(String ident) Id()	Constructeurs de la classe Id.

graphe:**Nheadbody:** Contient les informations de l'en-tête et le corps	
Données de la classe	
Head n_hd Body n_bd	Définie dans la classe Head Définie dans la classe Body.
Services offerts	
Nheadbody(Head hd, Body bd) Nheadbody(Body bd)	Constructeur de la classe Nheadbody.

graphe:**Body :** Contient les informations sur le corps d'un document SMIL.	
Données de la classe	
ListeMédia b_body; Id b_id;	Définie dans la classe ListeMédia. Définie dans la classe Id.
Services offerts	
Body(Id id,ListeMédia liste)	Constructeur de la classe Body.

La classe ListeMédia est développée au point 4.3.3.a.

graphe:**Head :** Donne les informations sur l'en-tête du documentSMIL.	
Données de la classe	
Id h_id ListeMeta h_meta Layout h_lay	Définie dans la classe Id. Définie dans la classe ListeMeta. Définie dans la classe Layout.
Services offerts	
Head(Id id, ListeMeta smeta,Layout lay) Head(Id id,Layout lay) Head(Id id,ListeMeta smeta)	Constructeurs de la classe Head.

La classe ListeMeta est développée au point 4.3.3.b

graphe:**Layout :** Définit de la disposition spatiale	
Données de la classe	
int l_choix Lregion l_reg Lrootlayout l_root; Ldefaut l_def;	Information sur le choix de disposition spatiale. Définie dans la classe Lregion. Définie dans la classe Lootlayout. Définie dans la classe Ldefaul.
Services offerts	
Layout(Lregion lreg, int choix) Layout(Lrootlayout root, int choix) Layout(Ldefaut def,int choix)	Constructeurs de la classe Layout.

graphe:**Ldefaut :** définit de la disposition spatiale par défaut	
Données de la classe	
Type l_type	Définie dans la classe Type.
Services offerts	
Ldefaut(Type type)	Constructeur de la classe Ldefaut.

La classe Lregion est développée au point 4.3.3.c
La classe Lrootlayout est développée au point 4.3.3.d

4.3.3.a Développement de la classe ListeMédia

La classe ListeMédia est définie dans le package traitementListe, mais elle utilise les classes définies dans le package graphe. Nous allons décrire les classes qu'elle utilise selon le graphe ci-dessous.

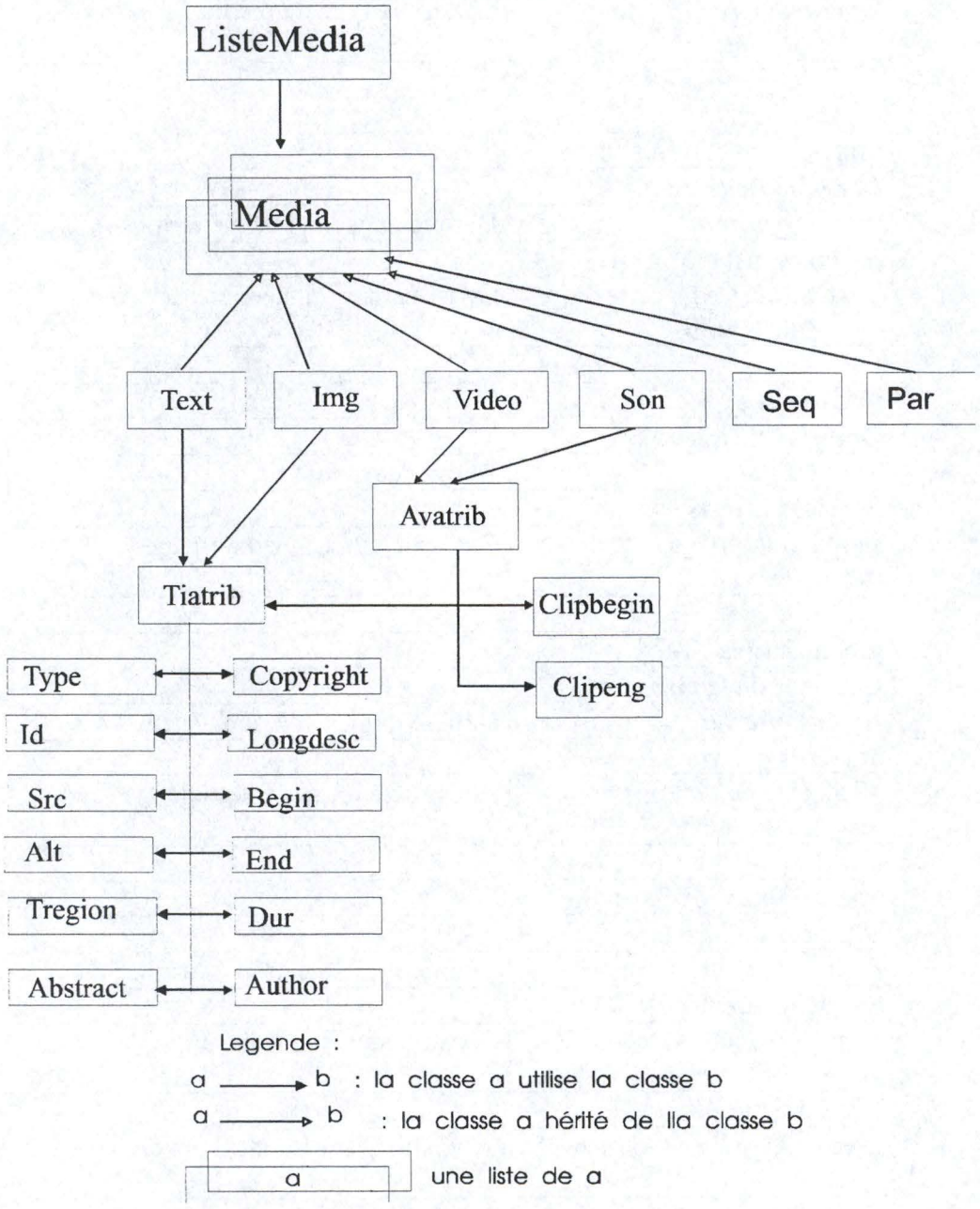


Figure 15 : Hiérarchie des médias

graphe:**Média:** Définit d'un média.	
Données de la classe	
String typemédia	Donne le type d'un média.
Services offerts	
typeMédia()	Change le type de média.

graphe:**Text :** Média texte	
Données de la classe	
Tiatrib t_atrib	Définie dans la classe Tiatrib.
Services offerts	
Text(String text, Tiatrib atrib)	Constructeur du média texte.

graphe:**Img :** Média image	
Données de la classe	
Tiatrib i_id	Définie dans la classe Tiatrib.
Services offerts	
Img(String image, Tiatrib atrib)	Constructeur du média image.

graphe:**Video :** Média video	
Données de la classe	
Avatrib s_atrib	Définie dans la classe Avatrib.
Services offerts	
Video(String video, Avatrib atrib)	Constructeur du média video.

graphe:**Son :** Média audio	
Données de la classe	
Avatrib s_atrib	Définie dans la classe Avatrib.
Services offerts	
Son(String son, Avatrib atrib)	Constructeur du média audio.

graphe:**Avatrib :** Contient les informations sur les attributs du média audio ou video	
Données de la classe	
Clipbegin a_begin; Clipend a_end; Tiatrib a_atrib;	Definie dans la classe Clipbegin. Definie dans la classe Clipend. Definie dans la classe Tiatrib.
Services offerts	
Avatrib(Tiatrib atrib, Clipbegin begin, Clipend end)	Constructeur de la classe Avatrib.

graphe:**Clipbegin**	
Données de la classe	
String c_name	Donne l'information sur l'instant de début d'un média par rapport à un autre média.
Services offerts	
Clipbegin(String ident) Clipbegin()	Constructeurs de la classe Clipbegin.

graphe:**Clipend**	
Données de la classe	
String c_name	Donne l'information sur l'instant de fin d'un média par rapport à un autre média.
Services offerts	
Clipend(String ident) Clipend()	Constructeurs de la classe Clipend.

graphe:**Tatrib :** Contient les informations sur les attributs du média image ou text	
Données de la classe	
Type t_tp Id t_id Src t_src Alt t_top Tregion t_region Title t_title Abstract t_abs Author t_author Copyright t_copy Longdesc t_desc Begin t_begin End t_end Dur t_dur	Définie dans la classe Type. Définie dans la classe Id. Définie dans la classe Src. Définie dans la classe Alt. Définie dans la classe Tregion. Définie dans la classe Title. Définie dans la classe Abstract. Définie dans la classe Author. Définie dans la classe Copyright. Définie dans la classe Longdesc. Définie dans la classe Begin. Définie dans la classe End. Définie dans la classe Dur.
Services offerts	
Tiatrib(Type tp,Id id, Tregion region,Src src, Alt top,Title title, Abstract abs,Authorauthor, Copyright copy, Longdesc desc,Begin begin, End end,Dur dur)	Constructeur de la classe Tiatrib.

Les classes suivantes sont utilisés par la classe Tiatrib.

graphe:**Type**	
Données de la classe	
String t_id	Donne le type d'un texte pour un média texte.
Services offerts	
Type(String ident) Type()	Constructeurs de la classe Type.

graphe:**Copyright**	
Données de la classe	
String a_name	Donne l'information sur le copyright du document.
Services offerts	
Copyright(String ident) Copyright()	Constructeurs de la classe Copyright.

graphe:**Longdesc**	
Données de la classe	
String d_id	Donne la description d'un média.
Services offerts	
Longdesc(String ident) Longdesc()	Constructeurs de la classe Longdesc.

graphe:**Src :** Définit l'URL d'un Média	
Données de la classe	
Strin s_name	Donne le nom de l'url.
Services offerts	
Src(String ident)	Constructeur de la classe url.

graphe:**Alt**	
Données de la classe	
String a_name	Donne la chaîne à afficher si le média à afficher n'est pas disponible.
Services offerts	
Alt(String ident) Alt()	Constructeurs de la classe Alt.

graphe:**Begin**	
Données de la classe	
String b_name	Donne l'information sur l'instant de début d'un média.
Services offerts	
Begin(String ident) Begin()	Constructeurs de la classe Begin.

graphe:**End**	
Données de la classe	
String e_name	Donne l'information sur l'instant de fin d'un média.
Services offerts	
End(String ident) End()	Constructeurs de la classe End.

graphe:**Tregion**	
Données de la classe	
String t_name	Référence à une région définie dans l'en-tête du document SMIL
Services offerts	
Tregion(String ident) Tregion()	Constructeurs de la classe Tregion.

graphe:**Dur**	
Données de la classe	
String d_name	Donne l'information sur la durée d'un média copyright du document.
Services offerts	
Dur(String ident) Dur()	Constructeurs de la classe Dur.

graphe:**Abstract**	
Données de la classe	
String a_name	Donne la description sur les scénarios temporels.
Services offerts	
Abstract(String ident) Abstract()	Constructeurs de la classe Abstract.

graphe:**Author**	
Données de la classe	
String a_name	Donne l'information sur l'auteur du document.
Services offerts	
Author(String ident) Author()	Constructeurs de la classe Auteur.

graphe:**Seq:** Définit un scénario temporel sequentiel.	
Données de la classe	
ListeMédia p_par Fseqpar p_s	Définie dans la classe ListeMédia. Définie dans la classe Fseqpar.
Services offerts	
Seq(String par, Fseqpar s,ListeMédia liste)	Constructeur de la classe Seq.

graphe:**Par:** Définit un scénario temporel parallèle.	
Données de la classe	
ListeMédia p_par Fseqpar p_s	Définie dans la classe ListeMédia. Définie dans la classe Fseqpar.
Services offerts	
Par(String par, Fseqpar s,ListeMédia liste)	Constructeur de la classe Par.

graphe:**Reapet:** Définit le nombre de répétition d'un média dans la présentation.	
Données de la classe	
String r_name	Donne le nombre de fois que le média est joué ou présenté.
Services offerts	
Reapet(String ident)	Constructeur de la classe Reapet.

graphe:**Fseqpar :** Information sur les scénarios temporels.	
Données de la classe	
Id f_id Abstract f_ab Author f_auth Copyright f_cop Begin f_big End f_end Dur f_dur Reapet f_rep	Définie dans la class Id. Définie dans la class Abstract. Définie dans la class Author. Définie dans la class Copyright. Définie dans la class Begin. Définie dans la class End. Définie dans la class Dur. Définie dans la class Reapet.
Services offerts	
Fseqpar(Id id, Abstract ab, Author auth, Copyright cop, Begin big, End end, Dur dur, Reapet rep)	Constructeur de la classe Fseqpar.

4.3.3.b Développement de la classe ListeMeta

La classe listeMeta est définit dans le package traitement de liste, mais elle utilise les classes du package graphe. Nous allons décrire les classes qu'elle utilise selon le schéma suivant.

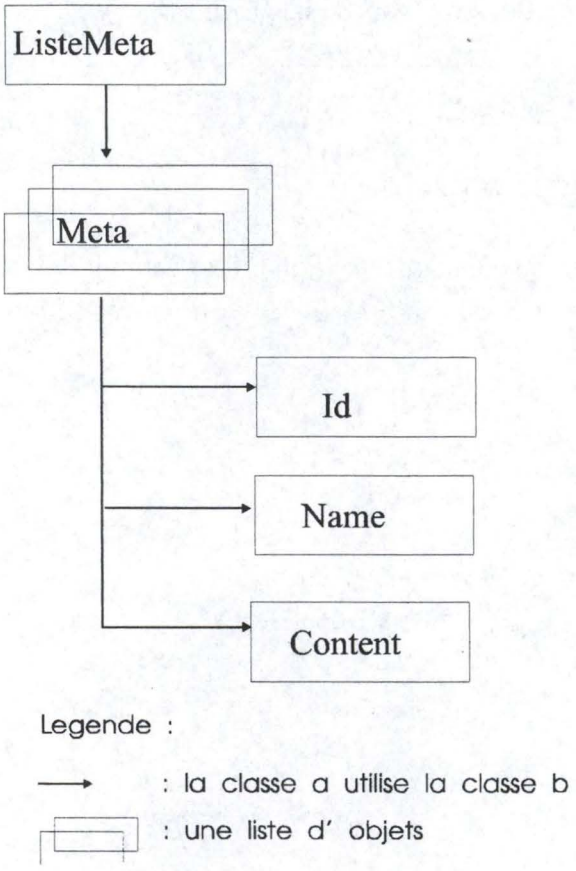


Figure 16 : Hiérachie de Meta

graphe:**Meta:** Définit d'un méta.	
Données de la classe	
Id m_id Name m_name Content m_content	Définie dans la classe Id Définie dans la classe Name. Définie dans la classe Content.
Services offerts	
Meta(Id id,Name name, Content content)	Constructeur de la classe Meta.

graphe:**Name:** Définit le nom d'un objet.	
Données de la classe	
String n_name	Donne le nom de la classe.
Services offerts	
Name(String ident)	Constructeur de la classe Name.

graphe:**Content**	
Données de la classe	
String c_ident	Donne l'information sur le contenue d'un média s'il est disponible.
Services offerts	
Content(String ident) Content()	Constructeurs de la classe Content.

4.3.3.c Développement de la classe Lregion.

Nous allons décrire les classes que la classe Lregion utilise selon le schéma suivant.

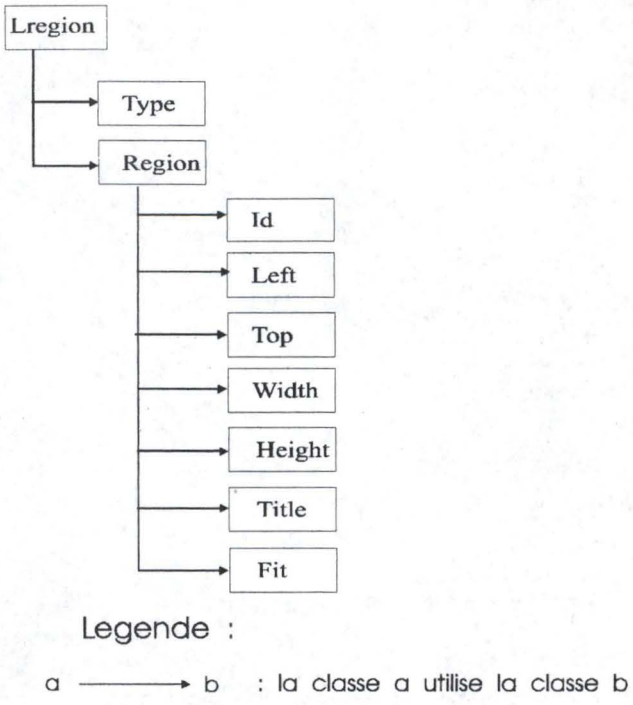


Figure 17 : Hiérachie de Lregion

graphe:**Lregion :** Donne l'information sur la disposition spatiale.	
Données de la classe	
Type l_type Region l_reg	Définie dans la classe Type Definie dans la classe Region
Services offerts	
Lregion(Type type, Region ident)	Constructeur de la classe Lregion.

graphe:**Region:** Définie un espace de présentation ou sera présenté ou sera affiché un média.	
Données de la classe	
Id r_id Left r_left Top r_top Width r_width Height r_height Title r_title Fit r_fit	Définie dans la classe Id. Définie dans la classe Left. Définie dans la classe Top. Définie dans la classe Width. Définie dans la classe Height. Définie dans la classe Title. Définie dans la classe Fit.
Services offerts	
Region(Id id,Title title, Left left,Top top, Width width,Height height, Fit fit)	Constructeur de la classe Region.

graphe:**Left**	
Données de la classe	
String l_id	Donne l'information sur l'abscisse d'un objet (region)
Services offerts	
Left(String ident) Left()	Constructeurs de la classe Left.

graphe:**Top**	
Données de la classe	
String t_id	Donne l'information sur l'ordonne d'un objet (région)
Services offerts	
Top(String ident) Top()	Constructeurs de la classe Top.

graphe:**Width**	
Données de la classe	
String w_id	Donne l'information sur la largeur d'un objet (region ou root layout)
Services offerts	
Width(String ident) Width()	Constructeurs de la classe Width.

graphe:**Height**	
Données de la classe	
String h_id	Donne l'information sur la hauteur d'un objet (region ou root layout)
Services offerts	
Height(String ident) Height()	Constructeurs de la classe Height.

graphe:**Title**	
Données de la classe	
String t_name	Donne le titre d'un objet.
Services offerts	
Title(String ident) Title()	Constructeurs de la classe Title.

graphe:**Fit**	
Données de la classe	
String e_name	Donne l'information sur le remplissage du média dans sa région d'affichage.
Services offerts	
Fit(String ident) sefit()	Constructeur de la classe End. Changement de l'information de remplissage.

4.3.3.d Développement de la classe Lrootlayout

Nous allons décrire les classes que la classe Lrootlayout utilise selon le schéma suivant.

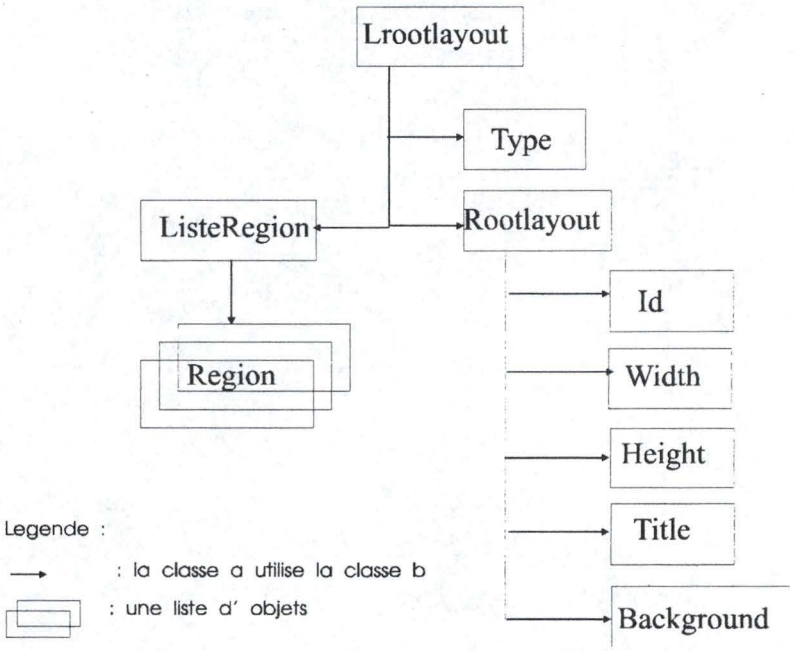


Figure 18 : Hiérachie de Lrootlayout.

graphe:**Lrootlayout :** Donne l'information sur la disposition spatiale de type rootlayout	
Données de la classe	
Type l_type Rootlayout l_root	Définie dans la classe Type. Définie dans la classe Rootlayout.

ListeRegion l_reg	Définie dans la classe ListeRegion.
Services offerts	
Lrootlayout(Type type, Rootlayout root, ListeRegion l)	Constructeur de la classe Lrootlayout.

La classe ListeRegion est defini dans le package traitementListe, mais elle utilise les classes du package graphe. Elle utilise la classe Region qui est décrit au 4.3.3.c.

graphe:**Rootlayout:** Définie une fenêtre de présentation du document SMIL.	
Données de la classe	
Id r_id Width r_width Height r_height Title r_title Background r_back	Définie dans la classe Id. Définie dans la classe Width. Définie dans la classe Height. Définie dans la classe Title. Définie dans la classe Background.
Services offerts	
Rootlayout(Id id,Title title, Width width,Height height, Background back)	Constructeur de la classe Rootlayout.

graphe:**Background**	
Données de la classe	
String b_id	Donne l'information sur le fond de la fenêtre de présentation.
Services offerts	
Background(String ident) Background ()	Constructeurs de la classe Background.

4.3.3.e Sous package parseur

Le sous packagee parseur contient les classes en rapport avec l'analyseur lexicale et syntaxique. Les classes Lexer, parser, sym et Scanner ont été produites automatiquement par les outils d'analyse à partir des fichiers de spécifications smil.flex et smil.cup dans l'annexe2. La seule classe produite dans ce travail est :

parseur:**ParseurSmil**	
Données de la classe	
parser p SmilApplication f	Définie dans la classe parser. Définie dans la classe SmilApplication.
Services offerts	
ParseurSmil(String fichiersmil, SmilApplication ff)	Constructeur de la classe ParseurSmil.

Les méthodes suivantes servent à traiter la partie du graphe du document SMIL(l'en-tête), pour en extraire les informations générales sur la disposition spatiale.	
TraiteSmil(Smil s)	Méthode qui traite la racine du graphe représentant le document SMIL
TraiteId(Id i,String key)	Méthode qui traite la classe ID
TraiteHead(Head h)	Méthode qui traite la classe Head
TraiteNlayout(Layout m)	Méthode qui traite la classe Layout
TraiteSuiteMeta(ListeMeta t)	Méthode qui traite la classe ListeMeta
TraiteNregion(Lregion m)	Méthode qui traite la classe Lregion
TraiteNldefaut(Ldefaut m)	Méthode qui traite la classe Ldefaut
TraiteNrootlayout(Lrootlayout m)	Méthode qui traite la classe Lrootlayout
TraiteMeta(Meta m, String s)	Méthode qui traite la classe Meta
TraiteType(Type m,String s)	Méthode qui traite la classe Type
TraiteRegion(Region tl, String s)	Méthode qui traite la classe Region
TraiteRootlayout(Rootlayout m)	Méthode qui traite la classe Rootlayout
TraiteListeRegion(ListeRegion m)	Méthode qui traite la classe ListeRegion
Les méthodes suivantes servent à traiter la partie du graphe du document SMIL(le corps),pour en extraire les informations sur les scénarios temporels du document.	
TraiteBody(Body body)	Méthode qui traite le corps du document SMIL
TraiteScénario(ListeMédia l)	Méthode qui traite les scénarios temporels
TraiteMédia(Média m)	Méthode qui traite la classe Média.
TraiteImg(Img m)	Méthode qui traite la classe Img.
TraiteText(Text m)	Méthode qui traite la classe Text.
TraiteVideo(Video m)	Méthode qui traite la classe Video.
TraiteSon(Son m)	Méthode qui traite la classe Son.
TraiteSeq(Seq m)	Méthode qui traite le scénario séquentiel
TraitePar(Par m)	Méthode qui traite le scénario parallèle

4.3.4 Package traitementFichier

Le package traitementFichier contient les classes qui implémentent toutes les fonctions en rapport au traitement des fichiers.

traitementFichier:**FenetreSource**	
Données de la classe	
SmilApplication parent	Explication de classe SmilApplication.
Services offerts	
FenetreSource(SmilApplication parent, String titre, boolean modale)	Constructeur de la fenêtre qui contient la source du document SMILl et a comme la référence à la classe SmilApplication, le titre de la fenêtre et l'argument modale qui indique la manière dont cette fenêtre sera visible par rapport à la fenêtre principale de l'application(parent).
lirefichier(String nomfile)	Méthode qui lit le fichier ligne par ligne et l'affiche dans la fenêtre d'affichage de la source du document SMIL.

traitementFichier:**SmilFilter**	
Données de la classe	
Class **UtilsSmil**	Cette classe donne l'extension d'un fichier.
Services offerts	
accept(File f)	Méthode qui accepte la visibilité d'un fichier dans l'arborescence de fichiers selon son extension.

4.3.5 Package coordinateur

Le package coordinateur contient les classes suivantes :

a. classe de gestion des canaux de présentation

La classe de gestion des canaux de présentation se charge de présenter ou d'afficher un document SMIL. Il se charge de la mise à jour de tous les changements au cour de la présentation et gère les différents scénarios temporels

Coordinateur:**PanelSmil**	
Données de la classe	
Timer timer	Définie tous les événements temporels (durée, instant de début et instant de fin).
SmilApplication f	Définie dans la classe SmilApplication.
Simg simg	Informations sur un serveur image
Stext stext	Informations sur un serveur texte
Svideo svideo	Informations sur un serveur vidéo
Son son	Informations sur un serveur audio
int choix	Détermine quel serveur il s'agit.
boolean visible	Donne l'information sur l'état d'affichage des informations d'un serveur.
Services offerts	
PanelSmil(Simg sim, SmilApplication frame) PanelSmil(Stext sxt, SmilApplication frame) PanelSmil(Svideo v, SmilApplication frame) PanelSmil(Sson s, SmilApplication frame)	Constructeurs de la classe PanelSmil.

b.classe de gestion des événements de la barre de menu.

Coordinateur:**GestionMenu**	
Données de la classe	
SmilApplication f	Définie dans la classe SmilApplication.
Services offerts	
GestionMenu(SmilApplication frame) actionPerformed(ActionEvent evt)	Constructeur de la classe GestionMenu. Traite les différents événements générés par la barre de menu.

c. classe de gestion des événements du toolbar.

Coordinateur:**GestionTool**	
Données de la classe	
SmilApplication f	Définie dans la classe SmilApplication.
Services offerts	
GestionTool(SmilApplication frame)	Constructeur de la classe GestionTool.
actionPerformed(ActionEvent evt)	Traite les différents événements générés par le toolbar.

4.3.6 Classe SmilApplication

La classe SmilApplication est chargée de lancer l'application et construire son l'interface graphique.

SmilApplication	
Données de la classe	
GestionMenu gmenu	Définit dans la classe GestionMenu.
GestionTool gtool	Définit dans la classe GestionTool.
Données en rapport avec le traitement de fichiers	
String nomfichier	Nom du fichier ouvert.
String nomdir	Nom du répertoire du fichier ouvert.
JFileChooser fouverture	Fenêtre pour choisir le fichier à visualiser.
boolean etatfsrc	Etat de la fenêtre qui montre le fichier source ouvert.
FenetreSource fsource	Définie dans la classe FenetreSource
Données en rapport avec l'analyseur lexicale et syntaxique	
Smil fsmil	Arbre du document SMIL(en-tête + corps).
ParseurSmil pars	Définit dans la classe PanelSmil.
Body body	Arbre du document SMIL(le corps seulement)
Données en rapport avec l'affichage du document SMIL	
Hashtable id	table de tous les identificateurs généraux.
Hashtable meta	table de tous les identificateurs des metas.
Hashtable region	table de tous les identificateurs des régions.
Hashtable type	table de tous les identificateurs des types.
Hashtable img	table de tous les identificateurs des images.
Hashtable son	table de tous les identificateurs des sons.
Hashtable txt	table de tous les identificateurs des textes.
Hashtable video	table de tous les identificateurs des videos.
Hashtable root	table de tous les éléments du root layout.
Services offerts	
SmilApplication()	Constructeur de la classe SmilApplication.
active(boolean etat)	Méthode qui active certains composantes de l'interface en fonction de l'état de l'application.
changerTaille(int width,int height)	Méthode qui change la taille de la fenêtre de l'application.
initTaille(int width,int height)	Méthode qui donne la taille initiale de la fenêtre de l'application.

ajouterPanel(Serveur serveur)	Méthode qui affiche le média d'un serveur de média
enleverPanel()	Méthode qui enléve les informations d'unserveur de média.
initialiser(boolean b)	Méthode qui initialise les donnés de l'anlyseur lexicale et syntaxique.

4.3.7 Réseau d'échanges.

Dans cette architecture, nous n'allons pas nous préoccuper du réseau d'échange qui est assuré par le système d'exploitation de la machine ou par d'autres mécanismes offerts par le langage de programmation.
De même les fonctions qui interagissent avec les différents périphériques, sont assurées par le système d'exploitation.

4.4 Codage

Dans cette section, nous allons cités brièvement les outils que nous avons utilisés pour la programmation.

A. JDK 1.1.8 (JAVA Development Kit)[16]

JDKun environnement de programmation pour le langage de JAVA.
Cet environnement comprend un compilateur, un debogeur et une machine virtuelle pour JAVA qui exécute le code produit par le compilateur.

B. JMF(JAVA Média Framework)[17]

JMF est une API(Application Programing Interface) qui s'occupe spécialement de gérer les médias temporels tels que le son et la vidéo et les intégrer dans une application JAVA. Les types de formats vidéo et audio sont regroupés dans les tableaux suivants :

Format	Type	Qualité	Utilisation du CPU	Bande passante
Cinepak	AVI (Quick time)	Moyen	Faible	Elevé
MPEG-1	MPEG	Elevé	Elevé	Elevé
H.261	AVI(RTP)	Faible	Moyen	Faible
H.263	QuickTime AVI RTP	Moyen	Moyen	Faible
JPEG	QuickTime AVI RTP	Elevé	Elevé	Elevé
Indeo	QuickTime	Moyen	Moyen	Moyen

[16] pour plus d'informations consulte le sité de SUN http://java.sun.com/products/jdk/1.1/
[17] pour plus d'informations consulte le sité de SUN http://java/sun.com/java-media/jmf/

Format	Type	Qualité	Utilisation du CPU	Bande passante
PCM	AVI (Quick time) WAV	Elevé	Faible	Elevé
Mu-Law	AVI QuickTime WAV RTP	Faible	Faible	Elevé
ADPCM	AVI	Moyen	Moyen	Moyen
DVI, IMA4	QuickTime WAV,RTP	Moyen	Moyen	Moyen
MPEG-1	MPEG	Elevé	Elevé	Elevé
G.723.1	WAV,RTP	Moyen	Moyen	Moyen
GSM	WAV,RTP	Faible	Faible	Faible
MPEG layer3	MPEG	Elevé	Elevé	Moyen

Formats audios

C. JFC(JAVA Fondation Class)[18]

JFC est un module additionnel pour le Kit de développement de JAVA, pour améliorer la programmation des interfaces d'une application. La version que nous avons utilisé, est le Swing 1.1

D. JFlex[19]

JFlex est un outil qui est utilisé pour l'analyse lexicale. Il est l'équivalant à l'outil Lex qui est couramment utilisé pour l'analyse lexicale. JFlex a été développé en JAVA. A partir d'un fichier texte qui comprend les termes à reconnaître, cet outil produit des classes JAVA qu'on intègre dans une application JAVA. Dans la suite nous nous référons au fichier smil.flex décrit dans l'annexe 2.
La structure du fichier d'analyse lexicale comprend trois sections :

- la première section comprend le code à copier dans la classe qui sera produit. Il s'agit par exemple des classes qu'il faut importer dans la classe qui sera crée.
- la seconde section comprend les directives pour l'automate et elle est comprise entre les « %% ». Il s'agit par exemple du nom de la classe qui sera crée.
- la troisième section comprend les expressions à reconnaître et les actions associées.

---

[18] pour plus d'informations consulte le site de SUN http://java.sun.com/products/jfc/
[19] pour plus d'informations consulté le site http:/ /www.jflex.de/

E. CUP[20]

CUP est un outil qui est utilisé pour l'analyse syntaxique. Il est l'équivalant à l'outil YAC. CUP a été aussi développé en JAVA. A partir du fichier qui définit le grammaire d'un langage, CUP produit d'autres classes qui analyse les informations produit par les classes de JFlex.
Dans la suite nous nous référons au fichier smil.cup décrit dans l'annexe 2.
La structure du fichier d'analyse syntaxique comprend aussi trois sections :

- la première section comprend le code à copier dans la classe qui sera produite. Il s'agit par exemple des classes qu'il faut importer dans la classe qui sera créée.
- la seconde section comprend une série des terminaux et non terminaux qui sont utilisés pour connaître les différentes productions.
- la troisième section définit une série des production du langage et les actions associées. Dans le cas de l'application que nous avons développé, nous utilisons les productions pour construire le graphe du document SMIL.

[20] pour plus d'informations consulté le site http:/ /www.princeton.edu/~appel/modern/java/CUP/

Conclusion générale.

Ce travail de fin d'études m'a permis d'atteindre les objectifs généraux suivants :
- comprendre le domaine du multimédia et en particulier l'édition des documents SMIL,
- programmer une application en JAVA,
- gérer un projet de petite taille moyenne. En effet, nous dévions organiser la rédaction et le codage en parallèle. Une fois que l'architecture globale a été figée, il ne reste qu'à coder en JAVA.
- mettre en pratique les connaissances acquises durant les deux années précédentes, surtout les cours de paradigmes de programmation.

En ce qui concerne les objectifs de ce mémoire, nous avons implémenté seulement les fonctionnalités de visualisation d'un document SMIL. C'est à dire la visualisation de petits documents SMIL.

Nous avons passé beaucoup de temps à implémenter un parseur de SMIL, ce qui m'a retardé dans l'implémentation des différentes fonctionnalités de l'application.

L'état d'avancement du logiciel nous permet :
- de visualiser un scénario séquentiel des médias image et texte,
- de restituer des médias audio et vidéo,
- de visualiser un scénario parallèle des médias image et texte.

Quant à l'évolution du logiciel, il serait intéressant de finir le codage de toutes les fonctionnalités pour nous permettre de visualiser n'importe quel document SMIL. Il serait aussi possible d'ajouter les fonctionnalités en rapport avec l'édition d'un document multimédia.

J'espère que ce mémoire vous a plu et que vous avez pris autant de plaisir que moi à l'étude du multimédia.

Bibliographie.

A. Livres.

1. J.C. Struhurmann, A.Misera et J.Paul, HTML4-XML-PUSH-FLASH,Micro Application,1 ère Edition, 1998.
2. L. Lemay et R.Cadenhead, Le Programmeur Java 1.2,Simon & Schuster Macmillan,1998.
3. A.Maret, Java Facile, Marabout, 1997.
4. Ministère de la Région Wallonne, Des Métiers pour le Multimédia,Septembre 1998.

B. Bibliographie sur sites Internet.

- http://www.w3c.org /AudioVideo/ (Langage SMIL)
- http://www.helio.org(Langage SMIL)
- http://justsmile.com(Langage SMIL)
- http://www.euroclid.fr/Cours SMIL W3C/(Langage SMIL)
- http://www3.real.com/products/tools(Langage SMIL)
- http://www.cwi.nl/GRINS(Langage SMIL)
- http://www.oratrix.com/GriNS(Langage SMIL)
- http:/ /smil.nist.gov/player/S2M2.html(Langage SMIL)
- http:/ /www.princeton.edu/~appel/modern/java/CUP/(Analyseur Syntaxique)
- http:/ /www.jflex.de/(Analysuer Lexicale)
- http://java.sun.com/products/jdk/1.1/(Java)
- http://java/sun.com/java-media/jmf/(JMf)
- http://java/sun.com/java-media/jfc/(Swing)
- http://opera.inrialpes.fr/Infos/Personnes/Nabil.Layaida (Thèse de Doctorat)

Annexe 1.

Restriction sur les balises et les attributs de SMIL

Selon le DTD de la version 1.0 de SMIL qu'on peut trouve sur le site internet http://www.w3.org/AudioVideo , nous nous sommes limités à implementer une partie de ce DTD. Les attributs ou les balises en gras ne seront pas prises en compte.

```
<!--
      Definition du DTD SMIL
 -->

<!-- Les attributs générales -->
<!ENTITY % id-attr "id ID #IMPLIED">
<!ENTITY % title-attr "title CDATA #IMPLIED">
<!ENTITY % skip-attr "skip-content (true|false) 'true'">
<!ENTITY % desc-attr "
        %title-attr;
        abstract        CDATA   #IMPLIED
        author          CDATA   #IMPLIED
        copyright       CDATA   #IMPLIED
">

<!--==================== La balise SMIL=====================================-->
<!--
     The root element SMIL contains all other elements.
-->
<!ELEMENT smil (head?,body?)>
<!ATTLIST smil
        %id-attr;
>

<!--==================== La balise Head ==================================-->
<!ENTITY % layout-section "layout|switch">

<!ENTITY % head-element "(meta*,((%layout-section;), meta*))?">

<!ELEMENT head %head-element;>
<!ATTLIST head %id-attr;>


<!--==================== La balise Layout ================================-->
<!--
     Layout contains the region and root-layout elements defined by
     smil-basic-layout or other elements defined an external layout
     mechanism.
-->
<!ELEMENT layout ANY>
<!ATTLIST layout
        %id-attr;
        type CDATA      "text/smil-basic-layout"
>


<!--==================== La balise Region ================================-->
<!ENTITY % viewport-attrs "
        height              CDATA    #IMPLIED
        width               CDATA    #IMPLIED
        background-color    CDATA    #IMPLIED
```

```
">

<!ELEMENT region EMPTY>
<!ATTLIST region
        %id-attr;
        %title-attr;
        %viewport-attrs;
        left                    CDATA     "0"
        top                     CDATA     "0"
        z-index                 CDATA     "0"
        fit                     (hidden|fill|meet|scroll|slice) "hidden"
        %skip-attr;
>

<!--==================== La balise Root-layout ========================-->
<!ELEMENT root-layout EMPTY>
<!ATTLIST root-layout
        %id-attr;
        %title-attr;
        %viewport-attrs;
        %skip-attr;
>


<!--==================== La balise Meta ================================-->
<!ELEMENT meta EMPTY>
<!ATTLIST meta
        name    NMTOKEN #REQUIRED
        content CDATA   #REQUIRED
        %skip-attr;
>

<!--==================== La balise Body ================================-->
<!ENTITY % media-object "audio|video|text|img|animation|textstream|ref">
<!ENTITY % schedule "par|seq|(%media-object;)">
<!ENTITY % inline-link "a">
<!ENTITY % assoc-link "anchor">
<!ENTITY % link "%inline-link;">
<!ENTITY % container-content "(%schedule;)|switch|(%link;)">
<!ENTITY % body-content "(%container-content;)">

<!ELEMENT body (%body-content;)*>
<!ATTLIST body %id-attr;>

<!--==================== Les attributs de Synchronisation ==============-->
<!ENTITY % sync-attributes "
        begin   CDATA   #IMPLIED
        end     CDATA   #IMPLIED
">

<!--==================== Les attributs de la balise Switch==============-->
<!ENTITY % system-attribute "
        system-bitrate              CDATA                 #IMPLIED
        system-language             CDATA                 #IMPLIED
        system-required             NMTOKEN               #IMPLIED
        system-screen-size          CDATA                 #IMPLIED
        system-screen-depth         CDATA                 #IMPLIED
        system-captions             (on|off)              #IMPLIED
        system-overdub-or-caption   (caption|overdub)     #IMPLIED
">
```

```
<!--==================== Les attributs de la balise Fill ==============-->
<!ENTITY % fill-attribute "
        fill    (remove|freeze)    'remove'
">


<!--==================== La balise Par ================================-->
<!ENTITY % par-content "%container-content;">
<!ELEMENT par    (%par-content;)*>
<!ATTLIST par
        %id-attr;
        %desc-attr;
        endsync CDATA           "last"
        dur     CDATA           #IMPLIED
        repeat  CDATA           "1"
        region  IDREF           #IMPLIED
        %sync-attributes;
        %system-attribute;
>

<!--==================== La balise Seq ================================-->
<!ENTITY % seq-content "%container-content;">
<!ELEMENT seq    (%seq-content;)*>
<!ATTLIST seq
        %id-attr;
        %desc-attr;
        dur     CDATA           #IMPLIED
        repeat  CDATA           "1"
        region  IDREF           #IMPLIED
        %sync-attributes;
        %system-attribute;
>

<!--==================== La balise Switch ================================-->
<!ENTITY % switch-content "layout|(%container-content;)">
<!ELEMENT switch (%switch-content;)*>
<!ATTLIST switch
        %id-attr;
        %title-attr;
>

<!--==================== Les objets Medias=============================-->
<!ENTITY % mo-attributes "
        %id-attr;
        %desc-attr;
        region      IDREF           #IMPLIED
        alt         CDATA           #IMPLIED
        longdesc    CDATA           #IMPLIED
        src         CDATA           #IMPLIED
        type        CDATA           #IMPLIED
        dur         CDATA           #IMPLIED
        repeat      CDATA           '1'
        %fill-attribute;
        %sync-attributes;
        %system-attribute;
">

<!ENTITY % mo-content "(%assoc-link;)*">
<!ENTITY % clip-attrs "
        clip-begin      CDATA   #IMPLIED
        clip-end        CDATA   #IMPLIED
```

```
">

<!ELEMENT ref              %mo-content;>
<!ELEMENT audio            %mo-content;>
<!ELEMENT img              %mo-content;>
<!ELEMENT video            %mo-content;>
<!ELEMENT text             %mo-content;>
<!ELEMENT textstream       %mo-content;>
<!ELEMENT animation        %mo-content;>

<!ATTLIST ref              %mo-attributes; %clip-attrs;>
<!ATTLIST audio            %mo-attributes; %clip-attrs;>
<!ATTLIST video            %mo-attributes; %clip-attrs;>
<!ATTLIST animation        %mo-attributes; %clip-attrs;>
<!ATTLIST textstream       %mo-attributes; %clip-attrs;>
<!ATTLIST text             %mo-attributes;>
<!ATTLIST img              %mo-attributes;>

<!--==================== Les Liens=====================================-->

<!ENTITY % smil-link-attributes "
        %id-attr;
        %title-attr;
        href               CDATA                    #REQUIRED
        show               (replace|new|pause)      'replace'
">


<!--==================== Inline Link Element ================================-->
<!ELEMENT a (%schedule;|switch)*>
<!ATTLIST a
        %smil-link-attributes;
>


<!--==================== Associated Link Element ============================-->
<!ELEMENT anchor EMPTY>
<!ATTLIST anchor
        %skip-attr;
        %smil-link-attributes;
        %sync-attributes;
        coords             CDATA                    #IMPLIED
>
```

Annexe 2.

Fichier d'analyseur lecical(smil.flex) et syntaxique(smil.cup)

*Smil.flex

```
/*
 ************************************************************
 * l'analyseur lexical du mini langage smil                 *
 *                                                          *
 ************************************************************
 */
package graphe.parseur;
import java_cup.runtime.*;

%%

%class Scanner
%public
%implements Lexer, sym
%pack
%full
%line
%column
%cup
%{
  StringBuffer string = new StringBuffer();
  private Symbol symbol(int type) {
    return new Symbol(type, yyline, yycolumn);
  }

  private Symbol symbol(int type, Object value) {
    return new Symbol(type, yyline, yycolumn, value);
  }

  private long parseLong(String s, int radix) {
    int  max = s.length();
    long result = 0;
    long digit;

    for (int i = 0; i < max; i++) {
      digit  = Character.digit(yy_buffer[i],radix);
      result*= radix;
      result+= digit;
    }

    return result;
  }
%}
```

```
finligne = \r|\n|\r\n
espaceblanc = {finligne} |[ \t\f]
chiffre = [0-9]
entier = {chiffre}+
lettre = [a-zA-Z]
metacar = [\.\\\/:\?\-\!\;\,\'\%\(\)\-\_]
identificateur = ({lettre}|{entier}|{metacar}|{espaceblanc})*
ident = (\"){identificateur}(\")
%%
<YYINITIAL> {
/* mots clés */
"smil"              { return symbol(SMIL); }
"id"               { return symbol(ID); }
"head"              { return symbol(HEAD); }
"meta"              { return symbol(META); }
"name"              { return symbol(NAME); }
"content"           { return symbol(CONTENT); }
"layout"            { return symbol(LAYOUT); }
"root-layout"         { return symbol(ROOTLAYOUT); }
"type"             { return symbol(TYPE); }
"width"             { return symbol(WIDTH); }
"height"            { return symbol(HEIGHT); }
"title"            { return symbol(TITLE); }
"background-color"     { return symbol(BACKGROUNDCOLOR); }
"region"            { return symbol(REGION); }
"left"             { return symbol(LEFT); }
"top"              { return symbol(TOP); }
"fit"              { return symbol(FIT); }
"body"              { return symbol(BODY); }
"par"              { return symbol(PAR); }
"seq"              { return symbol(SEQ); }
"abstract"          { return symbol(ABSTRACT); }
"author"            { return symbol(AUTHOR); }
"copyright"         { return symbol(COPYRIGHT); }
"reapet"           { return symbol(REAPET); }
"begin"            { return symbol(BEGIN); }
"end"              { return symbol(END); }
"img"              { return symbol(IMG); }
"text"             { return symbol(TEXT); }
"video"            { return symbol(VIDEO); }
"audio"            { return symbol(AUDIO); }
"src"              { return symbol(SRC); }
"alt"              { return symbol(ALT); }
"longdesc"          { return symbol(LONGDESC); }
"dur"              { return symbol(DUR); }
"clip-end"          { return symbol(CLIPEND); }
"clip-begin"        { return symbol(CLIPBEGIN); }

/*separateur*/

"="                { return symbol(EGAL); }
"<"                { return symbol(INF); }
">"                { return symbol(SUP); }
"/"                { return symbol(SLH); }
{espaceblanc}              {}
{ident}                    {string.setLength(0);
                            string.append(yytext());
                            return symbol(IDENT, string.toString());}

}
```

*smil.cup

```
/*
 ********************************
 * parseur de smil simplifie    *
 *                              *
 ********************************
 */
package graphe.parseur;
import java_cup.runtime.*;
import graphe.*;
import traitementListe.*;

/* initialisation du parseur */
parser code  {:
Lexer lexer;
public Smil filesmil;//variable qui contient la structure du document

public parser(Lexer lexer) {
      this.lexer = lexer;
     }
public void report_error(String message, Object info) {
     StringBuffer m = new StringBuffer("Error");
     if ( info instanceof java_cup.runtime.Symbol ) {
     java_cup.runtime.Symbol s = ((java_cup.runtime.Symbol)info);
     if (s.left >= 0) {
          m.append(" in line "+(s.left+1));
          if (s.right >= 0)
               m.append("column "+(s.right+1));
               }
            }
          m.append(" : "+message);
          System.out.println(m);
         }

public void report_fatal_error(String message, Object info) {
     report_error(message, info);
     throw new RuntimeException("Fatal Syntax Error");
     }


public Smil arbre_smil(){
        return parser.filesmil;
         }
:};

scan with {: return lexer.yylex(); :};

terminal SMIL;
terminal ID;
terminal HEAD;
terminal META;
terminal NAME;
terminal CONTENT;
terminal LAYOUT;
terminal ROOTLAYOUT;
terminal TYPE;
terminal WIDTH;
terminal HEIGHT;
```

```
terminal TITLE;
terminal BACKGROUNDCOLOR;
terminal REGION;
terminal LEFT;
terminal TOP;
terminal FIT;
terminal BODY;
terminal PAR;
terminal SEQ;
terminal ABSTRACT;
terminal AUTHOR;
terminal COPYRIGHT;
terminal REAPET;
terminal BEGIN;
terminal END;
terminal IMG;
terminal TEXT;
terminal VIDEO;
terminal AUDIO;
terminal SRC;
terminal ALT;
terminal LONGDESC;
terminal DUR;
terminal CLIPEND;
terminal CLIPBEGIN;

/* separateur */
terminal EGAL;
terminal INF;
terminal SUP;
terminal SLH;
terminal java.lang.String IDENT;

non terminal Longdesc nlongdesc;
non terminal Dur ndur;
non terminal Clipbegin nclipbegin;
non terminal Clipend nclipend;
non terminal Id nid;
non terminal Content ncontent;
non terminal Name nname;
non terminal Type ntype;
non terminal Width nwidth;
non terminal Height nheight;
non terminal Title ntitle;
non terminal Background nbackground;
non terminal Left nleft;
non terminal Top ntop;
non terminal Fit nfit;
non terminal Src nsrc;
non terminal Alt nalt;
non terminal Tregion tregion;
non terminal Meta nmeta;
non terminal ListeMeta suitemeta;
non terminal Region dregion;
non terminal ListeRegion suiteregion;
non terminal Abstract nabstract;
non terminal Author nauthor;
non terminal Copyright ncopyright;
non terminal Reapet nreapet;
non terminal Begin nbegin;
```

```
non terminal End nend;
non terminal Fseqpar fseqpar;
non terminal Tiatrib tiatrib;
non terminal Avatrib avatrib;
non terminal Img nimg;
non terminal Text ntext;
non terminal Video nvideo;
non terminal Son naudio;
non terminal Media media;
non terminal ListeMedia suitemedia;
non terminal Par npar;
non terminal Seq nseq;
non terminal Rootlayout drootlayout;
non terminal Layout nlayout;
non terminal Ldefaut nldefaut;
non terminal Lregion nregion;
non terminal Lrootlayout nrootlayout;
non terminal Head nhead;
non terminal Body nbody;
non terminal Nheadbody nheadbody;
non terminal Smil nsmil;

start with nsmil;

nsmil ::= INF SMIL nid:e1 SUP nheadbody:e2 INF SLH SMIL SUP
        {: Smil s = new Smil(e1,e2);
                                    RESULT = s;
                parser.filesmil = RESULT;
                    :}
    ;

nlongdesc ::= {: Longdesc lg = new Longdesc();
                        RESULT = lg;
            :}
    |LONGDESC EGAL  IDENT:e
                        {: Longdesc lg = new Longdesc(e.substring(1,(e.length()-1)));
                         RESULT = lg;
                        :}
    ;
ndur ::= {: Dur dur = new Dur();
                        RESULT = dur;
                        :}

        |DUR EGAL  IDENT:e
                        {: Dur dur = new Dur(e.substring(1,(e.length()-1)));
                         RESULT = dur;
                        :}
        ;

nclipbegin ::= {: Clipbegin big = new Clipbegin();
                        RESULT = big;
                        :}

        |CLIPBEGIN EGAL  IDENT:e
                        {: Clipbegin big = new Clipbegin(e.substring(1,
                          (e.length()-1)));
                         RESULT = big;
                        :}
        ;
```

```
nclipend ::={: Clipend end = new Clipend();
                        RESULT = end;
                       :}

        |CLIPEND EGAL  IDENT:e
                {: Clipend end = new Clipend(e.substring(1,(e.length()-1)));
                        RESULT = end;
                       :}
    ;
nid ::=  {: Id id = new Id();
                        RESULT = id;
                       :}

   |ID EGAL  IDENT:e
                {: Id id = new Id(e.substring(1,(e.length()-1)));
                 RESULT = id;
                       :}
    ;
ncontent ::= {: Content ct = new Content();
                        RESULT = ct;
                       :}
        |CONTENT EGAL  IDENT:e
                {: Content ct = new Content(e.substring(1,(e.length()-1)));
                 RESULT = ct;
                       :}
    ;
nname ::= {: Name nme = new Name();
                        RESULT = nme;
                       :}
   |NAME  EGAL  IDENT:e
                {: Name nme = new Name(e.substring(1,(e.length()-1)));
                 RESULT = nme;
                       :}
    ;
ntype ::= {: Type typ = new Type();
                        RESULT = typ;
                       :}
   |TYPE EGAL  IDENT:e
                {: Type typ = new Type(e.substring(1,(e.length()-1)));
                 RESULT = typ;
                       :}
    ;
nwidth ::= {: Width wdht = new Width();
                 RESULT = wdht;
                       :}

        |WIDTH EGAL  IDENT:e
          {: Width wdht = new Width(e.substring(1,(e.length()-1)));
                 RESULT = wdht;
                       :}
    ;
nheight ::= {: Height het = new Height();
                 RESULT = het;
                       :}
        |HEIGHT EGAL  IDENT:e
                {: Height het = new Height(e.substring(1,(e.length()-1)));
                 RESULT = het;
                       :}
    ;
```

```
ntitle ::= {: Title tit = new Title();
                        RESULT = tit;
                        :}
    |TITLE EGAL  IDENT:e
                {: Title tit = new Title(e.substring(1,(e.length()-1)));
                 RESULT = tit;
                        :}
    ;
nbackground ::= {: Background bac = new Background();
                 RESULT = bac;
                        :}

        |BACKGROUNDCOLOR EGAL  IDENT:e
                {: Background bac = new Background(e.substring(1,(e.length()-1)));
                 RESULT = bac;
                        :}
    ;
nleft ::= {: Left lef = new Left();
                 RESULT = lef;
                        :}
        |LEFT EGAL  IDENT:e
                {: Left lef = new Left(e.substring(1,(e.length()-1)));
                 RESULT = lef;
                        :}
        ;
ntop ::=  {: Top tp = new Top();
                 RESULT = tp;
                        :}
        |TOP EGAL  IDENT:e
                {: Top tp = new Top(e.substring(1,(e.length()-1)));
                 RESULT = tp;
                        :}
    ;
nfit ::={: Fit f = new Fit(); :}
        |FIT EGAL  IDENT:e
                {: Fit f = new Fit();
                if (e.equals("hidden"))
                        f.setfit(e.substring(1,(e.length()-1)),1);
                if (e.equals("fill"))
                        f.setfit(e.substring(1,(e.length()-1)),2);
                if (e.equals("meet"))
                        f.setfit(e.substring(1,(e.length()-1)),3);
                if (e.equals("scroll"))
                        f.setfit(e.substring(1,(e.length()-1)),4);
                if (e.equals("slice"))
                        f.setfit(e.substring(1,(e.length()-1)),5);
                RESULT = f;
                        :}
    ;
nsrc ::= SRC EGAL  IDENT:e
                {: Src sr = new Src(e.substring(1,(e.length()-1)));
                 RESULT = sr;
                        :}
    ;
nalt ::=    {: Alt al = new Alt();
                 RESULT = al;
                        :}
   |ALT EGAL  IDENT:e
                {: Alt al = new Alt(e.substring(1,(e.length()-1)));
                 RESULT = al;
```

```
                    :};
tregion ::= {: Tregion tr = new Tregion();
                    RESULT = tr;
                    :}

        |REGION EGAL  IDENT:e
                    {: Tregion tr = new Tregion(e.substring(1,(e.length()-1)));
                    RESULT = tr;
                    :}
    ;
nmeta ::= INF META nid:e1 nmame:e2 ncontent:e3 SLH SUP
                    {: Meta m = new Meta(e1,e2,e3);
                    RESULT = m;
                    :}

    ;
suitemeta ::= nmeta:e
                    {: ListeMeta lm = new ListeMeta();
                    lm.ListeMetaSet(1,e);
                    RESULT = lm;
                    :}
    |suitemeta:e1 nmeta:e2
                    {: int i = e1.ListeLength();
                    e1.ListeMetaSet(1,e2);
                    RESULT = e1;
                    :}
    ;
dregion ::= INF REGION nid:e1 ntitle:e2 nleft:e3 ntop:e4 nwidth:e5 nheight:e6
                    nfit:e7 SLH SUP
        {: Region r = new Region(e1,e2,e3,e4,e5,e6,e7);
                    RESULT = r;
                    :}
    ;

nabstract ::= {: Abstract a = new Abstract();
                    RESULT = a;
                    :}
    |ABSTRACT EGAL  IDENT:e
                    {: Abstract a = new Abstract(e.substring(1,(e.length()-1)));
                    RESULT = a;
                    :}

    ;
nauthor ::= {: Author au = new Author();
                    RESULT = au;
                    :}
    |AUTHOR EGAL  IDENT:e
                    {: Author au = new Author(e.substring(1,(e.length()-1)));
                    RESULT = au;
                    :}
    ;
ncopyright ::= {: Copyright co = new Copyright();
                    RESULT = co;
                    :}
        |COPYRIGHT EGAL  IDENT:e
                    {: Copyright co = new Copyright(e.substring(1,(e.length()-1)));
                    RESULT = co;
                    :}
    ;
```

```
nreapet ::= {: Reapet r = new Reapet();
                    RESULT = r;
                        :}
        |REAPET EGAL  IDENT:e
                    {: Reapet r = new Reapet(e.substring(1,(e.length()-1)));
                    RESULT = r;
                        :}      ;
nbegin ::= {: Begin b = new Begin();
                    RESULT = b;
                        :}
        |BEGIN EGAL  IDENT:e
                    {: Begin r = new Begin(e.substring(1,(e.length()-1)));
                    RESULT = r;
                        :}
    ;
nend ::= {: End en = new End();
                    RESULT = en;
                        :}
   |END EGAL  IDENT:e
                    {: End en = new End(e.substring(1,(e.length()-1)));
                    RESULT = en;
                        :}
      ;
fseqpar ::= nid:e1 nabstract:e2 nauthor:e3 ncopyright:e4 nbegin:e5 nend:e6
           ndur:e7  nreapet:e8
             {: Fseqpar sp = new Fseqpar(e1,e2,e3,e4,e5,e6,e7,e8);
                        RESULT = sp;
                        :}
                    ;
tiatrib ::= ntype:e0 nid:e1 tregion:e2 nsrc:e3 nalt:e4 ntitle:e5 nabstract:e6
       nauthor:e7 ncopyright:e8 nlongdesc:e9 nbegin:e10 nend:e11 ndur:e12
          {: Tiatrib at =
           new Tiatrib(e0,e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12);
                        RESULT = at;
                        :}
                    ;
avatrib ::= tiatrib:e1 nclipbegin:e2 nclipend:e3
                    {: Avatrib a = new Avatrib(e1,e2,e3);
                    RESULT = a;
                        :}
                    ;
nimg ::= INF IMG tiatrib:e SLH SUP
                    {: Img i = new Img("img",e);
                        RESULT = i;
                         :}
    ;
ntext ::= INF TEXT tiatrib:e SLH SUP
                    {: Text t = new Text("txt",e);
                        RESULT = t;
                         :}
    ;
nvideo ::= INF VIDEO avatrib:e SLH SUP
                    {: Video v = new Video("video",e);
                        RESULT = v;
                         :}
    ;
naudio ::= INF AUDIO avatrib:e SLH SUP
                    {: Son s = new Son("son",e);
                        RESULT = s;
                         :}
```

```
            ;
media ::= naudio:e {: RESULT = (Media)e; :}
      |nimg:e {:RESULT = (Media)e; :}
      |npar:e {: RESULT = (Media)e; :}
      |nseq:e {: RESULT = (Media)e; :}
      |ntext:e {: RESULT = (Media)e; :}
      |nvideo:e {: RESULT = (Media)e; :}
      ;
suitemedia ::= media:e
                  {: ListeMedia la  =new ListeMedia();
                        la.ListeMediaSet(1,e);
                        RESULT = la; :}
    |suitemedia:e1 media:e2
                  {: int i = e1.ListeLength();
                  e1.ListeMediaSet(1,e2);
                   RESULT = e1;
                        :}
      ;
npar ::= INF PAR fseqpar:e1 SUP suitemedia:e2 INF SLH PAR SUP
                  {: Par p = new Par("par",e1,e2);
                        RESULT = p;
                         :}
      ;
nseq ::= INF SEQ fseqpar:e1 SUP suitemedia:e2 INF SLH SEQ SUP
                  {: Seq s = new Seq("seq",e1,e2);
                        RESULT = s;
                         :}
                  ;
suiteregion ::=
                  dregion:e
                  {: ListeRegion lr = new ListeRegion();
                   lr.ListeRegionSet(1,e);
                  RESULT = lr;
                  :}
          |suiteregion:e1 dregion:e2
                  {: int i = e1.ListeLength();
                    e1.ListeRegionSet(1,e2);
                    RESULT = e1;
                   :}

      ;

drootlayout ::= INF ROOTLAYOUT nid:e1 ntitle:e2 nwidth:e3 nheight:e4
            nbackground:e5 SLH SUP
          {: Rootlayout rt = new Rootlayout(e1,e2,e3,e4,e5);
                        RESULT = rt;
                        :}
                  ;

nregion ::= INF LAYOUT ntype:e1 SUP dregion:e2 INF SLH LAYOUT SUP
                  {: Lregion lr =new Lregion(e1,e2);
                        RESULT = lr;
                   :}
      ;
nrootlayout ::= INF LAYOUT ntype:e1 SUP drootlayout:e2 suiteregion:e3
                  INF SLH LAYOUT SUP
                  {: Lrootlayout r =new Lrootlayout(e1,e2,e3);
                        RESULT = r;
                   :}
      ;
```

```
nldefaut ::= INF LAYOUT ntype:e SUP INF SLH LAYOUT SUP
        {: Ldefaut l = new Ldefaut(e);
        :}
         ;
nlayout ::= nldefaut:e
      {: Layout ln = new Layout(e,0);
           RESULT = ln;
           :}
           |nregion:e
          {: Layout ln = new Layout(e,1);
          RESULT = ln;
           :}
     |nrootlayout:e
           {: Layout lr = new Layout(e,2);
               RESULT = lr;
               :}
     ;

nhead ::=INF HEAD nid:e1 SUP nlayout:e2 INF SLH HEAD SUP
         {: Head h = new Head(e1,(Layout)e2);
          RESULT = h;
          :}

         |INF HEAD nid:e1 SUP suitemeta:e2 INF SLH HEAD SUP
         {: Head h = new Head(e1,(ListeMeta)e2);
          RESULT = h;
          :}

         |INF HEAD nid:e1 SUP suitemeta:e2 nlayout:e3 INF SLH HEAD SUP
                 {: Head h = new Head(e1,e2,e3);
          RESULT = h;
          :}
        ;

nbody ::= INF BODY  nid:e1 SUP suitemedia:e2  INF SLH BODY SUP
                 {: Body b = new Body(e1,e2);
          RESULT = b;
         :}
     ;
nheadbody ::= nhead:e1 nbody:e2
         {: Nheadbody hd = new Nheadbody(e1,e2);
                 RESULT = hd;
                 :}
         |nbody:e
         {: Nheadbody hd = new Nheadbody(e);
                 RESULT = hd;
                 :}

         ;
```

Annexe 3 : Interface de l'application

```
/*
 ************************************************************************
 *Cette classe definit le module coordinateur et client, il est chargé  *
 *de lancer l'application et géree les autres modules de l'application   *
 *Auteur:Kayitana Alain                                                  *
 *Institut d'informatique à Namur                                        *
 ************************************************************************
 */


import java.awt.*;
import java.awt.Image;
import java.awt.event.*;
import java.lang.String;
import java.io.*;
import java.net.*;
import java.util.*;
import javax.media.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.BorderFactory;
import javax.swing.filechooser.*;

import traitementFichier.*;
import graphe.*;
import graphe.parseur.*;
import coordinateur.*;
import traitementListe.*;
import serveur.*;
import serveur.serveurImg.*;
import serveur.serveurText.*;
import serveur.serveurVideo.*;

public class SmilApplication extends JFrame {


public GestionMenu gmenu;//class qui gere les evenements du menubar
public GestionTool gtool;//class qui gere les evenements du toolbar

/* variables en rapport avec le traitement de fichiers */

public String nomfichier = null;//nom du fichier ouvert
public String nomdir = null;//nom du repertaoire du fichier
public JFileChooser fouverture;//fenetre pour choisir le fichier à visualiser
public boolean etatfsrc =false;//variable d'etat qui montre le fichier source
ouvert ou ferme
public FenetreSource fsource;//fenetre qui montre le fichier source

/* variables en rapport avec l'analyseur lexicale et syntaxique */

public Smil fsmil = null; //arbre du fichier smil(head + body)
public ParseurSmil pars = null;//class qui definie le parseur
public Body body=null; //l'arbre du fichier(body seulement)

/*variables en rapport avec l'affichage du document smil */

public Hashtable id;// tables de tous les identificateurs generales id
public Hashtable meta;// tables de tous les identificateurs des metas
public Hashtable region; // tables de tous les identificateurs des regions
```

```
public Hashtable type; // tables de tous les identificateurs des types
public Hashtable img; // tables de tous les identificateurs des images
public Hashtable son; // tables de tous les identificateurs des sons
public Hashtable txt; // tables de tous les identificateurs des texts
public Hashtable video; // tables de tous les identificateurs des videos
public Hashtable root;// tables de tous les elements du root layout


/* variables principales */

public PanelSmil panelsmil;//voir package temp
public JMenuBar menubar;
public JMenu fichier;
public JMenuItem ouvrir;
public JMenuItem fermer;
public JMenuItem quitter;
public JMenu affichage;
public JMenuItem fsrc;
public JMenu aide;
public JMenuItem apropos;
public JToolBar toolbar;
public JButton boutonplay;
public JButton boutonstop;
public JButton boutonreplay;
public Insets insets;
public
        super("Visualisation de Fichier SMIL");

       /*
         * construction du menubar. ce menubar a comme menu:
         * Fichier(Itemmenu:Ouvrir un fichier smil,Fermer un fichier smil,
         * et Quitter l'application)
         *
         * menu Affichage(itemmenu: Fichier source)
         * menu Aide(Itemmenu:A propos de..)
         */

        gmenu = new GestionMenu(this);
        menubar = new JMenuBar();

        /*le menu Fichier*/

        fichier = new JMenu("Fichier");
        ouvrir = new JMenuItem("Ouvrir un fichier SMIL");
        fermer = new JMenuItem("Fermer un fichier SMIL");
        quitter = new JMenuItem("Quitter");
        ouvrir.addActionListener(gmenu);
        fermer.addActionListener(gmenu);
        quitter.addActionListener(gmenu);
        fichier.add(ouvrir);
        fichier.add(fermer);
        fichier.addSeparator();
        fichier.add(quitter);

        /* le menu Affichage*/

        affichage = new JMenu("Affichage");
        fsrc = new JMenuItem("Fichier source");
        fsrc.addActionListener(gmenu);
        affichage.add(fsrc);
```

```
        /*le menu Aide */
        aide = new JMenu("Aide");
        apropos = new JMenuItem("A propos de ..");
        apropos.addActionListener(gmenu);
        aide.add(apropos);

        /*ajout des menus dans l'ordre*/
        menubar.add(fichier);
        menubar.add(affichage);
        menubar.add(aide);

        /* creation du toolbar pour des boutons de visualisation
         * bouton play, bouton stop, bouton pause
         */
         gtool = new GestionTool(this);
         toolbar = new JToolBar();
         toolbar.setFloatable(false);
         boutonplay = new JButton("Play");
         boutonplay.setToolTipText("Ce bouton joue le fichier SMIL");
         boutonstop = new JButton("Stop");
         boutonstop.setToolTipText("Ce bouton arrete de jouer le fichier SMIL");
         boutonplay.addActionListener(gtool);
         boutonstop.addActionListener(gtool);
         toolbar.add(boutonplay);
         toolbar.addSeparator();
         toolbar.add(boutonstop);


        /* initialisation de la fenetre d'ouverture du fichier smil*/

        fouverture = new JFileChooser();
        fouverture.addChoosableFileFilter(new SmilFilter());

        /*
         *le code qui suit sert à positionner la fenetre principal à l'ecran
         * on utilise la resolution de l'ecran pour la positionner
         */

    /* ajout des differents panel ou contenaire principal*/

        active(false);
        setJMenuBar(menubar);
        JPanel tool = new JPanel();
        tool.add(toolbar);
            Border etched = BorderFactory.createEtchedBorder();
            tool.setBorder(etched);

     getContentPane().add(tool,BorderLayout.NORTH);



    /* taile de la fenetre initial*/

    insets = getInsets();
    setSize(insets.left + insets.right + 350,
            insets.top + insets.bottom + 300);


}
```

```
  /*
    Cette methode active ou desactive certains composants de
    l'interface selon qu'un fichier smil est ouvert ou pas
    */

   public void
        boutonplay.setEnabled(etat);
        boutonstop.setEnabled(etat);
        affichage.setEnabled(etat);
        fermer.setEnabled(etat);
        }

  /*
  Cette methode change la taille du panelsmil ou est affiche
  la présentation du document smil
  */

  public void changerTaille(int width,int height){
            if (panelsmil != null)
                  {panelsmil.setSize(width,height);}
            setSize(new Dimension(width+insets.left+insets.right,
            (height*7)/5));
            setVisible(true);

        }

/* dimensionner la taille initial de
              la fenetre smil */


public void initTaille(int width,int height){
        if (panelsmil != null)
            {enleverPanel();}
        setSize(new Dimension(300,250));

   }

/*cette fonction ajouter le panel de visualisation du document smil*/

public void ajouterPanel(Serveur serveur){

      String typeserv = serveur.m.typeMedia();
      if (typeserv.equals("txt"))
      {
      panelsmil = new PanelSmil((Stext)serveur,this);
      getContentPane().add(panelsmil, BorderLayout.CENTER);
System.out.println("j'ai ajouter le panel text");
      setVisible(true);
      }else
      if(typeserv.equals("img"))
      {
      panelsmil = new PanelSmil((Simg)serveur,this);
      getContentPane().add(panelsmil, BorderLayout.CENTER);
System.out.println("j'ai ajouter le panel image");
      setVisible(true);
      }else
        if(typeserv.equals("video"))
      {
        panelsmil = new PanelSmil((Svideo)serveur,this);
      getContentPane().add(panelsmil, BorderLayout.CENTER);
System.out.println("j'ai ajouter le panel video");
```

```
        setVisible(true);
        }

}


/*cette fonction enlever le panel de visualisation du document smil*/

public void enleverPanel(){
        getContentPane().remove(panelsmil);
        setSize(getSize().width-(insets.left+insets.right+5),
                getSize().height-(insets.top+insets.bottom+5));

        }


/* cette methode sert à initialiser les variables pour le parseur */

public void initialiser(boolean b){
        if (b==true){
          pars = null;
          fsmil = null;
          body = null;
          id = new Hashtable();
          meta = new Hashtable();
          region = new Hashtable();
          type = new Hashtable();
          img = new Hashtable();
          son = new Hashtable();
          txt = new Hashtable();
          video = new Hashtable();
          root = new Hashtable();
         }else{
              pars = null;
              fsmil = null;
              body = null;
              id = null;
              meta = null;
              region = null;
              type = null;
              img = null;
              son = null;
              txt = null;
              video = null;
              root =null;
        }

}

public static void main(String[] args){
 SmilApplication frame = new SmilApplication();
//gestion du look Metal de swing pour l'interface de l'application
         try{

UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
             }catch(Exception e){
                 System.err.println("impossible d'utiliser l'apparence"
                 +"système d'exploitation en cours: "+ e);
             }
```

```
        WindowListener l = new WindowAdapter(){
                public void windowClosing(WindowEvent e){
                        System.exit(0);
                        }
                  };

        frame.addWindowListener(l);
        frame.setVisible(true);


        }

}
```

Annexe 4 :Package coordinateur

```
/*
 * cette classe est chargée de gerer les évenements produits  *
 * par le menu bar                                            *
 */
package coordinateur;
import java.awt.*;
import java.awt.event.*;
import java.lang.String;
import java.io.*;
import java.util.*;
import javax.media.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.BorderFactory;
import javax.swing.filechooser.*;
import traitementListe.*;
import traitementFichier.*;
import graphe.*;
import graphe.parseur.*;
import SmilApplication;
import coordinateur.PanelSmil;
import graphe.parseur.ParseurSmil;

public class GestionMenu implements ActionListener{

public SmilApplication frame;//frame qui a le menu

public GestionMenu(SmilApplication parent){
        frame=parent;
        }

public void actionPerformed(ActionEvent evt){
        JMenuItem src = (JMenuItem)(evt.getSource());
         String text =src.getText();
        if (text.equals("Ouvrir un fichier SMIL"))
            if (frame.nomfichier != null)
                {
                String question = "Le fichier" +
                        frame.nomfichier + "est ouvert" + "\n" +
                        "Voulez vous fermer ce fichier ?";
                String tquestion = "Choix de fichier";

                int n = JOptionPane.showConfirmDialog(null,
                        question,
                        tquestion,
                        JOptionPane.YES_NO_OPTION);
                if (n == JOptionPane.YES_OPTION)
                        {
                        if (frame.etatfsrc == true)
                          {frame.fsource.setVisible(false);}
                        frame.initTaille(300,250);
                        int valouverture =
frame.fouverture.showOpenDialog(frame);
                        if (valouverture ==JFileChooser.APPROVE_OPTION)
                                {
                                File file =frame.fouverture.getSelectedFile();
                                frame.initialiser(true);
                                frame.nomfichier = file.getPath();
```

```
                                frame.nomdir = file.getParent();
                                //on passe le fichier au parseur smil

                                frame.pars = new
ParseurSmil(frame.nomfichier,frame);
                                //si pas de probleme, l'arbre du fichier est
                                //enregistre dans fsmil
                                frame.fsmil = frame.pars.p.filesmil;
                                frame.pars.TraiteSmil(frame.fsmil);
                                frame.setTitle(frame.nomfichier);
                                }
                        }
                } else
                {
                 int valouverture = frame.fouverture.showOpenDialog(frame);
                 if (valouverture ==JFileChooser.APPROVE_OPTION)
                      {
                       File file =frame.fouverture.getSelectedFile();
                       frame.initialiser(true);
                       frame.nomfichier = file.getPath();
                       frame.nomdir = file.getParent();
                       frame.pars = new ParseurSmil(frame.nomfichier,frame);
                       frame.fsmil = frame.pars.p.filesmil;
                       /*
                       affichage du document smil
                       on traite l'entête de ce document
                       le corps est fait par le bouton play
                       */
                       frame.pars.TraiteSmil(frame.fsmil);
                       frame.setTitle(frame.nomfichier);
                       frame.active(true);
                       }
                }


        if (text.equals("Fermer un fichier SMIL"))
            {
                frame.setTitle("Visualisation de Fichier SMIL");
                frame.initialiser(false);
                frame.nomfichier = null;
                frame.nomdir = null;
                frame.body = null;
                frame.active(false);
                frame.setVisible(true);

                   frame.enleverPanel();
                   frame.initTaille(300,250);
                if (frame.etatfsrc == true)
                        {
                        frame.fsource.setVisible(false);
                         }

            }

        if (text.equals("Quitter"))
            {
                System.exit(0);

            }
```

```
        if (text.equals("Fichier source"))
            {
                frame.fsource = new
FenetreSource(frame,frame.nomfichier,false);
                frame.fsource.setVisible(true);
                frame.etatfsrc = true;

            }

        if (text.equals("A propos de .."))
            {
                String message="Ce petit logiciel sert pour"+"\n"+
                        "visualiser un fichier SMIL "+"\n"+
                       "Fait par Kayitana Alain"+"\n"+
                        "pour l'obtention du diplome"+"\n"+
                        "de Licence en Informatique.";
                JOptionPane.showMessageDialog(null,message);

            }
    }

}


/*
 ****************************************************************
 * cette classe est chargée de gerer les évenements produits    *
 * par le tool bar                                              *
 ****************************************************************
 */
package coordinateur;
import java.awt.*;
import java.awt.event.*;
import java.lang.String;
import java.io.*;
import java.util.*;
import javax.media.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.BorderFactory;
import javax.swing.filechooser.*;
import traitementFichier.*;
import traitementListe.*;
import graphe.*;
import graphe.parseur.*;
import SmilApplication;
import coordinateur.PanelSmil;
import serveur.*;
import serveur.serveurImg.*;
import serveur.serveurText.*;
import serveur.serveurVideo.*;

public class GestionTool implements ActionListener{

public SmilApplication frame;//frame qui a le menu

public GestionTool(SmilApplication parent){
        frame=parent;
        }
```

```
public void actionPerformed(ActionEvent evt){
        JButton src = (JButton)(evt.getSource());
         String text =src.getText();
        if (text.equals("Play"))
            if (frame.body != null)
                {
                 frame.pars.TraiteBody(frame.body);
                }


        if (text.equals("Stop"))
            {
            frame.enleverPanel();
            //il faut arrete tous les threads
             }
    }


}

package coordinateur;

import java.awt.*;
import java.awt.event.*;
import java.lang.String;
import java.util.*;
import javax.media.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.BorderFactory;
import javax.swing.filechooser.*;

import SmilApplication;
import graphe.*;
import traitementListe.*;
import serveur.*;
import serveur.serveurText.*;
import serveur.serveurImg.*;
import serveur.serveurVideo.*;

public class PanelSmil extends JPanel implements ActionListener,
ControllerListener{

public SmilApplication f;

Timer timer;
public Simg simg = null;//cas d'une image seulement
public Stext stext = null;//cas d'un text seulement
public Svideo svideo = null;//cas d'un video
public Sson sson = null;//cas d'un video
public int choix;
boolean visible = false;
Component comp;
/*choix est: 1 pour un text, 2 pour un iamge,3 pour un son,
      4 pour un video, 5 pour un seq, 6 pour un par
*/

public PanelSmil(Simg sim,SmilApplication frame){
      f = frame;
      simg = sim;
```

```
        MediaTracker m_image = new MediaTracker(this);
          m_image.addImage(simg.image,1);
          try {
          m_image.waitForID(1);
          }catch(InterruptedException e){}
          if( m_image.checkID(1,true))
            { choix = 2;
              setLayout(null);
              Border etched = BorderFactory.createEtchedBorder();
              setBorder(etched);
              timer = new Timer(0,this);
              timer.setRepeats(false);
              timer.setCoalesce(true);
              timer.setInitialDelay(3000);

            }

    }

public PanelSmil(Stext stx,SmilApplication frame){
        f = frame;
        stext = stx;
        choix = 1;
        Border etched = BorderFactory.createEtchedBorder();
        setBorder(etched);
        timer = new Timer(0,this);
        timer.setRepeats(false);
        timer.setCoalesce(true);
        timer.setInitialDelay(3000);
        }
public PanelSmil(Svideo v,SmilApplication frame){
        f = frame;
        svideo = v;
        choix = 4;
        Border etched = BorderFactory.createEtchedBorder();
        setBorder(etched);
        svideo.p.addControllerListener(this);
        svideo.commencer();
        /*timer = new Timer(0,this);
        timer.setRepeats(false);
        timer.setCoalesce(true);
        timer.setInitialDelay(3000);*/
        }

public PanelSmil(Sson v,SmilApplication frame){
        f = frame;
        sson = v;
        choix = 3;
        Border etched = BorderFactory.createEtchedBorder();
        setBorder(etched);
        sson.p.addControllerListener(this);
        sson.commencer();
        /*timer = new Timer(0,this);
        timer.setRepeats(false);
        timer.setCoalesce(true);
        timer.setInitialDelay(3000);*/
        }


public void actionPerformed(ActionEvent evt){
```

```
System.out.println("test de "+ timer.getDelay());
                    timer.stop();
                    if (svideo != null)
                          svideo.detruire();
                    if (sson != null)
                          sson.detruire();


                    f.enleverPanel();
      }

public synchronized void controllerUpdate(ControllerEvent event){
       if(event instanceof RealizeCompleteEvent){

                  if((comp =svideo.p.getVisualComponent())!=null)
                    {
                    visible = true;
                    timer = new Timer(svideo.convertirmseconde(),this);
                    /*add(comp);
                    comp.setBounds(svideo.left,svideo.top,
                          svideo.width,svideo.height);
                    //svideo.p.setMediaTime(new Time(0));
                    //svideo.commencer();*/
                    }
                  /*if (event instanceof PrefetchCompleteEvent){
                    Component comp;
                  if((comp =svideo.p.getVisualComponent())!=null)
                    {
                    add(comp);
                    comp.setBounds(svideo.left,svideo.top,
                          svideo.width,svideo.height);

                    timer = new Timer(svideo.convertirmseconde(),this);
                    timer.setRepeats(false);
                    timer.setCoalesce(true);
                    timer.setInitialDelay(1000);
                    timer.start();
// faire aussi pour le son
              }*/
              }
       }


/*
 ************************************************************
 * cette fonction appelle le module d'affichage de ce       *
 * media                                                    *
 ************************************************************
 */

public void paintComponent(Graphics g) {
      super.paintComponent(g); //paint background
      switch(choix){
      case 1 :
            g.drawString(stext.text[0],stext.left,stext.top);
            g.drawString(stext.text[1],stext.left,stext.top+12);
            g.drawString(stext.text[2],stext.left,stext.top+24);
            timer.setDelay(stext.convertirmseconde());
            timer.start();
            break;
```

```
        case 2 :
              g.drawImage(simg.image,simg.left,simg.top, this); //85x62 image
              timer.setDelay(simg.convertirmseconde());
              timer.start();
              break;
        case 4 :
              while (visible == false){}
              add(comp);
              comp.setBounds(svideo.left,svideo.top,
                          svideo.width,svideo.height);
              timer.setRepeats(false);
              timer.setCoalesce(true);
              timer.setInitialDelay(1000);
              timer.start();
              break;
        case 3 :
              //while (visible == false){}
              //add(comp);
              //comp.setBounds(svideo.left,svideo.top,
                          svideo.width,svideo.height);
              timer.setRepeats(false);
              timer.setCoalesce(true);
              timer.setInitialDelay(1000);
              timer.start();
              break;

        }
  }


}
```

Annexe 5 : package graphe

```
/* non-terminal ABSTRACT*/
package graphe;
public class Abstract{
public String a_name;
      public Abstract(String ident){
        a_name=ident;
        }
      public Abstract(){
        a_name=null;
        }

}

/*non-terminal ALT /
package graphe;
public class Alt{
public String a_name;
      public Alt(String ident){
        a_name=ident;
        }
      public Alt(){
        a_name=null;
        }
}

/* non-terminal AUTHOR */
package graphe;
public class Author{
public String a_name;
      public Author(String ident){
        a_name=ident;
        }
      public Author(){
        a_name=null;
        }
}


/*
 ************************************************
 * non-terminal AVATRIB                         *
 *contient les attributs de l'audio ou la video *
 ************************************************
 */
package graphe;
public class Avatrib{
public Clipbegin a_begin;
public Clipend a_end;
public Tiatrib a_atrib;
      public Avatrib(Tiatrib atrib,
                Clipbegin begin,
                Clipend end){
                a_atrib = atrib;
                a_begin = begin;
                a_end = end;
               }
}
```

```
/*
 ****************************
 * non-terminal BACKGROUND  *
 ****************************
 */
package graphe;
public class Background{
public String b_id;
      public Background(String ident){
        b_id=ident;
        }
      public Background(){
        b_id=null;
        }
}

/* non-terminal BEGIN*/
package graphe;
public class Begin{
public String b_name;
      public Begin(String ident){
        b_name=ident;
        }
      public Begin(){
        b_name=null;
        }
}

/* BODY */
 package graphe;
 import traitementListe.*;
 public class Body{
 public  ListeMedia b_body;
 public  Id b_id;
       public Body(Id id,ListeMedia liste){
        b_id =id;
        b_body = liste;
       }
  }


/* non-terminal CLIPBEGIN  */
package graphe;
public class Clipbegin{
String c_name;
      public Clipbegin(String ident){
        c_name=ident;
        }
      public Clipbegin(){
        c_name=null;
        }
}
```

```
/* non-terminal CLIPEND */
package graphe;
public class Clipend{
String c_name;
      public Clipend(String ident){
        c_name=ident;
        }
      public Clipend(){
        c_name=null;
        }
}

/* non-terminal CONTENT        *
 * c'est une liste de string  *
 ******************************
 */
package graphe;
public class Content{
public String c_ident;
      public Content(String sident){
        c_ident = sident;
        }
      public Content(){
        c_ident = null;
        }
}

/*
 ***************************
 * non-terminal COPYRIGHT  *
 ***************************
 */
package graphe;
public class Copyright{
public String a_name;
      public Copyright(String ident){
        a_name=ident;
        }
      public Copyright(){
        a_name=null;
        }
}

/*non-terminal DUR */
package graphe;
public class Dur{
public String d_name;
      public Dur(String ident){
        d_name=ident;
        }
      public Dur(){
        d_name=null;
        }
}
```

```
/*non-terminal END */
package graphe;
public class End{
public String e_name;
      public End(String ident){
        e_name=ident;
        }
      public End(){
        e_name=null;
        }
}

/* non-terminal Fit */
package graphe;
public class Fit{
public String f_id;
int f_choix;
//1 hidden, 2 fill, 3 meet, 4 scroll, 5 slice
// f_id peut être hidden, fill, meet, scroll, slice
      public Fit(){
        f_id=null;
        }
      public void setfit(String s,int choix){
        f_id = s;
        f_choix =choix;
        }
}

/*
 ************************************************
 * non-terminal FSEQPAR                         *
 *contient les attributs de par et seq          *
 ************************************************
 */
package graphe;
public class Fseqpar{
Id f_id;
Abstract f_ab;
Author f_auth;
Copyright f_cop;
Begin f_big;
End f_end;
Dur f_dur;
Reapet f_rep;
public Fseqpar(Id id,
               Abstract ab,
               Author auth,
               Copyright cop,
               Begin big,
               End end,
               Dur dur,
               Reapet rep
               ){
        f_id = id;
        f_ab = ab;
        f_auth = auth;
        f_cop = cop;
        f_big = big;
        f_end = end;
        f_dur = dur;
```

```
        f_rep = rep;

              }
}

/* non-terminal HEAD*/
package graphe;
import traitementListe.ListeMeta;
public class Head{
public Id h_id;
public ListeMeta h_meta =null;
public Layout h_lay = null;
      public Head(Id id,ListeMeta smeta,Layout lay){
        h_id = id;
        h_meta = smeta;
        h_lay = lay;
        }
      public Head(Id id ,Layout lay){
        h_id = id;
        h_lay = lay;
        }
      public Head(Id id,ListeMeta smeta){
        h_id = id;
        h_meta = smeta;
        }
}

/*
 * non-terminal HEIGHT  *
 */
package graphe;
public class Height{
public String h_id;
      public Height(String ident){
        h_id=ident;
        }
      public Height(){
        h_id= null;
        }
}

/*
 * non-terminal ID
 */
package graphe;
public class Id{
public String i_id;
      public Id(String ident){
        i_id=ident;
        }
      public Id(){
        i_id=null;
        }
}
```

```
/*media image */
 package graphe;
 public class Img extends Media{
 public Tiatrib i_atrib;
      public Img(String image,Tiatrib atrib){
        if (image.equals("img")){
        this.typemedia = "img";
        i_atrib = atrib;
        }
     }
  }
/*
 * non-terminal NLAYOUT      *
 */
package graphe;
public class Layout{
//o pour rien ,1 pour Lregion, 2 pour Lrootlayout
public int l_choix;
public Lregion l_reg ;
public Lrootlayout l_root;
public Ldefaut l_def;
      public Layout(Lregion lreg, int choix){
        if (choix == 1){
                l_choix = 1;
                l_reg = lreg;
                }
           }
      public Layout(Lrootlayout root, int choix){
        if (choix == 2){
        l_choix = 2;
        l_root = root;
        }
     }
      public Layout(Ldefaut def,int choix){
        if (choix == 0){
        l_choix = 0;
        l_def = def;
                }
     }
}

/*
 * non-terminal NLDEFAUT *
 */
package graphe;
public class Ldefaut{
public Type l_type;
      public Ldefaut(Type type){
        l_type = type;
         }
}

/*
 * non-terminal LEFT    *
 */
package graphe;
public class Left{
public String l_id;
      public Left(String ident){
        l_id=ident;
```

```
        }
      public Left(){
        l_id=null;
        }
}

/*
 * non-terminal LONGDESC*
 */
package graphe;
public class Longdesc{
public String d_name;
      public Longdesc(String ident){
        d_name=ident;
        }
      public Longdesc(){
        d_name=null;
        }
}

/*
 * non-terminal LREGION *
 */
package graphe;
public class Lregion{
public Type l_type;
public Region l_reg;
      public Lregion(Type type,Region ident){
        l_type = type;
        l_reg = ident;
        }
}

/*
 * non-terminal Lrootlayout *
 */
package graphe;
import traitementListe.*;
public class Lrootlayout{
public Type l_type;
public Rootlayout l_root;
public ListeRegion l_reg;
      public Lrootlayout(Type type,Rootlayout root,ListeRegion l){
        l_type = type;
        l_root = root;
        l_reg = l;
          }
}

/*
 ************************************************************************
 * classe génerique de media avec un seul méthode qui teste si c'est*
 * une image , un text, un audio ou un video ou seq ou par          *
 *le cas seq singnifie une sequence de media et                     *
 *le cas par singnifie des media presente ensemble                  *
 ************************************************************************
 */
package graphe;
public class Media{
public String typemedia=null;
```

```
// = media image , txt = media text
// = media audio , videp = media video
// seq = sequence de medias
// par = medias  synchronis,s
     public String typeMedia(){
        String type=null;

       if ((this.typemedia).equals("img"))
              {
              type ="img";
              }
       if ((this.typemedia).equals("txt"))
              {
              type ="txt";
              }
       if ((this.typemedia).equals("son"))
              {
              type ="son";
              }

       if ((this.typemedia).equals("video"))
              {
              type ="video";
              }

       if ((this.typemedia).equals("seq"))
              {
              type ="seq";
              }

       if ((this.typemedia).equals("par"))
              {
              type ="par";
              }
      return type;

     }
 }

/*
 ********************************************
 * non-terminal META                        *
 * c'est une ID suivie de NAME puis CONTENT  *
 ********************************************
 */
package graphe;
public class Meta{
public Id m_id;
public Name m_name;
public Content m_content;
      public Meta(Id id,Name name,Content content){
       m_id = id;
       m_name = name;
       m_content =content;
       }
}
```

```
/*
 * non-terminal NAME     *
 */
package graphe;
public class Name{
public String n_name;
      public Name(String ident){
        n_name=ident;
        }
      public Name(){
        n_name=null;
        }
}

/*
 * NHEADBODY*
 */
package graphe;
public class Nheadbody{
public  Head n_hd=null;
public  Body n_bd=null;
      public Nheadbody(Head hd,Body bd){
        n_hd = hd;
        n_bd = bd;
       }
       public Nheadbody(Body bd){
        n_hd = null;
        n_bd = bd;
       }
  }


/*
 *media par(liste de medias … jouer en parallele)        *
 */
 package graphe;
 import traitementListe.ListeMedia;
 public class Par extends Media{
 public  ListeMedia p_par;
 public Fseqpar p_s;
 // la liste contient l'ordre de la sequence
 // des medias à jouer
 // il faut convertir les differents medias
 // en Objects
 // ex le premier elements de la liste est le premier
 // être jouer
       public Par(String par,Fseqpar s,ListeMedia liste){
        if (par.equals("par")){
        this.typemedia = "par";
        p_s =s;
        p_par = liste;
        }
    }
  }
```

```
/*
 * non-terminal REAPET  *
 */
package graphe;
public class Reapet{
public String r_name;//zero par defaut
      public Reapet(String ident){
        r_name=ident;
        }
      public Reapet(){
        r_name=null;
        }
}

/*
 * non-terminal Region  *
 */
package graphe;
public class Region{
public Id r_id;
public Left r_left;
public Top r_top;
public Width r_width;
public Height r_height;
public Title r_title;
public Fit r_fit;
      public Region(Id id,Title title,Left left,
              Top top,Width width,
              Height height,
              Fit fit ){
        r_id = id;
        r_left = left;
        r_top = top;
        r_width = width;
        r_height = height;
        r_title = title;
        r_fit = fit;
        }
}

/*
 * non-terminal ROOTLAYOUT  *
 */
package graphe;
public class Rootlayout{
public Id r_id;
public Width r_width;
public Height r_height;
public Title r_title;
public Background r_back;
      public Rootlayout(Id id,Title title,Width width,
              Height height,
              Background back){
        r_id = id;
        r_width = width;
        r_height = height;
        r_title = title;
        r_back = back;

        }}
```

```
/*
 *media sequence(liste de media … jouer en sequence)    *
 */
 package graphe;
 import traitementListe.ListeMedia;
 public class Seq extends Media{
 public  ListeMedia s_seq;
 public Fseqpar sp;
 // la liste contient l'ordre de la sequence
 // des medias … jouer
 // il faut convertir les differents medias
 // en Objects
 // ex le premier element de la liste est le premier
 // être jouer
       public Seq(String seq,Fseqpar s,ListeMedia liste){
        if (seq.equals("seq")){
        this.typemedia = "seq";
        sp = s;
        s_seq = liste;
        }
     }
  }


/*
 * SMIL                                                  *
 */
 package graphe;
 public class Smil{
 public Id s_id;
 public  Nheadbody s_hb;
       public Smil(Id id,Nheadbody hb){
        s_id =id;
        s_hb = hb;
       }
  }


/*media video*/
 package graphe;
 public class Son extends Media{
 public  Avatrib s_atrib;
       public Son(String son,Avatrib atrib){
        if (son.equals("son")){
        this.typemedia = "son";
        s_atrib = atrib;
        }
      }
  }


/*
 * non-terminal SRC     *
 */
package graphe;
public class Src{
public String s_name;
      public Src(String ident){
        s_name=ident;
```

```
        }
}

/*
 *media Text
 */
 package graphe;
 public class Text extends Media{
 public Tiatrib t_atrib;
      public Text(String text,Tiatrib atrib){
        if (text.equals("txt")){
        this.typemedia = "txt";
        t_atrib = atrib;
        }
      }
  }

/*
 * non-terminal TIATRIB                              *
 *contient les attributs de l'image ou le taxte *
 */
package graphe;
public class Tiatrib{
public Type t_tp;
public Id t_id;
public Src t_src;
public Alt t_top;
public Tregion t_region;
public Title t_title;
public Abstract t_abs;
public Author t_author;
public Copyright t_copy;
public Longdesc t_desc;
public Begin t_begin;
public End t_end;
public Dur t_dur;
      public Tiatrib(Type tp,Id id,Tregion region,Src src,
                Alt top,
                Title title,Abstract abs,
                Author author,Copyright copy,
                Longdesc desc,Begin begin,
                End end,Dur dur){
                t_tp = tp;
                t_id = id;
                t_src = src;
                t_top = top;
                t_region = region;
                t_title = title;
                t_abs = abs;
                t_author = author;
                t_copy = copy;
                t_desc = desc;
                t_begin = begin;
                t_end = end;
                t_dur = dur;
        }
}
```

```
/*
 * non-terminal Title   *
 */
package graphe;
public class Title{
public static String t_id;
      public Title(String ident){
        t_id=ident;
        }
      public Title(){
        t_id=null;
        }
}

/*
 * non-terminal TOP    *
 */
package graphe;
public class Top{
public static String t_id;
      public Top(String ident){
        t_id=ident;
        }
      public Top(){
        t_id=null;
        }
}

/*
 * non-terminal Tregion *
 */
package graphe;
public class Tregion{
public String t_name;
      public Tregion(String ident){
        t_name=ident;
        }
      public Tregion(){
        t_name=null;
        }

}

/*
 * non-terminal NTXT *
 */
package graphe;
public class Txt{
public String t_name;
      public Txt(String ident){
        t_name=ident;
        }
}

/*
 * non-terminal TYPE    *
 */
package graphe;
public class Type{
public String t_id;
```

```
      public Type(String ident){
        t_id=ident;
        }
public Type(){
        t_id=null;
        }
}

/*
 *media video
 */
 package graphe;
 public class Video extends Media{
 public Avatrib v_atrib;
       public Video(String video,Avatrib atrib){
        if (video.equals("video")){
        this.typemedia = "video";
        v_atrib = atrib;
        }
      }
  }
/*
 * non-terminal WIDTH
 */
package graphe;
public class Width{
public String w_id;
      public Width(String ident){
        w_id=ident;
        }
      public Width(){
        w_id=null;
        }
}
```

Annexe 6 :Package parseur

```
/* le parseur Smil */
package graphe.parseur;
import java.io.*;
import javax.swing.*;
import traitementListe.*;
import traitementFichier.*;
import graphe.*;
import SmilApplication;
import coordinateur.PanelSmil;
import SmilApplication;

public class ParseurSmil{
public parser p;
public SmilApplication frame;
public ParseurSmil(String fichiersmil,SmilApplication ff){
        frame=ff;
        try {
        p = new parser(new Scanner(new FileReader(fichiersmil)));
        Object result = p.parse().value;
        }
        catch (Exception e) {
        String reponse ="il y a eu un probléme"+"\n"+
        "de l'analyse lexical et syntaxique"+"\n"+ e.toString();
        JOptionPane.showMessageDialog(null,reponse);
        }
   }

/*
 ************************************************************************
 * Cette fonction traite le document smil pour pouvoir l'afficher       *
 ************************************************************************
 */

public void TraiteSmil(Smil s) {
        Nheadbody nb = s.s_hb;
        Id id = s.s_id;
        /* identificateur non absent */
        if ((id.i_id != null) && (nb.n_hd != null))
         {

           TraiteId(id,"idsmil");
           TraiteHead(nb.n_hd);
           frame.body = nb.n_bd;
           }else
        if ((id.i_id == null) && (nb.n_hd != null))
           {
           TraiteHead(nb.n_hd);
           frame.body = nb.n_bd;
           }else
        if ((id.i_id != null) && (nb.n_hd == null))
           {
           TraiteId(id,"idsmil");
           frame.body = nb.n_bd;
           }else
           {
           frame.body = nb.n_bd;
           }
```

```
   }

/*
 ***************************************************************************
 * Cette fonction traite l'identificateur                                  *
 ***************************************************************************
 */

public void TraiteId(Id i,String key) {

        frame.id.put(key,i);
         }

public void TraiteRoot(Rootlayout m,String key) {

        frame.root.put(key,m);
         }

/*
 ***************************************************************************
 * Cette classe traite les informations contenus dans le head si il y a en  *
 ***************************************************************************
 */

public void TraiteHead(Head h) {

        Id id = h.h_id;
        ListeMeta lm = h.h_meta;
        Layout h_la = h.h_lay;
        if ((lm != null)  && (h_la == null))//identificateur et meta non
absent non absent
                {
                if (id.i_id != null) {
                         TraiteId(id,"idheadmeta");
                         TraiteSuiteMeta(lm);
                         //taille par defaut
                         frame.changerTaille(300,240);

                        }else
                        {
                         TraiteSuiteMeta(lm);
                         frame.changerTaille(300,240);
                        }
                }else
        if ((lm == null)  && (h_la != null))//identificateur et layout non
absent non absent
                {
                if (id.i_id != null) {
                        TraiteId(id,"idheadlayout");
                        TraiteNlayout(h_la);
                        }else
                        {
                        TraiteNlayout(h_la);

                        }
                } else
         if ((lm != null)  && (h_la != null))
                {
```

```
                if (id.i_id != null) {
                        TraiteId(id,"idheadmetalayout");
                        TraiteSuiteMeta(lm);
                        TraiteNlayout(h_la);
                        }else
                       {

                        TraiteSuiteMeta(lm);
                        TraiteNlayout(h_la);
                        }
                }


    }

/*
 ************************************************************************
 * Cette fonction traite les informations dans la production nlayout si *
 * il y a en                                                            *
 ************************************************************************
 */

public void TraiteNlayout(Layout m) {

        if (m.l_choix == 1){
        TraiteNregion(m.l_reg);
        }else
        if (m.l_choix == 2){
        TraiteNrootlayout(m.l_root);
        }else
        if (m.l_choix == 0){
        TraiteNldefaut(m.l_def);
        }

    }

/*
 ************************************************************************
 * Cette fonction traite les informations contenus dans le head si      *
 * il y a en                                                            *
 ************************************************************************
 */

public void TraiteSuiteMeta(ListeMeta t) {
          Meta m =null;
        String sm = null;

          int lg = t.ListeLength();

                    m = t.ListeMetaGet(lg);
                    sm ="metaname" + m.m_name.n_name;
                    TraiteMeta(m,sm);

          for (int i=1; i< lg;i++)
                {
                    m = t.ListeMetaGet(i);
                    sm ="metaname" + m.m_name.n_name;
                    TraiteMeta(m,sm);
                }
```

```
        }
/*
 ****************************************************************************
 * Cette fonction traite les informations dans la production nregion        *
 * si il y a en                                                             *
 ****************************************************************************
 */

public void TraiteNregion(Lregion m) {

        String s = null;

        if (m.l_type.t_id != null){
        s = m.l_reg.r_id.i_id;
        TraiteType(m.l_type,"typelayoutregion");
        TraiteRegion(m.l_reg,s);

            }else
        {
        TraiteRegion(m.l_reg,s);
        }

    }
/*
 ****************************************************************************
 * Cette fonction traite les informations dans la production nregion si     *
 * il y a en                                                                *
 ****************************************************************************
 */


public void TraiteNldefaut(Ldefaut m) {

        if (m.l_type.t_id != null){
        TraiteType(m.l_type,"typelayoutbasic");
        frame.changerTaille(300,250);//taille par defaut

        }

    }
/*
 ****************************************************************************
 * Cette fonction traite les informations dans la production nrootlayout*
 * si  il y a en                                                            *
 ****************************************************************************
 */

public void TraiteNrootlayout(Lrootlayout m) {
        if (m.l_type.t_id != null){
        TraiteType(m.l_type,"typelayoutroot");
        TraiteRootlayout(m.l_root);
        TraiteListeRegion(m.l_reg);
        } else
        {
        TraiteRootlayout(m.l_root);
        TraiteListeRegion(m.l_reg);
        }


    }
```

```
/*
 ************************************************************************
 * Cette fonction traite les informations dans la liste des metas  si   *
 * il y a en                                                            *
 ************************************************************************
 */


public void TraiteMeta(Meta m, String s) {
      frame.meta.put(s,m);

        }
/*
 ************************************************************************
 * Cette fonction traite les informations dans la production type       *
 * si  il y a en                                                        *
 ************************************************************************
 */

public void TraiteType(Type m,String s) {
      frame.type.put(s,m);
    }


/*
 ************************************************************************
 * Cette fonction traite les informations dans la production dregion si *
 * il y a en                                                            *
 ************************************************************************
 */

public void TraiteRegion(Region tl, String s) {
           frame.region.put(s,tl);
        }
/*
 ************************************************************************
 * Cette fonctionclasse traite les informations dans la production      *
 * drootlayout  si il y a en                                            *
*
 ************************************************************************
 */

public void TraiteRootlayout(Rootlayout m) {

        if (m.r_id.i_id != null)
                {
                TraiteRoot( m,"idrootlayout");
                int widht = Integer.parseInt(m.r_width.w_id);
                int height =Integer.parseInt(m.r_height.h_id);
                // change la taille du panel principale
                // et la couleur du font d'ecran
                frame.changerTaille(widht,height);
                /*if (m.r_back.i_id != null)
                frame.changerCouleur(m.r_back.i_id)  refaire cette methode */
                }
                else
                {
                int widht = Integer.parseInt(m.r_width.w_id);
                int height =Integer.parseInt(m.r_height.h_id);
                // change la taille du panel principale
```

```
                // et la couleur du font d'ecran
                frame.changerTaille(widht,height);
                /*if (m.r_back.i_id != null)
                frame.changerCouleur(m.r_back.i_id)  refaire cette methode */
               }

    }

/*
 ************************************************************************
 * Cette fonction traite les informations dans la production suiteregion*
 * si  il y a en                                                        *
 ************************************************************************
 */

public void TraiteListeRegion(ListeRegion m) {
        Region r = null;
        String s = null;
        int lg = m.ListeLength();

        for ( int i = lg;i>=1 ;i--)
              {
                r = m.ListeRegionGet(i);
                s = "idregion" + r.r_id.i_id;
                TraiteRegion(r,s);
              }
   }

/*
 ************************************************************
 * cette fonction traite le corps du fichier smil           *
 ************************************************************
 */

public void TraiteBody(Body body){
      Id i= body.b_id;
      ListeMedia l = body.b_body;
      if (i.i_id != null)
            {
            frame.gmenu.TraiteId(i,"idbody");
            TraiteScenario(l);
            }else
            {
            TraiteScenario(l);

            }
      }

/*
 ************************************************************
 * cette fonction gere la liste des medias dans le corps    *
 ************************************************************
 */
public void TraiteScenario(ListeMedia l){
      int lg = l.ListeLength();
      if (lg != 0)
      {
      Media m =l.ListeMediaGet(lg);
      TraiteMedia(m);
      for(int i=1;i<lg;i++)
```

```
            {m =l.ListeMediaGet(i);
            TraiteMedia(m);
            }
      }
   }

public void TraiteMedia(Media m){
if (m.typeMedia().equals("img"))
      {
            TraiteImg((Img)m);
            }
            else
            if (m.typeMedia().equals("txt"))
      {
                  TraiteText((Text)m);
                  }
                  else
                  if (m.typeMedia().equals("son"))
{
                  TraiteSon((Son)m);}
                        else
                        if (m.typeMedia().equals("video"))
{
                        TraiteVideo((Video)m);}
                        else
                        if (m.typeMedia().equals("seq"))
{                 TraiteSeq((Seq)m);}
                        else
{                 TraitePar((Par)m);}
   }

/*
 ************************************************************
 * cette fonction se charge de gere le media  Img           *
 ************************************************************
 */

public void TraiteImg(Img m){

Simg simg = null;
String s = m.i_atrib.t_region.t_name;
if (s!= null)
      {
      frame.img.put(s,m);
      simg = new Simg(m,frame);
      frame.ajouterPanel(simg);
      }else
      {
      simg = new Simg(m,frame);
      frame.ajouterPanel(simg);
      }

}



/*
 ************************************************************
 * cette fonction se charge de gere le media  Text          *
 ************************************************************
```

Annexe 7 :package traitementListe

```
/*
 ****************************************************************
 * la classe liste implemente les fonctions g,neriques pour     *
 * le traitement des listes d'object quelconques                *
 ****************************************************************
 */

package traitementListe;
public class Liste {
    Object element;
    Liste next;
    Liste tete;

   //creer une nouvelle liste
   public Liste(){
        tete=null;
     }
  //decale les elements de i,i+1 de une unite
   //vers la droite  et insere "element" a la position i

   public void ListeSet(int i,Object elem){
        Liste courant,preced,m;
        int s=1;
        courant=tete;
        preced=null;
        while(s<i){
            preced=courant;
            courant=courant.next;
            s++;
        }
        if (preced==null){
            m=new Liste();
            m.element=elem;
            m.next=courant;
            tete=m;
        }else{
            m=new Liste();
            m.element=elem;
            preced.next=m;
            m.next=courant;
        }
    }

   //retourne la longeur de la liste

   public int ListeLength(){
        int i=0;
        Liste l=tete;
        while(l!=null){
            i++;
            l=l.next;
        }
        return i;
    }

   //supprime l'element … la position i de la liste
```

```
   public void ListeRemove(int i){
        int s=1;
        Liste courant=tete;
        Liste preced=null;
        while(s<i){
            preced=courant;
            courant=courant.next;
            s++;
        }
        preced.next=courant.next;
    }
  //retourne l'element qui est … la position i

  public  Object ListeGet(int i){

        int s=1;
        Liste courant=tete;
        while(s<i){
            courant=courant.next;
            s++;
        }
        return (courant.element);
    }

}

/*
 ****************************************************************
 * la classe ListeInt implemente les fonctions  pour            *
 * le traitement des listes d'entier                            *
 ****************************************************************
 */
package traitementListe;
import graphe.*;
public class ListeInt extends Liste  {

  public ListeInt(){
        super();
        }

   public int ListeIntGet(int i){
        Integer s =(Integer)super.ListeGet(i);
        return s.intValue();
    }

   public void ListeIntSet(int i,int elem){
      Integer s = new Integer(elem);
      super.ListeSet(i,s);
    }

}

/*
 ****************************************************************
 * la classe ListeMedia implemente les fonctions  pour          *
 * le traitement des listes de Media(voir package graphe        *
*
 ****************************************************************
 */
package traitementListe;
```

```
import graphe.*;
public class ListeMedia extends Liste  {

   public ListeMedia(){
        super();
        }


   public Media ListeMediaGet(int i){
        Media s=(Media)super.ListeGet(i);
        return s;
    }

   public void ListeMediaSet(int i,Media elem){
        super.ListeSet(i,(Media)elem);
    }

}

/*
 ****************************************************************
 * la classe ListeMeta implemente les fonctions  pour           *
 * le traitement des listes de Meta(voir package graphe         *
*
 ****************************************************************
 */
package traitementListe;
import graphe.*;
public class ListeMeta extends Liste  {

   public ListeMeta(){
        super();
        }


   public Meta ListeMetaGet(int i){
        Meta s=(Meta)super.ListeGet(i);
        return s;
    }

   public void ListeMetaSet(int i,Meta elem){
        super.ListeSet(i,(Meta)elem);
    }

}

/*
 ****************************************************************
 * la classe ListeRegion implemente les fonctions  pour         *
 * le traitement des listes de Region(voir package graphe       *
*
 ****************************************************************
 */
package traitementListe;
import graphe.*;
public class ListeRegion extends Liste  {

   public ListeRegion(){
        super();
        }
```

```
   public Region ListeRegionGet(int i){
        Region s=(Region)super.ListeGet(i);
        return s;
    }

   public void ListeRegionSet(int i,Region elem){
        super.ListeSet(i,(Region)elem);
    }

}

/*
 ****************************************************************
 * la classe ListeMeta implemente les fonctions  pour           *
 * le traitement des listes de Serveur (voir package graphe     *
*
 ****************************************************************
 */
package traitementListe;
import graphe.*;
import serveur.*;
public class ListeServeur extends Liste  {

   public ListeServeur(){
        super();
        }


   public Serveur ListeServeurGet(int i){
        Serveur s=(Serveur)super.ListeGet(i);
        return s;
    }

   public void ListeServeurSet(int i,Serveur elem){
        super.ListeSet(i,(Serveur)elem);
    }

}
```

Annexe 8 : package serveur

```
/*
 ********************************************************************************
 * cette classe implemente les fonctions géneris=que pour tous les serveurs *
 ********************************************************************************
 */

package serveur ;

import coordinateur.*;
import traitementListe.*;
import graphe.*;
import graphe.parseur.*;
import SmilApplication;

public class Serveur{

public Media m;
public String nomfile;
public SmilApplication f;
public int left,top,width,height;
public String dur,begin,end;

public Serveur(Media media,SmilApplication frame){
      m = media;
      f = frame;
      getSrc();
      getDur();
      getBegin();
      getEnd();
      regionAffichage();

      }

/* cette methode fournie la durée de presentatiopn du media s'il y a en */

public void getDur(){
      String s = m.typeMedia();
      if  (s.equals("txt") )
            {
            Text tt = (Text)m;
            Tiatrib t = tt.t_atrib;
            if ( t.t_dur.d_name != null)
            dur=t.t_dur.d_name;
            }else
            if (s.equals("img"))
            {
            Img mm = (Img)m;
            Tiatrib t = mm.i_atrib;
            if ( t.t_dur.d_name != null)
              {
                  dur = t.t_dur.d_name;
                  }
            }else
            if (s.equals("son"))
            {
            Son ss = (Son)m;
            Avatrib a = ss.s_atrib;
                Tiatrib t = a.a_atrib;
```

```
                if ( t.t_dur.d_name != null)
                dur=t.t_dur.d_name;
            }else
            {
            Video vv = (Video)m;
            Avatrib a= vv.v_atrib;
            Tiatrib t = a.a_atrib;
            if ( t.t_dur.d_name != null)
            dur=t.t_dur.d_name;
            }

      }

/* cette methode fournie l'instant de debut pour la presentatiopn du media
s'il y a en */

public void getBegin(){
      String s = m.typeMedia();
      if  (s.equals("txt") )
            {
            Text tt = (Text)m;
            Tiatrib t = tt.t_atrib;
            if (t.t_begin.b_name != null)
            begin = t.t_begin.b_name;
            }else
            if (s.equals("img"))
            {
            Img mm = (Img)m;
            Tiatrib t = mm.i_atrib;
            if (t.t_begin.b_name != null)
            begin = t.t_begin.b_name;
            }else
            if (s.equals("son"))
            {
            Son ss = (Son)m;
            Avatrib a = ss.s_atrib;
            Tiatrib t = a.a_atrib;
            if (t.t_begin.b_name != null)
            begin=t.t_begin.b_name;
            }else
            {
            Video vv = (Video)m;
            Avatrib a= vv.v_atrib;
            Tiatrib t = a.a_atrib;
            if (t.t_begin.b_name != null)
            begin=t.t_begin.b_name;
            }

      }

/* cette methode fournie l'instant de fin pour la presentatiopn du media s'il
y a en */

public void getEnd(){
      String s = m.typeMedia();
      if  (s.equals("txt") )
            {
            Text tt = (Text)m;
            Tiatrib t = tt.t_atrib;
            if (t.t_end.e_name != null)
```

```
            end=t.t_end.e_name;
            }else
            if (s.equals("img"))
            {
            Img mm = (Img)m;
            Tiatrib t = mm.i_atrib;
            if (t.t_end.e_name != null)
            end=t.t_end.e_name;
            }else
            if (s.equals("son"))
            {
            Son ss = (Son)m;
            Avatrib a = ss.s_atrib;
            Tiatrib t = a.a_atrib;
            if (t.t_end.e_name != null)
            end=t.t_end.e_name;
            }else
            {
            Video vv = (Video)m;
            Avatrib a= vv.v_atrib;
            Tiatrib t = a.a_atrib;
            if (t.t_end.e_name != null)
            end=t.t_end.e_name;
            }

      }
/* cette methode fournie la source du media */

public void getSrc(){
        String s=m.typeMedia();
        String separator =System.getProperty("file.separator");
      /*cas du fichier qui se trouve dans le meme repertoire
      que le fichier smil, on consulte le hashtable meta*/
      if ((f.meta.isEmpty()== false)||(f.meta.get("metanamebase")==null))
         {
            if  (s.equals("txt") )
            {
            Tiatrib t = ((Text)m).t_atrib;
            nomfile = f.nomdir+separator+t.t_src.s_name;
            }else
            if (s.equals("img"))
            {
            Tiatrib t = ((Img)m).i_atrib;
            nomfile = f.nomdir+separator+t.t_src.s_name;
            }else
            if (s.equals("son"))
            {
            Avatrib a = ((Son)m).s_atrib;
            Tiatrib t = a.a_atrib;
            nomfile = f.nomdir+separator+t.t_src.s_name;
            }else
            {
            Avatrib a= ((Video)m).v_atrib;
            Tiatrib t = a.a_atrib;
            nomfile = f.nomdir+separator+t.t_src.s_name;
            }
        }else
            /* cas ou il faut ajouter la partie base pour avoir le chemin
complet*/
                if ( f.meta.get("metanamebase")==null)
```

```
        {
            if  (s.equals("txt") )
            {
            Tiatrib t = ((Text)m).t_atrib;
            Meta m = (Meta)f.meta.get("metanamebase");
            nomfile = m.m_name.n_name+"/"+ t.t_src.s_name;
            }else
            if (s.equals("img"))
            {
            Tiatrib t = ((Img)m).i_atrib;
            Meta m = (Meta)f.meta.get("metanamebase");
            nomfile = m.m_name.n_name+"/"+ t.t_src.s_name;
            }else
            if (s.equals("son"))
            {
            Avatrib a = ((Son)m).s_atrib;
            Tiatrib t = a.a_atrib;
            Meta m = (Meta)(f.meta.get("metanamebase"));
            nomfile = m.m_name.n_name+"/"+ t.t_src.s_name;
            }else
            {
            Avatrib a= ((Video)m).v_atrib;
            Tiatrib t = a.a_atrib;
            Meta m = (Meta)(f.meta.get("metanamebase"));
            nomfile = m.m_name.n_name+"/"+ t.t_src.s_name;
            }

         }

      }

/* cette methode teste s'ilexiste la region definie par l'attribut region et
lance
le calcul  des informations de disposition pour l'affichage
 du media par rapport à une region s'ily a en*/

public void regionAffichage(){
      String s = m.typeMedia();
      if  (s.equals("txt") )
            {Text tt =(Text)m;
            Tiatrib t = tt.t_atrib;
            if (t.t_region.t_name != null)
                  {if (f.region.get("idregion"+t.t_region.t_name)!= null)
                  getLeftTopWidthHeight("idregion"+t.t_region.t_name);
                  }
            }else
            if (s.equals("img"))
            {Img mm=(Img)m;
            Tiatrib t = mm.i_atrib;
            if (t.t_region.t_name != null)
                  {if (f.region.get("idregion"+t.t_region.t_name)!= null)
                  getLeftTopWidthHeight("idregion"+t.t_region.t_name);
                  }

            }else
            if (s.equals("son"))
            {
            Son ss = (Son)m;
            Avatrib a = ss.s_atrib;
            Tiatrib t = a.a_atrib;
```

```
            if (t.t_region.t_name != null)
                  {if (f.region.get("idregion"+t.t_region.t_name)!= null)
                  getLeftTopWidthHeight("idregion"+t.t_region.t_name);
                  }

            }else
            {
            Video vv = (Video)m;
            Avatrib a= vv.v_atrib;
            Tiatrib t = a.a_atrib;
            if (t.t_region.t_name != null)
                  {if (f.region.get("idregion"+t.t_region.t_name)!= null)

                  getLeftTopWidthHeight("idregion"+t.t_region.t_name);
                  }

            }
      }

/*des informations de disposition pour l'affichage
 du media par rapport à une region s'ily a en*/

public void getLeftTopWidthHeight(String s){
      Region r = (Region)(f.region.get(s));
      String l = r.r_left.l_id;
      String lfin = l.substring((l.length()-1),l.length());
      String ltete = l.substring(0,(l.length()-1));
      String t = r.r_top.t_id;
      String tfin = t.substring((t.length()-1),t.length());
      String ttete = t.substring(0,(t.length()-1));
      String w = r.r_width.w_id;
      String wfin = w.substring((w.length()-1),w.length());
      String wtete = w.substring(0,(w.length()-1));
      String h = r.r_height.h_id;
      String hfin = h.substring((h.length()-1),h.length());
      String htete = h.substring(0,(h.length()-1));
      Rootlayout m = (Rootlayout)f.root.get("idrootlayout");
      int rootwidth = Integer.parseInt(m.r_width.w_id);
      int rootheight =Integer.parseInt(m.r_height.h_id);
      //si la largeur est en pourcentage il faut convertir en pixels
      //par rapport au root layout
      if ( lfin.equals("%")&& tfin.equals("%")&& wfin.equals("%")&&
hfin.equals("%"))
            {

            left = ((Integer.parseInt(ltete))*rootwidth)/100;
            top = ((Integer.parseInt(ttete))*rootheight)/100;
            width = ((Integer.parseInt(wtete))*rootwidth)/100;
            height = ((Integer.parseInt(htete))*rootheight)/100;
            }else
            {
            left = Integer.parseInt(l);
            top = Integer.parseInt(t);
            width = Integer.parseInt(w);
            height = Integer.parseInt(h);

            }
      }
```

```
public int convertirmseconde(){
      int msec =0;
if(!dur.equals("indefinite"))
{
      String msectete=dur.substring(0,(dur.length()-1));
      String msecfinsh =dur.substring((dur.length()-1),dur.length());
      //voir si la durée est en secondes ex 5s ou en heure ex 5h

      if(msecfinsh.equals("s"))
            {msec = (Integer.parseInt(msectete))*1000;}
            else
      if(msecfinsh.equals("h"))
            {msec = (Integer.parseInt(msectete))*3600*1000;}
            else
      {
       String msectetems =dur.substring(0,(dur.length()-2));
       String msecfinms =dur.substring((dur.length()-2),dur.length());
       //voir si la durée est en ms ex 5ms
            if (msecfinms.equals("ms"))
            {msec = (Integer.parseInt(msectetems));}
            else
            {
             String msectetemin =dur.substring(0,(dur.length()-3));
             String msecfinmin =dur.substring((dur.length()-3),dur.length());
             //voir si la durée est en min ex 5min
             if (msecfinmin.equals("min"))
             {msec = (Integer.parseInt(msectetemin))*60*1000;}
            }
      }
 }
return msec;
}

}
package serveur.serveurImg ;
import coordinateur.*;
import traitementListe.*;
import SmilApplication;
import graphe.*;
import graphe.parseur.*;
import serveur.*;
import java.net.*;
import java.io.*;
import java.awt.*;
public class Simg extends Serveur{

public Image image;
public Simg(Img img,SmilApplication frame){
      super((Media)img,frame);
      getImg();
      }

public void getImg(){
        image = Toolkit.getDefaultToolkit().getImage(nomfile);

   }
}
```

```
package serveur.serveurText ;
import coordinateur.*;
import traitementListe.*;
import SmilApplication;
import graphe.*;
import graphe.parseur.*;
import serveur.*;
import java.net.*;
import java.io.*;
import java.util.*;
public class Stext extends Serveur{
public SmilApplication f;
public int iligne=0;
public String[] text=null;
public Stext(Text txt,SmilApplication frame){
        super((Media)txt,frame);
      getText();
      }

public void getText(){
      StringTokenizer st =new StringTokenizer(nomfile,":");
       text = new String[20];//par default text de 20 ligne
      if (st.hasMoreTokens()){
        String s=st.nextToken();
               if (! s.equals("http"))
           {

           String line;
           try{
                 FileReader file = new FileReader(nomfile);
                 BufferedReader buferfile = new BufferedReader(file);
                 while((line = buferfile.readLine()) !=null) {
                 text[iligne] = line ;iligne++;
                             }
                 buferfile.close();

             }catch(IOException e) {
           text[0]="impossible de lire le fichier" + e.toString();
          }

       }else//ouvrir connection et lire le text
      {
        URL page = null;
        try {page =new URL(nomfile);}
        catch(MalformedURLException e){System.out.println("Mauvaise
URL:"+page);}
        URLConnection conn = null;
      InputStreamReader in;
      BufferedReader d;
      String line;
      try{
            conn = page.openConnection();
            conn.connect();
            in=new InputStreamReader(conn.getInputStream());
            d= new BufferedReader(in);
            while((line = d.readLine()) !=null) {
                  text[iligne] = line;iligne++;
                              }
                  d.close();
      }catch(IOException e) {
```

```
            text[0]="impossible de lire le fichier" + e.toString();
           }
      }
   }
}

}
```

```
package serveur.serveurVideo ;
import java.awt.*;
import java.net.*;
import javax.media.*;
import javax.swing.*;
import java.io.*;
import java.awt.event.*;
import java.util.*;
import SmilApplication;
import coordinateur.*;
import traitementListe.*;
import graphe.*;
import graphe.parseur.*;
import serveur.*;
public class Svideo extends Serveur{
public Player p =null;
public Svideo(Video video,SmilApplication frame){
      super((Media)video,frame);
      getVideo();
    }
 public void commencer(){
        p.start();
        }

 public void arreter(){
        p.stop();
        p.deallocate();
        }

 public void detruire(){
        p.close();
        }

public void getVideo() {
      MediaLocator murl = null;
      try{
            if ((murl = new MediaLocator("file:/"+nomfile))==null)
            {System.err.println("Mauvais URL " + "file:/"+nomfile);}
            p = Manager.createPlayer(murl);
            }catch(Exception e){
           System.err.println("Mauvais URL "+e);
           }
      }

}
```

```
package serveur.serveurSon ;
import java.awt.*;
import java.net.*;
import javax.media.*;
```

```
import javax.swing.*;
import java.io.*;
import java.awt.event.*;
import java.util.*;
import SmilApplication;
import coordinateur.*;
import traitementListe.*;
import graphe.*;
import graphe.parseur.*;
import serveur.*;
public class Sson extends Serveur{
public Player p =null;
public Sson(Son son,SmilApplication frame){
      super((Media)son,frame);
      getSon();
    }

public void commencer(){
        p.start();
        }

public void arreter(){
        p.stop();
        p.deallocate();
        }

 public void detruire(){
        p.close();
        }

public void getSon() {
      MediaLocator murl = null;
      try{
            if ((murl = new MediaLocator("file:/"+nomfile))==null)
            {System.err.println("Mauvais URL " + "file:/"+nomfile);}
            p = Manager.createPlayer(murl);
            }catch(Exception e){
           System.err.println("Mauvais URL "+e);
           }
      }

}
```

Annexe 9 : package traitementFichier

```
/*
 ************************************************************************
 * cette classe montre le fichier source qu'on va visualiser , ce fichier *
 * sous format texte ou ascii ou Unicode                                  *
 * il est affiche dons son propre fenetre                                 *
 ************************************************************************
 */

package traitementFichier;
import java.awt.*;
import java.awt.event.*;
import java.lang.String;
import java.io.*;
import java.util.Properties;
import javax.media.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.BorderFactory;
import javax.swing.filechooser.*;
import SmilApplication;
public class FenetreSource extends JDialog{
public SmilApplication frame;
public FenetreSource(SmilApplication parent,String titre,boolean modale){
        super(parent,titre,modale);
        frame=parent;
        JPanel panelsource =new JPanel();
        JTextArea txtsource = new JTextArea();
        JScrollPane scrollpanel = new JScrollPane(panelsource);
        txtsource.setMargin(new Insets(5,5,5,5));
        txtsource.setEditable(false);
        if (frame.nomfichier != null)
               {txtsource.setText(lirefichier(frame.nomfichier));}
                else txtsource.setText("il n ya pas de fichier ouvert");

        Dimension ecran =Toolkit.getDefaultToolkit().getScreenSize();
        setBounds( ecran.width/4,(ecran.height*2)/3,
                   (ecran.width*7)/10,
                   (ecran.height*5)/12

                  );


        panelsource.add(txtsource);
        setContentPane(scrollpanel);

        }

  public String lirefichier(String nomfile){
        StringBuffer buffer = new StringBuffer();
        String line;
        try{
            FileReader file = new FileReader(nomfile);
            BufferedReader buferfile = new BufferedReader(file);
            while((line = buferfile.readLine()) !=null) {
                buffer.append(line + "\n");
                }
              buferfile.close();
              return buffer.toString();
```

```
            }catch(IOException e) {
            return ("impossible de lire le fichier" + e.toString());
            }

        }

    }


/*
 ************************************************************
 *ce module implemente les fonctions pour selectionner      *
 *seulement les fichier .smim ou .SMIL                      *
 ************************************************************
 */
package traitementFichier;
import java.awt.*;
import java.awt.event.*;
import java.lang.String;
import java.net.URL;
import java.net.MalformedURLException;
import java.io.*;
import java.util.Properties;
import javax.media.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.BorderFactory;
import javax.swing.filechooser.*;
public class SmilFilter extends FileFilter {

    public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }

        String extension = UtilsSmil.getExtension(f);
        if (extension != null) {
            if (extension.equals(UtilsSmil.smil)) {
                    return true;
            } else {
                return false;
            }
      }

        return false;
    }


    //  description de ce filtre
    public String getDescription() {
        return "*.smil ou *.SMIL";
    }
}


class UtilsSmil {

    public final static String smil = "smil";

    /*
```

```
     * Get the extension of a file.
     */
    public static String getExtension(File f) {
        String ext = null;
        String s = f.getName();
        int i = s.lastIndexOf('.');

        if (i > 0 &&  i < s.length() - 1) {
            ext = s.substring(i+1).toLowerCase();
        }
        return ext;
    }
}
```