POLYPUBLIE
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Création automatique des animations 3D
Title:

Auteur: Ricardo Marcos Polar Hito
Author:

Date: 2012

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Polar Hito, R. M. (2012). Création automatique des animations 3D [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/815/
Citation:

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: https://publications.polymtl.ca/815/
PolyPublie URL:

Directeurs de recherche: Michel Gagnon, & Benoît Ozell
Advisors:

Programme: Génie informatique
Program:

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

CRÉATION AUTOMATIQUE DES ANIMATIONS 3D

RICARDO MARCOS POLAR HITO
DÉPARTEMENT DE GÉNIE IINFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AVRIL 2012

© Ricardo Marcos Polar Hito, 2012.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CRÉATION AUTOMATIQUE DES ANIMATIONS 3D

présenté par : POLAR HITO, Ricardo Marcos
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de :

Mme. BELLAICHE, Martine, Ph.D., présidente.
M. OZELL, Benoit, Ph.D., membre et directeur de recherche.
M. GAGNON, Michel, Ph.D., membre et codirecteur de recherche.
M. PAL, Christopher J., Ph.D., membre.

REMERCIEMENTS

Je tiens à remercier les gens ayant contribué à la réalisation de cette thèse. Je voudrais tout d'abord exprimer ma reconnaissance à mes directeur et codirecteur de recherche, Benoît Ozell et Michel Gagnon, pour leur soutien, leur disponibilité, leur mentorat, leur enthousiasme, leur support financier et les nombreuses opportunités qu'ils m'ont offertes.

J'aimerais aussi remercier Laurent Ruhlmann, associé de recherche du département de génie informatique et génie logiciel qui a travaillé avec notre équipe d'infographie, pour son aide au cours de mes travaux de recherche, pour ses idées et ses commentaires constructifs ; ainsi qu'à Daniel Petels, pour les modèles 3D et animations prédéfinies utilisés dans ma recherche.

Je tiens à remercier ma famille qui m'a toujours encouragé et le soutien inconditionnel qui m'a toujours été donné. Un remerciement spécial à mon père que j'ai toujours admiré et qui a été une source d'inspiration et motivation au cours de mes études.

RÉSUMÉ

La production traditionnelle d'animations 3D pour un jeu vidéo ou un film d'animation est un processus lourd. Les animateurs ont besoin de plusieurs années de pratique et de bons logiciels de création de contenu numérique pour réussir à créer des animations 3D. Cela est dû à la complexité du logiciel et à la complexité de la tâche. La création d'un cycle de marche réaliste dans une scène complexe nécessite de nombreux détails de bas niveau pour atteindre un niveau élevé de réalisme. Ce mémoire propose une vue de haut niveau dans la création automatique des animations 3D afin de simplifier le processus global de production de l'animation.

Afin d'aborder cette problématique, l'objectif général de la recherche a consisté à élaborer un prototype logiciel capable de générer automatiquement des animations 3D qui représentent le sens d'une phrase simple. Ce projet faisait partie intégrale du projet GITAN dans le domaine de l'infographie. GITAN proposait une solution pour générer des animations 3D à partir du texte. La solution proposée dans ce mémoire constitue principalement le module graphique qui génère la scène 3D animée qui représente la phrase d'entrée. Avec ce système, la complexité de la construction de la scène animée est considérablement réduite, puisque nous utilisons une représentation textuelle pour décrire l'animation et les différents objets dans la scène.

La revue bibliographique a suggéré que les systèmes semblables qui permettent de générer automatiquement des animations 3D à partir du texte sont souvent très orientés vers un domaine d'application spécifique, par exemple les accidents automobiles, les comportements ou les interactions des personnages. L'automatisation de la génération de la scène sur ces systèmes se base souvent sur des langages script ou des formalismes qui étaient souvent orientés au domaine d'application. De plus, nous voulions générer l'animation en utilisant un format d'échange 3D à la place d'afficher directement l'animation. Nous pensons que l'utilisation d'un format d'échange 3D nous permet de bien générer la scène 3D, puisqu'un bon format d'échange intermédiaire permet de bien définir une animation de façon standard et fournit des outils nécessaires pour son utilisation. Pour cette raison, nous avons utilisé COLLADA comme format 3D pour représenter nos animations. D'après ces observations, nous avons émis trois hypothèses de recherche. La première supposait qu'il était possible de créer un formalisme capable de décrire une scène animée à partir d'une phrase simple. Le formalisme nous permet de faire une description de la scène animée en utilisant des noeuds, des contraintes et des images clés. La deuxième hypothèse supposait qu'il est possible de traduire le script qui décrit la scène vers le fichier COLLADA. Nous avons proposé un système

logiciel qui permet de traduire le script vers un fichier COLLADA qui contient l'animation 3D. Finalement, la troisième hypothèse supposait que l'animation générée par le système permet de communiquer le sens de la phrase initiale. Le système doit pouvoir communiquer le message de la phrase qui décrit la scène vers les observateurs.

Pour tester ces hypothèses, la méthodologie que nous avons retenue consiste, premièrement, à la création du formalisme qui permet de décrire la scène 3D. Nous avons proposé un schéma XML qui permet de déclarer des noeuds, des animations prédéfinies, des contraintes et des images clés qui décrivent la scène à générer. Par la suite, nous avons proposé une architecture logicielle modulaire qui traduit le script vers le fichier COLLADA. Le système utilise des algorithmes pour positionner correctement les objets dans la scène et pour synchroniser les animations. Finalement, nous avons effectué un sondage pour valider la communication du message par les scènes 3D générées. Le résultat du sondage nous permet d'analyser la compréhension du message par les observateurs et l'influence de l'environnement de la scène 3D sur le message, et ainsi, déterminer s'il est possible de transmettre le sens de la phrase initiale avec l'animation 3D.

Les résultats que nous avons obtenus sont très satisfaisants. Nous avons été capables de décrire les scènes avec le formalisme proposé. De plus, le système logiciel génère des fichiers COLLADA bien structurés et il est capable de générer deux types de scènes : des scènes statiques et des scènes animées. Finalement, l'analyse des résultats du sondage montre que les scènes animées permettent de mieux communiquer les messages que les scènes statiques, mais l'utilisation correcte de deux types de scènes en fonction de la phrase permet de bien communiquer le message. En effet, les phrases qui contiennent des verbes d'état seront mieux représentées par des scènes statiques, tandis que des animations 3D permettent de mieux représenter des phrases qui contiennent des verbes d'action. De plus, l'analyse de l'influence de l'environnement nous a permis de constater qu'il n'offre pas d'amélioration dans la communication du message.

Ces résultats nous ont permis de constater que le système est capable de générer de façon automatique des animations 3D qui transmettent le sens d'une phrase simple ce qui permet de simplifier le processus de production traditionnelle des animations 3D.

ABSTRACT

The traditional production of 3D animations for a video game or an animated film is a cumbersome process. Animators need several years of practice and excellent skills using Digital Content Creation (DCC) software to successfully create 3D animations. This is due to the complexity of the software and the complexity of the task. Creating a realistic walk cycle in a complex scene requires many low-level details for achieving a high level of realism. This thesis proposes a high-level view in the automatic creation of 3D animations to simplify the overall process of animation production.

To address this problem, the overall objective of the research was to develop a software prototype able to automatically generate 3D animations that represent the meaning of a simple sentence. This project was an integral part of the project GITAN in computer graphics. GITAN proposed a solution to generate 3D animations from text. The solution proposed in this paper is mainly the graphics module that generates animated 3D scene representing the input sentence. With this system, the complexity of building the animated scene is greatly reduced, since we use a textual representation to describe the animation and the various objects in the scene.

The literature review suggested that similar systems that automatically generate 3D animations from text are often related to a specific application domain such as automobile accidents, behavior or interactions of the characters. The automation of the scene generation for these systems is often based on scripting languages related to an application domain. In addition, we wanted to generate the animation using a 3D exchange format instead of directly display the animation. We believe that using a 3D exchange format allows us to better generate the 3D scene, since a good intermediate exchange format allows to define animations as building blocks and provides the tools to use them. For this reason, we used COLLADA as 3D format to represent our animations. From these observations, we formulated three research hypotheses. The first one assumed that it was possible to create a formalism able to describe an animated scene from a simple sentence. The formalism allows us to make an animated description of the scene using nodes, constraints and keyframes. The second hypothesis assumed that it is possible to translate the script that described the scene to a COLLADA file. We proposed a software system that translates the script to a COLLADA file that contains the 3D animation. Finally, the third hypothesis assumed that the animation generated by the system communicate the original meaning of the sentence. The system must be able to communicate the message of the sentence describing the scene to the observers.

To test these hypotheses, the methodology we have adopted consists, first of all, in the

creation of the formalism for describing the 3D scene. We have proposed an XML schema for declaring nodes, animation presets, constraints and keyframes to describe the scene. Subsequently, we proposed a modular software architecture that translates the script into the COLLADA file. The system uses algorithms to correctly position the objects in the scene and to synchronize animations. Finally, we conducted a survey to validate the communication of the message contained in the 3D scenes. The result of the survey allows us to analyze the transmission of the message to the observers and the influence of the environment of the 3D scene on the message, and so, determine if it's possible to transmit the original meaning of the sentence with the 3D animation.

The results we obtained are very rewarding. We were able to describe the scenes with the proposed script language. In addition, the software system is generating well structured COLLADA files and it is capable of generating two types of scenes: static scenes and animated scenes. Finally, analysis of survey results shows that the animated scenes can better communicate messages than static scenes, but the proper use of the two types of scenes according to the phrase can effectively communicate the message. Indeed, sentences that contain state verbs will be better represented by static scenes, while 3D animations can more adequately represent sentences that contain action verbs. Furthermore, in the analysis of the influence of the environment, we found that it offers no improvement in communicating the message.

These results revealed that the system is able to automatically generate 3D animations that convey the sense of a simple sentence to simplify the production process of traditional animation.

TABLE DES MATIÈRES

REMERCIEMENTS iii

RÉSUMÉ iv

ABSTRACT vi

TABLE DES MATIÈRES viii

LISTE DES TABLEAUX x

LISTE DES FIGURES xi

LISTE DES ANNEXES xiii

INTRODUCTION 1

CHAPITRE 1 REVUE DE CONNAISSANCES ET DE LITTÉRATURE 6

1.1 Le média 3D 6

1.2 Graphe de scène 9

1.3 Animation 3D 10

1.4 Création des scènes 3D 12

1.4.1 Interfaces de programmation 12

1.4.2 Cadre d'applications 14

1.4.3 Plateformes logicielles 17

1.4.4 Langages script d'animation 17

1.5 Formats de représentation 3D 19

1.6 Systèmes de génération de scènes animées 3D 23

1.7 Animation et positionnement de la caméra virtuelle 25

1.8 Récapitulation des connaissances 26

1.9 Hypothèses de recherche et objectifs 27

1.9.1 Objectif général 28

1.9.2 Hypothèses 28

1.9.3 Objectifs spécifiques 28

CHAPITRE 2 MÉTHODOLOGIE 29
2.1 Déterminer les entrées du système 29
2.1.1 Modèles et animations 3D utilisés dans le projet 29
2.1.2 Détermination du formalisme décrivant l'animation 3D 34
2.1.3 Création des fichiers d'entrée du système 40
2.2 Analyse du format de la représentation 3D des animations 41
2.2.1 Comparaison entre COLLADA et X3D 42
2.3 Développement d'un système de création automatique des animations 3D COLLADA 43
2.4 Méthodologie de la validation des résultats 57
2.4.1 Le sondage 58
2.4.2 Les scénarios 59
2.4.3 Les types de scénarios 60

CHAPITRE 3 RÉSULTATS 66
3.1 Performance du système 66
3.2 Fichiers COLLADA générés 66
3.2.1 Scènes statiques 66
3.2.2 Scènes animées 68
3.3 Résultat attendu 71
3.4 Résultats du sondage 73
3.5 Analyse des résultats 74

CHAPITRE 4 DISCUSSION 78
4.1 Sondage 78
4.2 Automatisation et l'aspect visuel de la scène 3D 78
4.3 Fichier de sortie COLLADA 79
4.4 Performance du système 80
4.5 Limites du système 80
4.6 Autres systèmes de génération des scènes 3D 81

CHAPITRE 5 CONCLUSION 83

RÉFÉRENCES 86

ANNEXES 92

LISTE DES TABLEAUX

Tableau 3.1 Résultats du sondage pour les deux phrases demandées 74
Tableau 3.2 Description de l'état des scènes statiques et animées 74
Tableau 3.3 Communication du message avec des scènes statiques et animées 75
Tableau 3.4 Influence de l'environnement sur des scènes animées 76
Tableau 3.5 Influence de l'environnement sur des scènes statiques 76

LISTE DES FIGURES

Figure 0.1 Architecture de GITAN 3
Figure 0.2 Architecture du module d'infographie 4
Figure 1.1 Niveaux de modélisation 7
Figure 1.2 Objet et géométrie 3D 7
Figure 1.3 Scène 3D très réaliste 8
Figure 1.4 Graphe de scene et représentation 3D 9
Figure 1.5 Animation par images clés 10
Figure 2.1 Modèle de la ville 30
Figure 2.2 Modèles 3D dans la ville 30
Figure 2.3 Modèles 3D des maisons 31
Figure 2.4 Modèles 3D de l'intérieur 31
Figure 2.5 Modèles 3D des véhicules 31
Figure 2.6 Modèles 3D des personnages 32
Figure 2.7 Hiérarchie des modèles 3D 32
Figure 2.8 Modèles 3D du plongeoir 33
Figure 2.9 Hiérarchie du plongeoir 33
Figure 2.10 Séquence d'images du cycle de marche des personnages 34
Figure 2.11 Formalisme de représentation des animations 3D 35
Figure 2.12 Séquence d'une animation 39
Figure 2.13 Collada vs X3D 43
Figure 2.14 Architecture du Système 44
Figure 2.15 Librairies du fichier COLLADA 46
Figure 2.16 Contenu du nœud arbre dans le fichier COLLADA 46
Figure 2.17 Reférence vers la géométrie de l'arbre 47
Figure 2.18 Sommets de la géométrie arbre 47
Figure 2.19 Points de référence sur le modèle 49
Figure 2.20 Nœuds de la ville 51
Figure 2.21 Contraintes sur les objets de la scène 52
Figure 2.22 Collision entre objets 54
Figure 2.23 Librairie d'animations 55
Figure 2.24 Exemple de scène statique 59
Figure 2.25 Scène 1 60
Figure 2.26 Scène 2 61

Figure 2.27 Scène 3 61
Figure 2.28 Scène 4 62
Figure 2.29 Scène 5 62
Figure 2.30 Scène 6 63
Figure 2.31 Scène 7 63
Figure 2.32 Scène 8 64
Figure 2.33 Scène 9 64
Figure 2.34 Scène 10 65
Figure 3.1 Exemple de scène Statique 67
Figure 3.2 Contenu du fichier COLLADA 68
Figure 3.3 Geométries de la scène 69
Figure 3.4 Animations générées 71

LISTE DES ANNEXES

Annexe A Contraintes de position 92
Annexe B Réponses au sondage 93
Annexe C Compilation des réponses 125

INTRODUCTION

Être capable de communiquer le sens d'un texte avec une séquence d'images animées est un défi de grande ampleur. En effet, le développement des technologies multimédia nous permet la création d'outils qui pourraient améliorer la communication visuelle. On dit souvent qu'une image vaut mille mots, donc une séquence d'images animées pourrait être assez puissante pour communiquer n'importe quelle idée à n'importe quel observateur.

Ce projet de recherche s'insère dans le cadre de la recherche faite par l'équipe GITAN, en partenariat avec la compagnie UNIMA, dirigée par les professeurs B. OZELL et M. GAGNON. Le projet GITAN propose de faire un système qui puisse permettre à la machine de comprendre du texte pour qu'elle puisse créer une animation 3D qui soit compréhensible par n'importe quel observateur, quelle que soit sa langue. Un système comme celui-ci pourrait toucher plusieurs domaines. Par exemple, dans l'apprentissage des langues, regarder une séquence animée d'une phrase qui n'est pas dans notre langue pourrait énormément faciliter la compréhension de celle-ci et accélérer son apprentissage. On pourrait aussi l'utiliser dans la scénarisation, puisqu'il serait beaucoup plus facile de prévisualiser une scène avec une simple description textuelle, elle serait donc fort utile dans l'industrie du cinéma par exemple. Les applications possibles de ce système logiciel auraient beaucoup d'importance pour la recherche et l'industrie également.

Au cours des dernières années, plusieurs d'autres projets ont essayé de développer des systèmes semblables, mais aucun d'entre eux n'a été capable de bâtir un système assez complet qui puisse gérer les ambiguïtés et complexité de la langue. La plupart d'entre eux se sont limités à un domaine d'application en particulier comme les accidents automobiles (Akerberg *et al.*, 2003) ou se sont concentrés sur certaines applications fonctionnelles comme pour l'animation des expressions faciales (Cassell *et al.*, 2001). Le projet GITAN tente de créer un système robuste et complet capable de représenter un texte par une animation 3D de façon à ce que le sens du texte soit bien transmis par l'animation et compris par n'importe quel observateur.

Pour ce faire, nous avons besoin d'un engin capable de générer des animations 3D en utilisant les concepts et les événements identifiés dans la phrase. La création traditionnelle des animations 3D dans l'industrie des jeux vidéo ou des films d'animation utilise une procédure très ardue et longue. Les logiciels de création de contenu numérique utilisés pour la création des animations 3D (ex. Maya, 3ds Max) sont très complexes et il faut des années d'expérience pour faire des animations réalistes, puisqu'il y a plusieurs détails de bas niveau qu'il faut gérer pour réussir un haut niveau de réalisme. Donc, l'engin 3D qui permettra la création de

l'animation 3D pour notre projet doit faire abstraction des certains détails afin de simplifier la création des animations 3D traditionnelle. Cette abstraction de l'animation devra faciliter leur création, mais elle doit garder un niveau de réalisme suffisant pour que le sens du message soit transmis correctement. De plus, la création doit se faire de façon automatique ce qui n'est pas le cas dans la création d'animations 3D traditionnelle.

Le projet de recherche présenté dans ce mémoire propose une solution afin de permettre la création de cet engin d'animations 3D. Premièrement, nous ferons une mise en contexte pour situer mon projet dans le cadre de GITAN et nous présenterons la problématique du projet. Par la suite, nous ferons une revue des connaissances théoriques et de la littérature pour bien comprendre et bien identifier des éléments importants qu'il faut savoir et qui nous ont aidés à développer notre système. Le chapitre deux présentera les hypothèses et les objectifs principaux sur lesquels se base mon projet. Ensuite, le troisième chapitre montrera la méthodologie utilisée tout au cours du développement et les différentes étapes. Finalement, les deux derniers chapitres présenteront les résultats obtenus et nous ferons la discussion sur les éléments importants que nous avons identifiés d'après ces résultats.

Le projet de recherche présenté dans ce mémoire est un module du projet GITAN. L'objectif du projet proposé est de permettre de valider le processus de production automatique des animations 3D. Donc, l'approche expérimentale consiste à créer, à partir d'une seule phrase simple non ambiguë, une animation 3D qui représente le sens de la phrase. Un texte au complet présente des ambiguïtés et d'autres difficultés qu'il faut gérer. Mon projet de recherche se limitera à une seule phrase assez simple qui nous permettra de valider notre système de génération d'animations 3D. Pour ce faire, nous créerons un système pour traiter les concepts de la phrase et nous générerons un fichier COLLADA pour représenter l'animation 3D. COLLADA est un format de fichier d'échange pour les applications 3D interactives.

Mise en contexte

L'engin d'animation 3D que nous avons développé fait partie intégrale du projet GITAN. Ce dernier est développé par deux sous domaines du génie logiciel : le traitement automatique des langues naturelles (TAL) et l'infographie. L'engin 3D est le dernier module qui permet la création de l'animation 3D qui représente le sens de la phrase d'entrée.

La figure 0.1 montre l'architecture générale en pipeline de tout le système GITAN. À gauche, nous pouvons voir le module de TAL où la phrase est analysée et traitée sur les différents niveaux syntaxique et sémantique. Ce traitement nous permet d'identifier les objets, les actions ou toute autre information qui pourraient être extraits de la phrase qui sera nécessaire pour le module graphique.

La partie de droite de la figure montre le module d'infographie où nous traiterons les

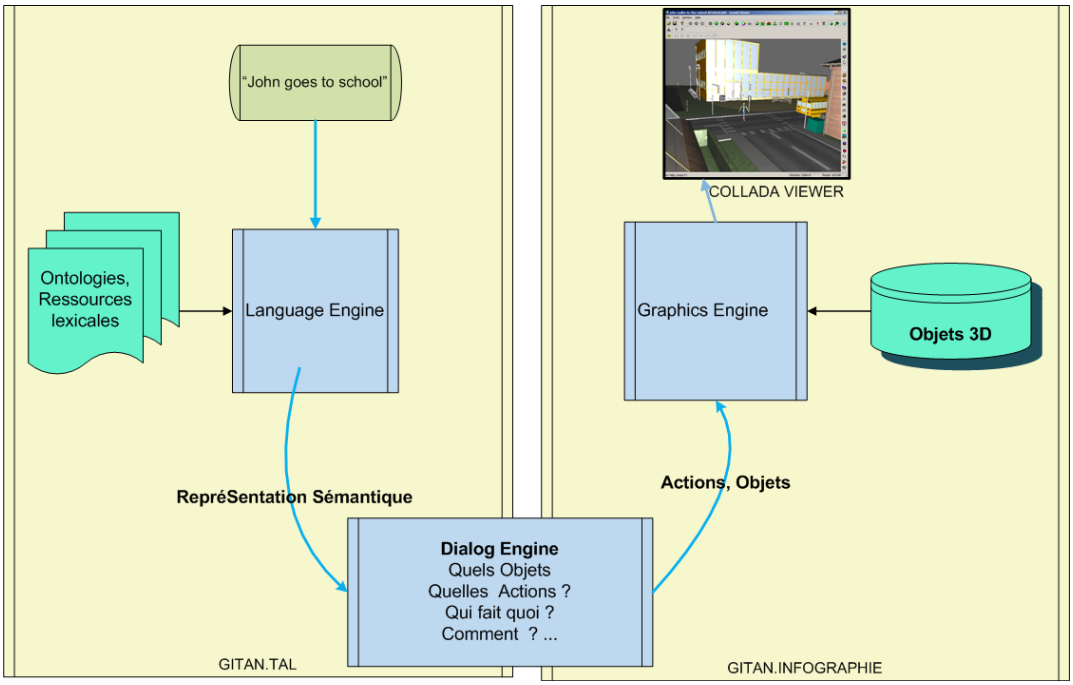

Figure 0.1 Architecture de GITAN

objets et les actions afin de créer l'animation 3D avec l'engin d'animation 3D. La figure 0.2 montre le module graphique et les différents sous modules nécessaires pour la création de l'animation 3D.

Le premier module, incorporation des objets, nous permet le placement des objets dans la scène. Son rôle principal est de déterminer le positionnement initial des objets et de traiter les collisions qui pourraient exister.

Le deuxième module, incorporation des actions, nous permet de traiter les actions dans la scène. Le rôle de ce module est d'implémenter les animations en utilisant le concept d'image clé. De cette façon, il est possible de déterminer la position des objets en fonction du temps.

Le troisième module joue le rôle de gestionnaire des données. Il permet la manipulation des objets 3D et des animations prédéfinies à partir d'une base de données ou sur le web en utilisant des ontologies.

Finalement, le dernier module, création des animations, est le moteur d'animation 3D. Toute l'information sur la scène qui sera déterminée par les autres modules sera utilisée par celui-ci afin de créer l'animation 3D de façon automatique. Lorsque nous sommes rendus à ce module, il n'y a plus des ambiguïtés, les concepts sont clairs et les actions définies. Ce module doit permettre de créer les animations 3D qui soient représentatives afin de transmettre le message de phrase initiale. Le présent projet de recherche concerne le travail réalisé dans le développement de ce module du système.

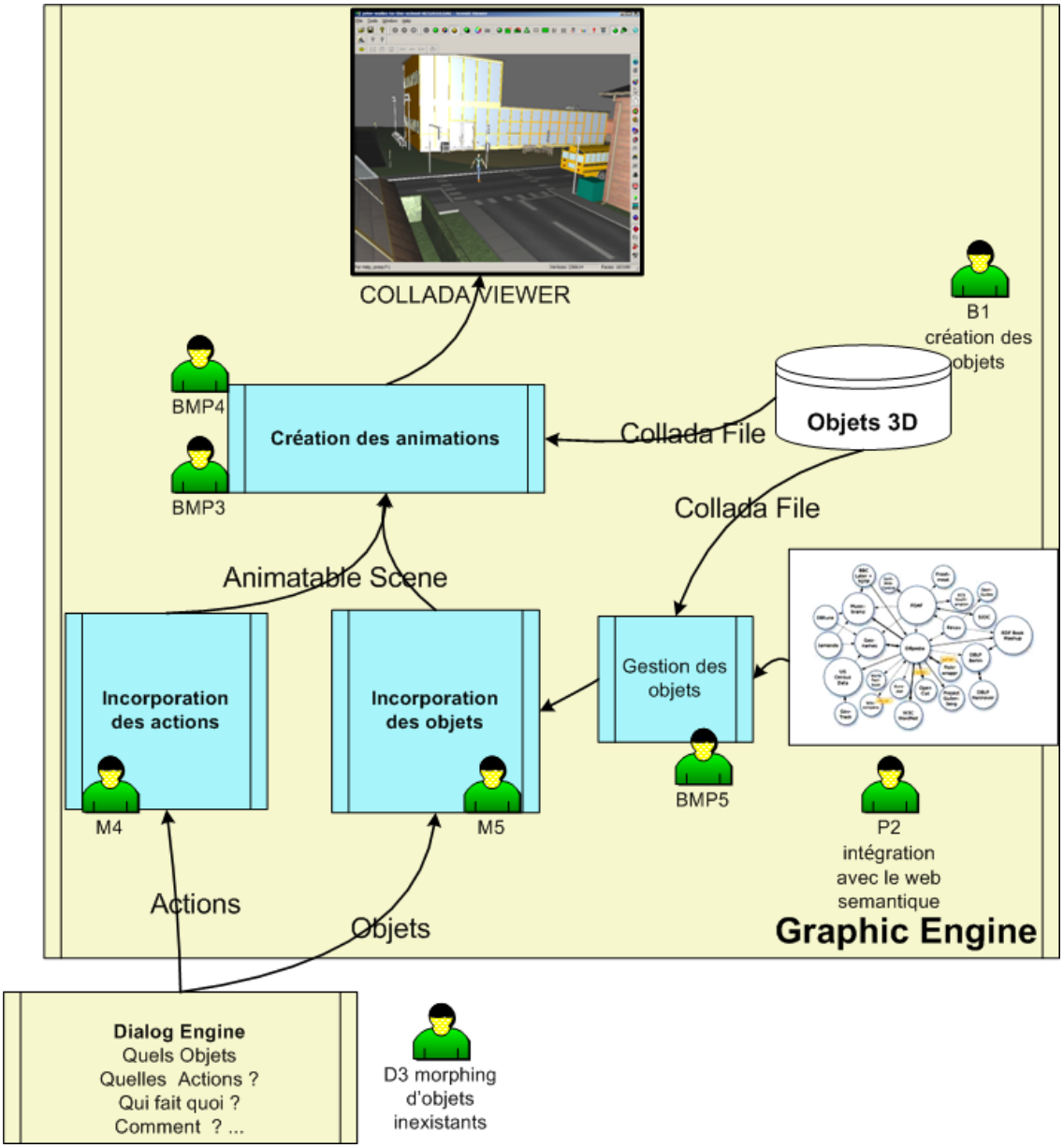

Figure 0.2 Architecture du module d'infographie

Problématique

Nous avons vu le pipeline de tout le système GITAN pour passer d'une simple phrase textuelle vers l'animation 3D qui la représente. À travers chaque module où la phrase est traitée, on extrait de l'information de la phrase comme les objets, les actions ou événements dans la scène. Toute cette information doit être traitée et transformée en une séquence d'images animées pour reconstituer le sens de la phrase. Dans la création traditionnelle des animations 3D, il faudrait qu'une personne dessine les objets 3D nécessaires en utilisant un logiciel de création numérique comme 3ds Max et ensuite les place dans la scène pour créer une animation 3D avec le même ou un autre logiciel comme Blender. Il faut des années d'expérience pour maîtriser ces logiciels et des personnes ressources avec des compétences artistiques dans la modélisation et l'animation afin de produire une animation 3D très réaliste comme nous le constatons dans les jeux de vidéo ou les films d'animation. Par contre, l'animation que nous désirons produire n'a pas besoin de ce niveau de réalisme. Nous avons besoin de faire abstraction d'une animation 3D pour transmettre un message de la façon la plus simple possible. Donc, le réalisme n'est pas une priorité tant que le message soit transmis correctement.

Le système dont nous avons besoin doit gérer l'information d'entrée qui contient la description de l'animation 3D et à partir de cette description générer automatiquement une animation 3D simple. Cette animation doit permettre de communiquer le message à n'importe quel observateur. Donc, la question à la base de cette recherche est la suivante : est-il possible de générer de façon automatique une animation 3D qui reflète le sens d'une phrase de départ. L'objectif général du projet est donc de proposer un système capable de traduire l'information extraite de la phrase en animation 3D de façon automatique.

CHAPITRE 1

REVUE DE CONNAISSANCES ET DE LITTÉRATURE

La revue de littérature qui suit permet de donner le cadre théorique de cette recherche en la situant par rapport aux travaux antérieurs connexes, similaires ou précurseurs. Nous présenterons premièrement les notions des données 3D et leur représentation. Par la suite, nous verrons les types de représentation d'une scène animée et quelques systèmes de génération des scènes 3D. De plus, nous montrerons les principaux travaux de conversion du texte en animation 3D. Finalement, la dernière section présentera les objectifs et hypothèses de cette recherche.

1.1 Le média 3D

Dans les dernières années, nous avons été témoins de l'évolution des technologies multimédia soit pour la bande passante Internet, les techniques de compression, la visualisation et l'affichage multimédia dans un environnement professionnel comme personnel. Le média 3D est la représentation digitale des objets physiques ou virtuels qui peuvent être traités par des applications informatiques. Le 3D est devenu rapidement un nouveau type de média qui apporte une autre dimension au monde multimédia. Ces représentations 3D peuvent être définies directement à partir du monde virtuel avec un système de modélisation (ex. 3ds Max) ou obtenues par le balayage de surfaces de l'objet physique. Le média 3D est relativement récent dans le monde multimédia, par contre, dans la dernière décennie, l'infographie est dans une phase très mature où les problèmes fondamentaux sur la modélisation, visualisation et la diffusion en continu des formes 3D, soit statiques ou dynamiques, ont été bien compris et résolus. La figure 1.1 montre les phases de la modélisation traditionnelle du média 3D. À partir d'un monde virtuel ou physique, on utilise un modèle mathématique pour modéliser l'objet, ensuite nous pouvons représenter le modèle par sa géométrie (triangles ou polygones) et finalement l'implémentation dans l'ordinateur qui est la forme de stockage ou structure de données de l'objet 3D.

La géométrie est la dimension fondamentale de tout média 3D. Elle contient la forme physique de l'objet et constitue le squelette de la scène 3D. Par exemple, dans la figure suivante 1.2, nous avons une représentation d'une géométrie à gauche qui est un ensemble de polygones qui ont la forme d'un lapin. La figure à droite montre la géométrie du lapin affiché dans une scène 3D avec une texture blanche et de l'ombre qui permettent de donner

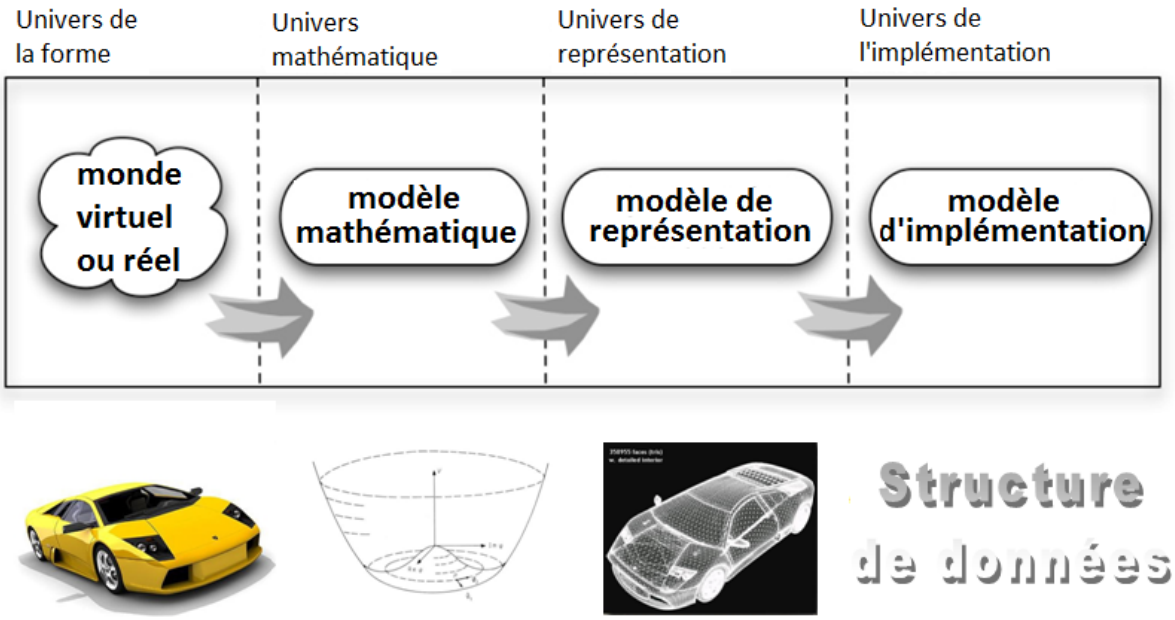

Figure 1.1 Niveaux de modélisation

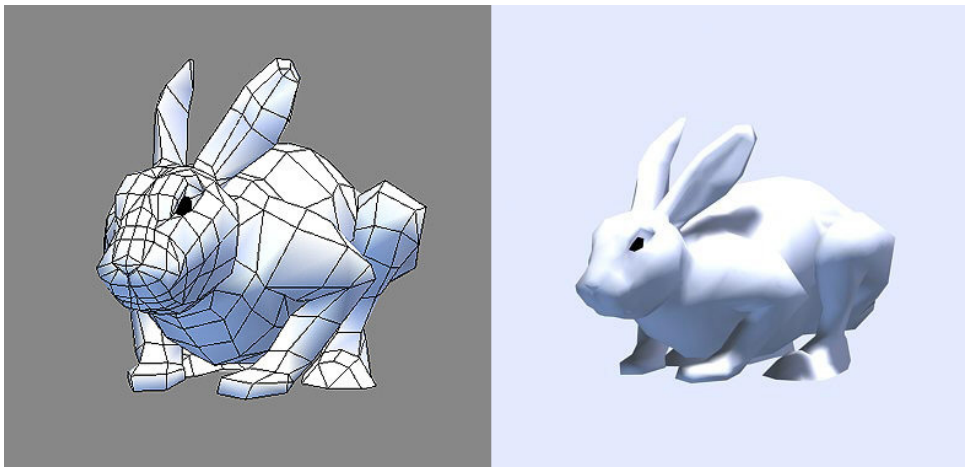

Figure 1.2 Objet et géométrie 3D

une apparence plus réaliste.

Si nous parlons de média 3D, nous devons introduire le concept de scène 3D qui représente l'espace utilisé pour le rendu visuel et ainsi permettre toute interaction et manipulation de celui-ci. C'est un monde virtuel où l'utilisateur peut visualiser une partie ou la totalité du média 3D. En général, les scènes 3D sont définies par leur géométrie et leur apparence, mais d'autres caractéristiques peuvent enrichir la scène comme des lumières, des ombres ou des sons. Dans une scène 3D, des transformations sont appliquées sur des géométries pour les déplacer ou les déformer dans le monde virtuel, puisque la géométrie contient seulement de l'information relative en fonction de ses axes de référence. La figure 1.3 montre une scène 3D qui contient des géométries, de la lumière, des ombres et un environnement très réaliste.

Figure 1.3 Scène 3D très réaliste

1.2 Graphe de scène

Les géométries sont affichées dans la scène 3D, mais pour ce faire nous avons besoin d'une organisation des formes, des transformations et tout autre élément de la scène pour nous permettre un rendu efficace et optimisé. En effet, les graphes de scène nous permettent une organisation hiérarchique des formes et des groupes de formes qui définissent le contenu d'une scène 3D. Ce sont des arbres acycliques qui représentent les éléments d'une scène. Nous pouvons diviser les nœuds en trois catégories : la racine, les nœuds intérieurs qui permettent de regrouper d'autres nœuds et les feuilles qui sont localisées à la fin de chaque branche. La racine est le premier nœud, tous les autres nœuds sont connectés à elle directement ou indirectement. Les nœuds intérieurs ont différentes propriétés comme pour les transformations 3D. Finalement, les feuilles contiennent les géométries. Dans la figure 1.4, nous avons à droite le graphe de scène représentant deux arbres à côté d'une maison. À gauche, nous avons le rendu dans la scène 3D de celui-ci.

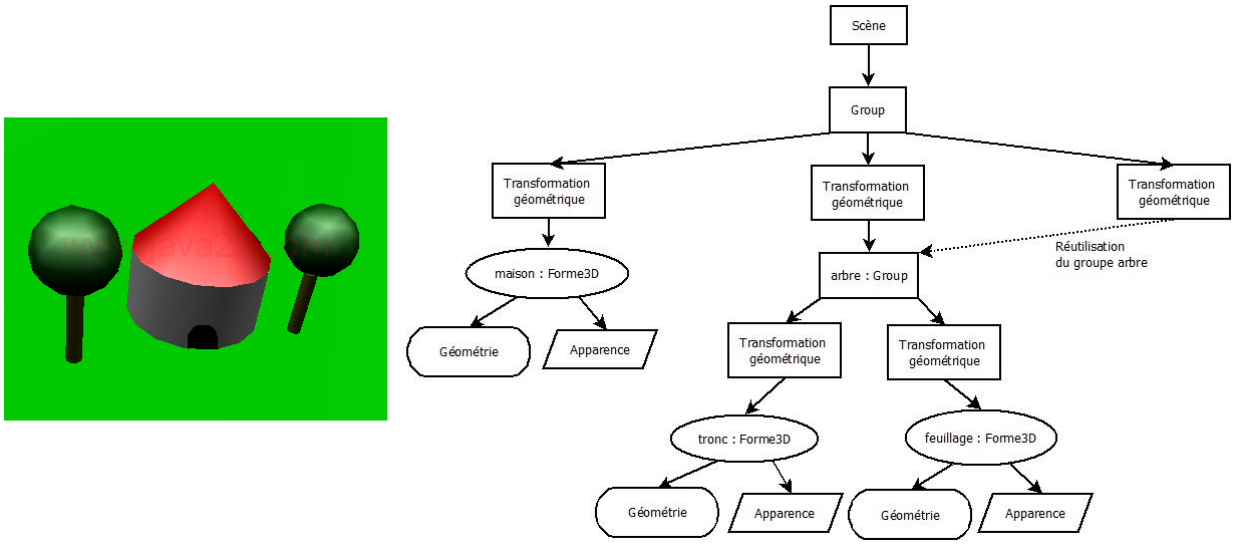

Figure 1.4 Graphe de scene et représentation 3D

Le graphe de scène nous permet de réutiliser des éléments de notre scène qui se répètent afin d'optimiser le rendu. Dans notre exemple, l'arbre est affiché deux fois, mais nous utilisons le même nœud pour afficher les deux arbres. Les nœuds de transformation, quant à eux, nous permettent de déplacer les objets dans la scène ou les déformer en utilisant des déplacements, des rotations ou le redimensionnement sur les trois axes. Donc, la notion de graphe de scène est très important pour notre projet, puisque la scène qui sera générée devra se baser sur des graphes de scènes pour construire l'animation (Reiners, 2002).

1.3 Animation 3D

Des images transmettent beaucoup d'information, car le système visuel humain est un processeur sophistiqué d'information. Donc, des images animées ont le potentiel de transmettre encore plus d'information, c'est un des points importants que cette recherche essayera de valider. En infographie, on dit que lorsque la géométrie ou les éléments d'une scène 3D sont susceptibles de varier dans le temps, on parle d'une scène 3D animée ou d'une animation 3D (Bilasco, 2007).

L'animation par ordinateur a été présente tout au long de l'existence de l'infographie. Il y a principalement trois méthodes par lesquels l'animation 3D peut être générée. La première méthode est l'animation par image clé où l'animateur spécifie les poses importantes pour certains cadres et l'ordinateur calcule les images entre deux images clés avec une technique d'interpolation (Izani *et al.*, 2003). La figure 1.5 montre un exemple très simple de trois images clés qui sont interpolées pour créer ce type d'animation. La deuxième méthode d'animation est la simulation physique, mais à cause de la complexité des calculs cette méthode n'est pas idéale pour l'animation des personnages. Finalement, l'animation par détection de mouvement est une méthode plus réaliste qui est utilisée avec plus de succès pour l'animation des personnages. Avec cette méthode, des capteurs sont placés sur une personne vivante et les données produites par le mouvement de la personne sont appliquées sur le personnage 3D (Horprasert *et al.*, 1998). Également, on peut faire la reconnaissance et simulation des expressions faciales en utilisant des capteurs sur les muscles et en analysant la relation entre le muscle et la surface de déformation du visage (Choe *et al.*, 2001).

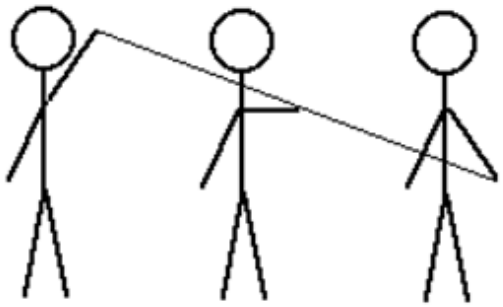

Figure 1.5 Animation par images clés

Magnenat-Thalmann (Magnenat-Thalmann et Thalmann, 1991) propose trois différentes approches pour la classification des scènes animées par ordinateur en fonction de la nature des données soit géométrique, physique ou comportementale. Les trois approches sont :

– L'animation dépendant de l'animateur.

– L'animation par simulation dynamique.
– L'animation comportementale.

La première approche correspond aux méthodes invoquées par l'animateur. On retrouve des méthodes comme la cinématique inverse utilisée dans l'animation des squelettes et les méthodes d'animation par image clé et le morphage. Ces méthodes sont normalement régies par des données géométriques et les animations ne se basent pas sur des faits physiques comme la gravité ou l'inertie. La cinématique inverse permet dans les systèmes articulés le contrôle de la posture finale et le système calcule les angles correspondants des joints de toute l'articulation (Kim *et al.*, 2002). Les systèmes articulés basés sur l'animation par images clés manipulent des angles et permet de spécifier les angles à différents moments clés dans le temps et par la suite, l'ordinateur est capable de trouver les positions à tout moment de l'animation par interpolation. Le morphage est la technique qui permet la transformation d'un modèle 3D source vers un autre modèle 3D en changeant le positionnement des sommets du modèle source en fonction du temps (Kaneko et Okada, 2008). Cette approche est très intéressante aux fins de notre recherche, puisqu'elle permet de créer des animations de façon simple et efficace.

La deuxième approche permet la création des animations réalistes, puisque le mouvement dans l'animation est produit par la simulation des lois physiques. L'animateur doit fournir des données physiques qui correspondent au mouvement simulé. Les lois physiques simulées sont principalement celles de la mécanique. La trajectoire et la vitesse sont obtenues par la résolution d'équations en utilisant des forces, des couples, des contraintes et les propriétés de la masse des objets. On utilise différentes formulations comme celle de Newton-Euleur, Lagrange, Gibbs-Appel ou d'Alembert pour simuler des mouvements plus réalistes. Les méthodes basées sur l'ajustement de paramètres sont les plus populaires dans cette approche et correspondent aux méthodes sans contraintes (Kiss, 2002).

La troisième approche est l'animation comportementale qui correspond à la simulation du comportement des personnages soit pour le calcul simple de leurs trajectoires ou pour l'interaction émotionnelle entre personnages. Dans l'animation comportementale idéale, c'est impossible de jouer exactement la même scène deux fois puisque, par exemple, une personne marchera ou agira plus ou moins toujours de la même façon, parce que cela dépend de l'état émotionnel de la personne. Si elle est triste, fatiguée ou énervée, cela peut influencer sa façon de marcher par exemple. Ce type d'animation est un défi dans le domaine de l'animation 3D. Dans cette approche, on retrouve différentes méthodes d'animation de façon à simuler le comportement des animaux ou des humains de la façon la plus réaliste (Thalmann, 1996). Par exemple, on essaie de simuler la trajectoire des groupes comme le troupeau d'oiseaux, des troupeaux d'animaux, bancs de poissons, etc. Ce type d'animation a été étudié par Reynolds

afin de simuler la trajectoire des troupeaux des oiseaux où la trajectoire est le résultat de l'interaction entre le comportement de tous les oiseaux (Reynolds, 1987).

1.4 Création des scènes 3D

Il existe plusieurs technologies qui nous permettent la création des scènes 3D en plus d'utiliser des outils de conception et de modélisation 3D comme 3ds Max ou Maya. Il existe d'autres outils pour la création de scènes 3D qui sont utilisés dans plusieurs domaines comme la visualisation scientifique, la réalité virtuelle, la simulation, les jeux de vidéos, etc. Nous pouvons distinguer quatre catégories pour la création des scènes animées 3D : interfaces de programmation graphiques, cadre d'applications (*frameworks*), plateformes logicielles et les langages script d'animation. Il est important de bien identifier les différents outils et solutions disponibles pour la création de scènes de notre système. Nous présenterons une brève description des outils disponibles dans ces quatre catégories.

1.4.1 Interfaces de programmation

Une interface de programmation est considérée comme une ou plusieurs bibliothèques définies qui fournissent un accès à du matériel ou des ressources logiciels. Une interface de programmation fait abstraction des fonctionnalités de ressources de bas niveau et fournit un ensemble de méthodes bien documentées qui fonctionnent comme une enveloppe des fonctionnalités. Elles n'offrent pas la commodité des plateformes ou des cadres d'applications, mais elles fournissent une meilleure approche pour manipuler de nouvelles technologies d'infographie dans le génie logiciel. Parmi les interfaces graphiques les plus utilisées, nous pouvons nommer FreeVR, Java3D, OGRE, OpenSceneGraph, OpenSG, OpenVRML, X3D Tool Kit et Xj3D.

FreeVR

C'est une librairie multiplateforme qui permet d'accéder et d'intégrer une grande variété de systèmes de réalité virtuelle et ses configurations. Elle compile sous UNIX, OS X et Cygwin. Elle est recommandée dans l'utilisation de systèmes hautement immersifs comme la CAVE. L'affichage graphique est géré par OpenGL, OpenSceneGraph ou OpenSG. Elle permet d'associer les périphériques physiques à ceux qui sont virtuels pour interagir avec le monde virtuel. Elle facilite la programmation et l'accès de toute sorte de mécanismes externe. Par contre, elle n'a pas de support pour les composantes d'audio ou de réseau (freevr, 2010).

Java3D

Java3D permet de créer des applications graphiques 3D et applets. Elle est un système basé sur des graphes de scène pour la manipulation du 3D. De cette façon, elle permet une approche hiérarchique pour décrire les objets et les relations entre eux. Le graphe de scène peut contenir des objets graphiques, des lumières, des sons, des effets d'environnement et des objets interactifs qui peuvent aussi modifier l'état d'autres objets dans le graphe de scène. Java3D peut utiliser OpenGL ou Microsoft DirectX pour l'affichage des objets. Cette interface est multiplateforme et peut être exécutée sous Apple OS X, Linux, Sun Solaris et Windows comme application ou sur le Web. Elle permet l'utilisation des modèles VRML 2, mais elle n'offre pas de support pour X3D (Sowizral *et al.*, 1997).

OGRE

Ogre est une interface de programmation graphique en C++ qui se concentre sur la modularité et la fonctionnalité. Elle est multiplateforme et peut être exécutée sur Linux, Apple OS X et Windows. Son objectif principal est de faire une interface intuitive et facile à utiliser pour les développeurs des applications graphiques 3D. Cette interface de programmation est spécifiquement architecturée pour être utilisée avec des projets qui manipulent du contenu 3D, puisqu'elle supporte les principales techniques d'affichage avec des graphes de scène : le niveau de détail, le « mipmapping », l'affichage sur des textures, affichage des ombres, systèmes de particules, etc. Ogre est conçue pour être extensible à travers des plugiciels. OGRE est principalement un engin graphique pour des applications 3D. Cette interface n'a pas de support pour les composantes audio ou de réseaux. Elle ne supporte pas non plus des modèles VRML ou X3D (Junker, 2006).

OpenSceneGraph et OpenSG

OpenSG est une interface de programmation semblable à OpenSceneGraph. Les deux sont basées sur OpenGL, elles sont écrites en C++ et elles sont multiplateformes. Les deux supportent le multithreading, mais OpenSG est explicitement équipée pour gérer les grappes graphiques. Les deux offrent une gestion de haute performance du graphe de scène et finalement les deux supportent un large éventail de formats de fichiers 3D ainsi que VRML2. OpenSG est facilement extensible, donc on peut ajouter des classes sans trop modifier le code. OpenSG a besoin également d'autres ressources pour gérer le réseautage et le son (Dirk *et al.*, 2002; Paul, 2007).

OpenVRML

C'est une interface de programmation pour les standards VRML et X3D. Écrit en C++, elle est multiplateforme et peut être exécutée sur les systèmes où la librairie OpenGL est supportée. Le GIMP Tool Kit Plus est disponible pour les interfaces usager et des plugiciels pour le navigateur web Mozilla peuvent être utilisés. Elle est construite en fonction des graphes de scènes. VRML et X3D offrent un moyen basé sur un script pour décrire le rendu et le contrôle de scènes 3D. Cette interface possède du support pour l'audio, des environnements dynamiques et des avatars. Par contre, elle ne possède pas de support pour le réseautage. De plus, en dépit de son implémentation de X3D, elle ne supporte pas l'encodage XML de X3D. Finalement, il n'y a pas beaucoup de documentation et elle a besoin de plus d'expansion (Chris et Braden, 2010).

The X3D Tool Kit

C'est un projet abandonné qui a été repris. De 2002 à 2004, l'auteur a développé une implémentation du X3D standard pour le support de scènes 3D statiques. En 2006, le projet a été repris par un groupe de développeurs qui avait mis l'accent sur le comportement de X3D tel que spécifié par les normes. Cette interface est écrite en C++, elle est basée sur les graphes de scènes et elle utilise des mécanismes pour gérer et traiter de l'information 3D, mais elle utilise OpenGL pour l'affichage. Elle est multiplateforme et elle peut être exécutée là où OpenGL est supporté. Le problème avec cette interface de programmation est qu'elle a besoin d'autres ressources pour le réseautage et toute la documentation a été gelée depuis 2004 (Web3D, 2010).

1.4.2 Cadre d'applications

Un cadre d'applications est un groupe d'interfaces de programmation qui font abstraction dans multiples domaines du logiciel comme des ressources matérielles et utilise le support également de divers outils pour la création des applications (ex. des éditeurs de contenu, des utilitaires d'administration). Un cadre d'applications robuste peut couvrir plusieurs domaines pour la création des scènes 3D et surtout pour des environnements virtuels 3D comme pour l'affichage, le contrôle de ressources de grappes, la mise en réseau, le son, etc. Ceci permet au développeur logiciel de travailler sur une interface et éviter ainsi la documentation et la maintenance. Dans le monde de l'infographie, les cadres d'applications sont plus souvent utilisés pour la création des jeux vidéo et on les nomme plus souvent « moteur de jeux ». Il existe un grand nombre de moteurs de jeux qui sont de code source libre et d'autres commerciaux. L'utilisation d'un moteur de jeu dans notre projet pourrait être intéressante

pour la création de la scène et de l'animation 3D. Il existe une grande quantité des moteurs de jeu actuellement, mais nous ferons une brève description de quelques-uns.

Crystal Space

À partir de 1997, Crystal Space a évolué en une collection modulaire de librairies C++ pour permettre la création des applications 3D avec du support pour le réseau, le son entre autres. C'est un moteur gratuit, il peut être exécuté sur Linux, Apple OS X et Windows. Il est extensible à travers son système de plugiciels pour l'ajout des interfaces de programmation, donc il permet au développeur d'inclure ou exclure les modules qui l'intéressent. Il possède une interface de haut niveau, CELstart, qui rend possible de programmer une application au complet en Python ou XML. Des liaisons pour le langage Java et Perl sont également disponibles. Finalement, une extension du logiciel de modélisation Blender3D permet l'exportation directe vers Crystal Space. Le point faible de Cristal Space est le peu de support des standards 3D et cela complique le partage des modèles 3D avec d'autres systèmes (Space, 2010).

Delta3D

Il a été créé et maintenu par « Modeling Virtual Environnement and Simulation » (MOVES) en Californie. Ce moteur de jeux vise des applications de réalité virtuelle dans le domaine de la défense ainsi que la modélisation et la simulation. Il est approprié pour une grande variété d'utilisations y compris la formation, l'éducation, la visualisation et le divertissement. Delta3D s'appuie sur large éventail d'interfaces de programmation et d'outils de support. Il est écrit en C++ et il est compatible avec Linux, Windows et Apple OS X. Il offre une interface de haut niveau tout en permettant l'accès de bas niveau si on le souhaite. Il offre du support pour l'affichage 3D, la mise en réseau, le son et d'autres utilitaires de support comme un éditeur de cartes 3D, un éditeur des effets de particules et un visionneur de modèles 3D. D'autres points forts comprennent des interfaces de programmation pour la création d'avatars, la fixation de la météo, la gestion de terrains et pour la liaison avec des scripts Python. L'interface pour le rendu est basée sur OpenGL et OpenSceneGraph intégrant ainsi le standard VRML2. Par contre, il ne supporte pas le standard X3D (Darken *et al.*, 2005).

Quake III Engine

Il a été initialement créé pour le jeu vidéo Quake III Arena par Id Software en 1999. En 2005, Id Software a publié le moteur de jeu comme logiciel libre sous la licence de GNU

GPL. C'est un moteur riche en fonctionnalités, il offre du support pour le rendu graphique, le son 3D et la mise en réseau. Il est entièrement écrit en C et compatible avec Linux, Apple OS X et Windows. Il est optimisé pour les performances du rendu graphiques et la mise en réseau. Le rendu est géré par OpenGL avec l'aide de l'accélération matérielle 3D et le réseau comprend une compression intégrée pour transmettre les données plus efficacement. Il comprend l'utilisation d'une machine virtuelle « QVM » pour facilement et en toute sécurité permettre la modification du jeu par les utilisateurs finaux. Finalement, il offre un support complet pour les avatars et le clavardage entre les usagers. Les défis d'utilisation de ce moteur de jeu sont reliés au manque de soutien des standards 3D, il utilise le format 3D MD3 et BSP dont il est propriétaire pour les avatars et les cartes respectivement. Ces formats peuvent être difficiles à créer ou à manipuler et ils sont moins compatibles avec d'autres logiciels 3D. De plus, la documentation de Quake III est abondante, mais dispersée dans de nombreux endroits sur Internet. Toutefois, la publication du moteur par Id Software contient très peu de documentation (Trenholme et Smith, 2008).

Uni-Verse

Basé sur le protocole de réseau Verse, ce moteur de jeu a comme objectif de créer une plateforme Internet code ouvert multi-usager et interactif de haute qualité pour les graphiques 3D et l'audio (Uni-verse). Il est multiplateforme, donc il peut être exécuté sur Apple OS X, Linux et Windows. Il est orienté client-serveur et possède plusieurs interfaces de programmation en C et plusieurs outils de support. Le protocole Verse a été créé dans le but de partager de contenu 2D et 3D efficacement entre les applications. Le rendu graphique est assuré par OpenGL avec le soutien de OpenSG sous Microsoft Windows. Avec OpenSG, le standard VRML2 est disponible si les librairies OpenVRML sont utilisées. La gestion du son est faite par le système audio/vidéo appelé Pure Data qui doit être installé sur chaque machine client de l'application. D'autres composantes du système incluent des liaisons pour le langage Python, un outil pour le scripting d'environnement appelé « Purple » fait pour Uni-Verse, des outils pour enregistrer et charger l'état du serveur Uni-Verse et sécurité de réseau utilisant un algorithme de chiffrement cryptographique (RSA). Bien que Uni-Verse offre un environnement complet, il possède quelques inconvénients. La composante audio est compliquée à cause de l'environnement de programmation Pure Data installé sur les machines serveur. Finalement, il supporte VRML2, mais il ne supporte pas le standard X3D (Uni-Verse.org, 2010).

1.4.3 Plateformes logicielles

Une plateforme logicielle est un environnement spécialisé dans lequel une solution peut être développée, testée et exécutée. Ces plateformes reprennent tout ce que les cadres d'applications supportent et permettent de faire. L'utilisation des plateformes spécialisées au développement des scènes ou graphiques 3D dépasse le cadre de notre projet, donc nous nous limiterons à nommer deux plateformes existantes : Croquet (Smith *et al.*, 2003) et Project Wonderland (Microsystems, 2010). Ces deux plateformes permettent la création des environnements virtuels 3D, mais ce n'est pas ce que nous recherchons pour notre projet.

1.4.4 Langages script d'animation

Les animations 3D décrivent des scènes qui sont composées par entités visibles ou objets qui changent en fonction du temps. Ces entités que nous pouvons appeler « acteurs », ont un rôle dans l'animation. Dans cette analogie, l'animateur joue le rôle de « directeur », il donne également des directions aux acteurs et il assume qu'ils savent quoi faire et à quel moment. L'animateur fournit aux objets une liste d'ordres ou de messages, un temps initial et un temps final. Donc, ces objets, pendant qu'ils sont actifs, envoient et reçoivent des messages aveuglément dans leurs listes d'envoi en supposant que les objets qui les reçoivent les comprennent. Ainsi, les scripts sont des listes des messages et toutes les activités qui sont déclenchées sont des réponses aux messages et il n'y a pas de limitations sur ce qui peut faire un script (Getto et Breen, 1990). Il existe plusieurs langages script d'animation, pas autant que de moteurs de jeux, mais il y a différents langages scripts qui permettent non seulement l'animation, mais aussi le comportement, la représentation humaine, des dialogues verbaux ou non verbaux, etc. Les langages script sont très flexibles ce qui pourrait être très intéressant pour notre projet. Nous présenterons brièvement quelques-uns.

AML

Le langage script « Avatar Mark-up Language » a été développé dans le contexte du projet IST SoNG. L'objectif du projet était de faire le design et développer un cadre d'applications multimédia pour supporter des avatars 3D, le clavardage et l'animation des personnages autonomes. Premièrement, ils ont développé une interface pour définir la séquence d'animation en sélectionnant et en manipulant des animations prédéfinies. Donc, les personnages seraient contrôlés de façon similaire au rôle des assistants de ventes dans les magasins virtuels. Le projet voulait créer un système qui permettait aux usagers ou développeurs d'animer leurs avatars en utilisant des commandes non procédurales tout en essayant de ne pas limiter leur créativité en imposant des expressions faciales ou gestes prédéfinis. Donc, pour ce faire, le

projet avait trois composantes : une base de données avec les expressions faciales et les animations prédéfinies, le système permettant de fusionner plusieurs gestes ou animations avec des dialogues verbaux et le langage script pour permettre aux animateurs de spécifier tous les paramètres des animations. Donc ce langage permettait la synchronisation des expressions faciales, des gestes et de la parole, puisque l'animateur peut spécifier les temps et les durées de chacune d'entre elles. Ce langage est basé sur XML et il est facile à comprendre par les animateurs (Arafa *et al.*, 2002).

CML

Le langage script « Character Mark-up Language » permet de décrire l'animation des personnages 3D et ses comportements. Il existe plusieurs outils pour l'animation des personnages 3D ou des avatars, également, il existe plusieurs moteurs d'émotions pour générer des comportements réalistes aux personnages 3D, par contre il n'existe pas un outil pour fusionner ces éléments. CML a été créé pour faire le lien entre les moteurs d'émotions pour générer des gestes ou expressions faciales et les outils d'animation. Il permet de spécifier l'animation en décrivant l'expression et le comportement de la face et du corps, la personnalité, le rôle, l'émotion et les gestes. Il est basé sur XML et il peut être exécuté sur plusieurs plateformes. La description pour les expressions faciales est celle utilisée dans FACS (Facial Action Coding System) qui définit toutes les expressions faciales qui performe l'être humain(Ekman et Rosenberg, 2005). Le moteur d'animation et de comportement est basé sur le travail de Blumberg et Russel (Russell et Blumberg, 1999) dont l'architecture de leur système se divise en trois niveaux : la géométrie, le moteur de mouvement et le système de comportement. Donc, CML décrit les actions et les séquences dans laquelle les actions seront exécutées dans l'animation. Le script est une collection des commandes qui disent aux objets dans le monde quoi faire et comment performer ses actions (Arafa *et al.*, 2002).

STEP

STEP est un langage script pour l'animation des personnages virtuels humanoïdes 3D. Il est basé sur le standard H-Anim. Il a été créé pour les raisons suivantes : fait pour les auteurs non professionnels, besoin de combiner des opérations, permettre une spécification de haut niveau, adaptation des actions et interaction avec un environnement virtuel. STEP fait une distinction entre la spécification pour les actions externes et l'état interne des personnages virtuels en les séparant. Donc, il n'utilise pas le même langage pour leur description. De plus, STEP supporte des références VRML ce qui est utile pour des utilisateurs expérimentés. La version encodée en XML s'appelle XSTEP (Huang *et al.*, 2003).

VHML

Le « Virtual Human Markup Language » est un projet réalisé par l'université de Curtin pour permettre de spécifier l'animation des têtes humanoïdes interactives sous le format XML. Le langage est conçu pour bien gérer l'interaction entre la machine et l'humain sur plusieurs aspects dont : les expressions faciales, l'animation corporelle, interaction du dialogue, le discours, la représentation des émotions et les postures. Il est utilisé dans l'application « Mentoring System » (MentorSystems, 2009). Ce langage est divisé en plusieurs sous-systèmes pour la description de chaque aspect du personnage virtuel. Donc, VHML permet de faciliter une interaction naturelle et réaliste des têtes humanoïdes ou personnages virtuels avec l'utilisateur (School et Marriott, 2007).

Langage script UNITY

C'est un moteur de jeux qui possède un langage script très important. Un jeu de vidéo au complet peut être créé avec ce langage, puisqu'il permet la création d'une scène, l'animation des objets, les lumières et le mouvement de la caméra. Il supporte l'interaction avec l'utilisateur. Il est basé sur le principe général d'avoir un affichage en boucle constant qui évalue la position de tous les objets dans la scène et les objets sont affichés à travers la caméra (FrameForge, 2009).

1.5 Formats de représentation 3D

Les premiers outils qui ont été créés pour la conception du média 3D sont issus des domaines de la « conception assistée par ordinateur » (CAO). Ces outils se concentraient principalement sur la géométrie et l'apparence des objets modélisés. Par la suite, ces applications se sont intéressées de plus en plus à la description complète de la scène. Le comportement des objets (des animations temporelles) et l'environnement de la scène se sont ajoutés. Des outils de plus complets en plus complexes se sont développés et chaque compagnie développait son propre langage de description 3D ce qui causait des problèmes, puisque le contenu 3D pouvait seulement être utilisé ou lu par l'outil qui l'avait créé. À partir des années 90, la communauté d'infographie avait besoin d'un format standard d'échange pour des applications 3D. Il y a eu plusieurs formats qui ont été créés à partir de ce moment, plusieurs d'entre eux n'ont pas eu de succès, mais certains ont été adoptés par la communauté. Nous avons bien analysé les différents formats 3D puisque l'animation que nous voulons générer devra être encodée sous le format 3D plus approprié. Nous présenterons une brève description des principaux formats que nous avons analysés. L'analyse de ces formats sera présentée plus tard dans la méthodologie.

VRML

Le « Virtual Reality Modeling Language » (VRML) est un langage de présentation pour décrire des mondes virtuels 3D qui peuvent être diffusés via Internet par le World Wide Web. Plusieurs professionnels de la 3D ont travaillé à la création de ce langage. Pesce et Parisi qui développaient Labyrinth en 1994, un prototype d'interface 3D usager pour internet et Carey et Strauss from Silicon Graphics qui développaient un format d'échange pour les applications 3D dans leur système Open Invertor(Wernecke, 1993) ont été les premiers impliqués. Les deux équipes, Labyrinth et Open Inventor étaient un des premiers collaborateurs pour créer la base de VRML. Le but premier de ce langage est de permettre la représentation des mondes virtuels 3D incorporant des géométries 3D, des lumières, le brouillard, l'animation et même des effets sonores. Chaque monde virtuel est décrit par un ou plusieurs fichiers VRML d'extension « .wrl » et transmis à travers le Web. Pour visualiser le monde virtuel VRML, il faut un navigateur VRML qui est intégré comme plugiciel dans les navigateurs internet. Tous les aspects d'affichage des mondes virtuels, d'interaction et de réseautage peuvent être spécifiés en utilisant VRML. L'intention des designers était de faire de VRML un langage standard pour la simulation interactive 3D par le Web (Nadeau, 1999; Carson *et al.*, 1999).

U3D

Universal 3D (U3D) est un format d'échange pour des applications 3D. Ce format a été créé parce qu'il y avait peu de standards pour la conception assistée par ordinateur (CAO) et qu'il y avait une forte demande parce que l'utilisation de contenu 3D augmentait rapidement. Le format U3D est léger ce qui permet une distribution efficace à travers Internet et sur des réseaux privés en plus de faciliter le travail des applications qui l'utilisent. De plus, il permet une manipulation très simple des composantes de la scène ce qui est intéressant pour notre projet, par contre, ce format ne supporte pas des animations 3D (Geer, 2005).

XAML

eXtensible Application Markup Language (XAML) est un langage déclaratif développé par Microsoft pour les besoins du système d'exploitation Windows Vista. Ce langage, basé sur XML, permet la création des interfaces utilisateur pour les applications de Windows Vista et applications Web. Il est semblable à d'autres langages pour l'affichage sur les navigateurs Internet. L'idée est de séparer la construction de l'interface utilisateur du code sous-jacent. XAML permet la manipulation du contenu 3D. Il est possible d'écrire et modifier les applications XAML à l'aide d'un simple éditeur de texte. Cependant, il y a des outils tels que Microsoft Visual Studio qui prennent en charge la production de code XAML. Par rap-

port à X3D, l'intérêt de son utilisation est surtout parce qu'il fonctionne directement dans le navigateur Internet sous les plateformes Windows sans besoin d'un plugiciel spécifique. Cependant, la portabilité des scènes XAML vers d'autres systèmes n'est pas possible, car les primitives utilisées dans XAML utilisent l'implémentation dans les classes du système Windows (MacVittie, 2006).

3DXML

C'est un format de données 3D créé par Dassault Systèmes, compagnie qui développe le logiciel de conception assistée par ordinateur CATIA. La spécification de ce format a été rendue publique en juin 2005. 3DXML s'appuie sur le langage XML pour modéliser et partager du contenu 3D. Ce format n'est pas très différent du format X3D au niveau conceptuel. La différence est relative à l'évolution et au support accordé aux deux langages, par conséquent, l'adoption et l'évolution de 3DXML reposent principalement sur le succès de la compagnie Dassault Systèmes (Ding *et al.*, 2007).

CITYGML

CityGML est un modèle de données ouvert développé pour le stockage et l'échange de données 3D de villes. Il a été adopté comme standard officiel par l'Open Geospatial Consortium ou OGC. C'est un consortium international créé pour développer et promouvoir des standards ouverts. CityGML est implémenté avec Geography Markup Language (GML), mais il utilise seulement un sous-ensemble du schéma GML. Contrairement à des formats comme X3D ou COLLADA, CityGML ne représente pas seulement l'aspect graphique des modèles de villes, mais il permet la représentation des propriétés sémantiques et thématiques par exemple : taxonomies, bâtiments, végétation, plans d'eaux, moyens de transport et mobilier urbain. L'intégration de la géométrie avec les données sémantiques dans le même format permet l'utilisation du contenu 3D dans d'autres cas en plus de la simple visualisation. CityGML distingue quatre niveaux différents de détails, à partir d'un simple bloc jusqu'à la construction des modèles d'architecture (Herrlich *et al.*, 2010).

X3D

X3D est un standard « open source » pour le rendu Web des graphiques 3D. Il spécifie un langage déclaratif de définition de la géométrie, un moteur d'exécution et une interface de programmation qui fournissent un environnement interactif en temps réel pour les graphiques 3D qui fonctionnent au-dessus d'un réseau. Il a été initialement proposé en 2002 comme une version améliorée de VRML. En août 2004, il a été reconnu comme standard ISO/IEC 19775

dans les besoins de communication pour les scènes 3D en temps réel. Les documents de spécification X3D sont disponibles gratuitement et ce standard peut être utilisé librement. X3D est conçu pour des applications où des modèles 3D et des comportements peuvent mieux illustrer des relations spatiales et des événements interactifs qui sont normalement difficiles à montrer. Par exemple, le stade de football en Allemagne (visualisation), l'analyse visuelle des acides aminés (scientifique), simulateur de radiothérapie et des systèmes chirurgicaux (traitement médical), l'Atlas de la terre (éducation et recherche scientifique), la force de protection contre le terrorisme pour les États-Unis (planification de la mission), etc. X3D fournit une variété des capacités utilisées par la communauté 3D pour le rendu, la modélisation, l'animation et l'interactivité avec l'utilisateur. Entre les fonctionnalités avancées, nous retrouvons le positionnement géospatial, l'animation humanoïde et la simulation interactive repartie Protocol de réseaux (DIS). Le rendu, les textures et les fonctions de modélisation sont multiplateforme basée sur les fonctionnalités des moteurs OpenGL et DirectX. X3D fournit des fonctionnalités simples pour l'animation et l'interactivité. Il utilise l'animation par image clé avec l'interpolation linéaire. Des animations complexes et l'interactivité peuvent être pilotées par un logiciel écrit en JavaScript. X3D a subi des évolutions au cours des années en fonction des besoins relevés par la communauté 3D provoquant des rectificatifs en 2006 de la spécification initiale afin de prendre en compte les nombreuses suggestions apportées par la communauté 3D. Donc, dès le départ, X3D a été bâti en fonction des besoins réels des utilisateurs (Daly et Brutzman, 2008; Arnaud et Parisi, 2010).

COLLADA

COLLADA est un format ouvert de fichier d'échange pour des applications interactives 3D. Il fournit un langage de description qui est utilisé par les plus grands logiciels de création 3D comme 3ds Max, Maya ou SoftImage XSI ce qui motive à plusieurs d'autres logiciels de création 3D le supporter également. Il a été développé par Sony et d'autres compagnies ont collaboré pour créer un outil qui serait avantageux au plus large public possible. COLLADA est toujours en évolution grâce aux efforts de contributeurs Khronos Group.

COLLADA permet de définir du contenu 3D basé sur le schéma XML, donc il dévient possible de transporter du contenu 3D entre des applications 3D sans aucune perte d'information. Habituellement, chaque application 3D est uniquement compatible avec son propre format pour leurs modèles géométriques, les données des matériaux, les données de la scène, etc. Donc, le contenu 3D d'une application ne peut pas être manipulé, ni accepté par d'autres applications 3D. Si l'utilisateur veut exporter du contenu 3D vers une autre application, il doit l'exporter, mais il y aura une perte d'information parce que les données 3D d'une application vers une autre sont différentes. Pour régler ce problème, COLLADA a été proposé en tant que

futur format de fichier standard de contenu graphique 3D. Il permet le transport du contenu 3D entre des applications de manière approprié en utilisant des étiquettes XML pour chaque type de donnée comme la géométrie, les matériaux et les animations. Pour d'autres données plus avancées, COLLADA propose COLLADA FX pour des effets avec des nuanceurs et COLLADA PHYSICS pour la simulation de la physique. De plus, d'autres données peuvent être ajoutées, il est facilement extensible sans aucune modification nécessaire(Miyahara et Okada, 2009). Il est surtout un format intermédiaire qui a comme objectif de représenter du contenu 3D de diverses façons(Arnaud et Parisi, 2010). Ces caractéristiques pourraient faire de COLLADA le fichier d'échange pour les logiciels de modélisation 3D (Preda *et al.*, 2010).

1.6 Systèmes de génération de scènes animées 3D

Dans cette section, nous présenterons quelques systèmes de génération des scènes 3D qui ressemblent à certains niveaux au projet GITAN. Ces systèmes montrent les différentes façons de générer des scènes 3D animées ou non à partir du texte ou des dialogues et en fonction du domaine d'application. Nous décrirons brièvement le fonctionnement de ces systèmes.

CAMEO

C'est un système d'animation automatique 3D pour la cinématographie immersive créé à l'institut avancé de technologie Samsung en Corée du Sud. Ce système prend trois entrées : le dialogue, la capture d'écran et le style du scénario. Avec ce système les utilisateurs peuvent créer des animations 3D en écrivant l'histoire ou le dialogue et en choisissant les modèles 3D, les personnages et le style du scénario. Le dialogue permet d'écrire ce que les personnages ou le narrateur diront. Les captures d'écran permettent de fournir l'information à propos des modèles 3D qui sont affichés à l'écran, les lumières et les caméras. Finalement, le scénario contient l'information générale à propos de la scène, l'atmosphère, le temps ou le thème. Pour la description de ces entrées, ils ont structuré trois types de schémas XML qui permettent d'entrer ces données. Leur système est composé par une base de données des directions et un ensemble de directeurs comme la caméra, le son, la lumière, le personnage et directeur d'art. Chaque directeur prend comme entrée les trois documents XML et produit les directions nécessaires en utilisant les règles définies dans la base de données de directions. Cette base de données fournit l'information pour réaliser les effets de cinématique pour chaque directeur. Donc, à partir des trois documents XML qui décrivent l'animation le système générera l'animation automatiquement (Shim et Kang, 2008).

BEHAVIOR 3D

C'est un projet développé par Raimund Dachselt à Dresten University of Technology. Ce système permet la déclaration des comportements sur des objets 3D afin de permettre une interaction avec l'utilisateur. Il se base sur le standard X3D et utilise le concept de prototype de X3D pour introduire un nouveau nœud de comportements. Le concept de nœuds permet l'utilisation des fonctionnalités orientées objet telles que l'héritage, le typage et le polymorphisme. Avec cette extension, des animations plus complexes peuvent être définies à partir de X3D. Une abstraction de l'animation de haut niveau est disponible en utilisant une interface simple qui est définie pour chaque nœud de comportement. Un nœud est un comportement constituant une partie du graphe de comportements et représente certaines fonctionnalités de l'application 3D. Ils peuvent être utilisés avec les autres nœuds du graphe de scène. Ils proposent un riche ensemble de comportements prédéfinis pour la création des animations et des machines à état (Dachselt et Rukzio, 2003).

PUT

C'est un système interactif de manipulation d'objets basé sur le langage naturel. Il a été développé par Silicon Graphics à l'Université de Californie à Santa Cruz. Il permet d'utiliser le langage naturel pour décrire une scène et spécifier des relations spatiales entre les objets 3D. Pour l'affichage 3D, le système utilise Iris Inventor qui est une boîte à outils graphique 3D de Silicon Graphics. Le système permet la spécification des objets géométriques, ses propriétés et ses relations spatiales. Le langage utilisé est simple et possède une grammaire limitée pour garder le système prévisible et facile à utiliser. Pour ce faire, le système identifie l'objet à être placé comme 'Trajector' (TR) et l'objet de référence comme 'Landmark' (LM) ou point de repère. De cette façon, des relations spatiales comme 'on' ('sur' en français) créent de relations ou contraintes entre les objets TR et le LM. Les verbes de placement, comme 'PUT', spécifient le placement des objets TR. Pour établir les relations entres les objets, PUT définit et redéfinit les régions de placement sur chaque objet LM. Le système permet de composer plusieurs relations pour des descriptions plus complexes. Par contre, les relations spatiales sont assez limitées, mais permettent une bonne interaction pour le placement des objets (Clay et Wilhelms, 1996).

WordsEye

C'est un système qui permet de convertir un texte en anglais en scènes 3D statiques. Le projet a été développé par le laboratoire de recherche AT&T. Il gère la création des scènes à travers des poses prédéfinies, des règles et des contraintes graphiques. Il se base sur des

cadres verbaux pour interpréter sémantiquement les textes et utilise WordNet pour régler le problème de sous-spécification et la désambigüisation causés par la langue. Après l'analyse linguistique, les scènes sont définies en termes d'un ensemble de 'depictors' qui sont des objets 3D avec ses propriétés spatiales et graphiques. Le module de représentation graphique de WordsEye permet de traduire le haut niveau de représentation sémantique produit par l'analyse sémantique dans ces 'depictors' de bas niveau. Par la suite, ces 'depictors' sont traités et le système résout les contraintes implicites et les contradictions possibles. Ensuite, les références des objets 3D sont lues et les 'depictors' et elles sont attribuées et appliquées en respectant les contraintes. Finalement, la scène est mise en place progressivement et le système ajoute l'environnement d'arrière-plan, le plancher de base et les lumières (Coyne et Sproat, 2001).

Système de représentation graphique du texte de l'Université de Rhodes

Le système créé par le département d'informatique de l'université de Rhodes en Grahamstown South Africa permet l'utilisation de n'importe quel texte sans restriction et permet de le représenter en forme graphique. Le système est basé sur des contraintes et permet l'animation en fonction du temps. Il représente les objets par des entités et il crée des relations entre elles pour fixer des contraintes. Il utilise une base de données avec des modèles 3D standardisés et annotés. Tous ces objets ont été créés avec l'orientation sur l'axe des Z et ils sont annotés avec un ensemble de mots clés qui le décrivent. Pour localiser les objets dans la base de données, le système utilise 'WordNet' pour faire abstraction de la terminologie lexicale. Par la suite, en fonction du texte, les contraintes sont actives sur un intervalle de temps et les trajectoires sont calculées par la résolution des contraintes en utilisant l'arithmétique d'intervalles (Glass et Bangay, 2007).

1.7 Animation et positionnement de la caméra virtuelle

La caméra et son déplacement sont très importants dans les scènes 3D animées. Son positionnement dans des mondes virtuels 3D pose les mêmes problèmes que les cinématographes ont au cinéma ou à la télévision. Chaque image montrée doit communiquer un message spécifique. Par conséquent, la caméra doit être placée attentivement pour voir clairement les éléments importants pour permettre garder l'attention de l'observateur. Plusieurs projets ont essayé de manipuler la caméra virtuelle afin d'améliorer l'expérience usager et augmenter l'immersion dans le monde virtuel. On peut par exemple déterminer la position de la caméra en utilisant l'aire de déplacement (Kwon et Lee, 2008), en utilisant les concepts de la cinématographie (Li et Cheng, 2008) ou en utilisant des contraintes (Bares et Lester, 1999).

Nous ne donnerons pas une description plus détaillée, puisque la gestion de la caméra est un aspect assez complexe qui est traité par un autre projet de recherche. Dans cette recherche, la caméra n'est pas gérée, mais on considère que c'est un point très important pour le projet GITAN.

1.8 Récapitulation des connaissances

Dans ce chapitre, nous avons présenté la revue bibliographique et la revue des connaissances des éléments importants pour cette recherche. Nous avons identifié les concepts importants, les outils existants, et les projets similaires ou précurseurs dans la génération des scènes 3D. Nous ferons une brève récapitulation des éléments que nous avons exposés jusqu'au présent.

Premièrement, nous avons expliqué les concepts du média 3D, les graphes de scène, les géométries et les animations 3D. Ce sont des éléments importants qu'il faut bien définir, puisque ce sont les éléments de base de cette recherche. Nous avons également présenté les différents types d'animations existantes. Nous avons pu identifier que l'animation par image clé est une méthode d'animation qui nous permet facilement de créer des animations simples. Notre projet propose de créer des animations simples capables de passer des messages, donc ce type d'animation est le plus indiqué pour générer nos animations.

De plus, nous devons être capables de générer la scène 3D avec le monde virtuel. Donc, nous avons présenté les différentes façons de générer des scènes et nous avons énuméré les différents outils et les solutions disponibles pour notre système. Les interfaces de programmation graphiques que nous avons présentées sont de très bons outils dans la génération des scènes 3D et ils permettent facilement de créer des scènes complexes et très réalistes. Les plateformes graphiques, quant à elles, sont très puissantes et complètes pour la création des jeux de vidéo. De plus, nous avons vu les langages script d'animation qui sont une solution très complète dans la génération des scènes 3D. Les scripts que nous avons présentés permettent la représentation des expressions faciales, l'animation des personnages virtuels et la création des scènes 3D. Les scripts d'animation sont une solution très intéressante pour notre projet.

Par la suite, nous avons montré les différents formats de fichier d'échange 3D. Si nous voulons représenter une scène animée avec un format 3D, il doit être simple, il doit être complet, il doit permettre de représenter des animations et il doit supporter le type d'animation par image clé. Nous avons présenté les formats de représentation 3D les plus utilisés et les plus populaires comme VRML, U3D, XAML, 3DXML, CITYGML, X3D et COLLADA. Chaque format offre des caractéristiques particulières qui sont appliquées à certains domaines d'application. Les formats X3D et COLLADA sont les formats les plus intéressants à utiliser

pour représenter des animations 3D. Nous présenterons au chapitre 2 une analyse plus en profondeur des différents formats et le choix que nous utiliserons dans notre système.

Finalement, nous avons présenté différents systèmes de génération de scènes 3D à partir des textes ou des dialogues. Les différents systèmes se ressemblent sur certains niveaux, mais ils sont surtout influencés par le domaine d'application de chacun d'entre eux. En ce qui concerne le module graphique, les différents systèmes utilisent des scripts pour la description et l'animation de la scène au moyen des règles, des contraintes ou des relations entre les objets ce qui est très intéressant comme solution.

Nous avons présenté différentes concepts, méthodes et solutions dans la création des scènes animées 3D. Le système qui en découle de cette recherche s'inspire des systèmes semblables existants et propose une solution simple et efficace pour créer des scènes animées de façon automatique. Les systèmes comme WordsEye, Put ou CAMEO utilisent des scripts qui servent comme langage de description pour l'animation à créer. Ces scripts sont souvent très en lien avec le domaine d'application du système. Pour cette raison, nous avons décidé de créer notre propre formalisme de description de la scène animée. De plus, pour la conversion de ce formalisme vers l'animation 3D, nous avons jugé que les interfaces de programmation graphiques et les moteurs de jeu sont des outils très puissants et trop complexes pour les besoins de notre système. Nous n'avons pas besoin du réalisme des jeux vidéo, notre système mise sur la simplicité et la bonne capacité à transmettre des messages. Donc, nous nous sommes inspirés de BEHAVIOR3D et nous pensons aussi qu'il est intéressant de construire une scène animée en utilisant un format de représentation 3D. En effet, BEHAVIOR 3D se base sur X3D pour représenter ses animations, mais de façon différente pour prédéfinir des comportements. Nous nous baserons sur COLLADA pour définir l'animation au complet. Par la suite, le fichier contenant l'animation 3D sera joué sur un visualiseur 3D qui supporte ce type de format. Les scripts d'animation sont très flexibles et si le format d'échange 3D est assez complet, nous pourrons représenter une scène animée 3D sous ce format et regarder l'animation 3D sur un visualiseur. Pour ce faire, dans la section suivante nous présenterons les hypothèses et les objectifs de cette recherche.

1.9 Hypothèses de recherche et objectifs

Dans cette section, nous présenterons l'objectif général du projet, les hypothèses de recherche posées pour atteindre cet objectif et les objectifs spécifiques découlant des hypothèses.

1.9.1 Objectif général

Proposer un système logiciel capable de générer automatiquement des animations 3D qui représentent le sens d'une phrase.

1.9.2 Hypothèses

Hypothèse 1 : Une phrase simple peut être représentée par un langage script ou formalisme spécifique qui fasse abstraction et qui décrive les événements d'une phrase afin d'avoir l'information nécessaire pour la création d'une séquence animée 3D.

Hypothèse 2 : Il est possible de traduire la description d'une phrase dans le formalisme implanté vers une animation 3D dans un format d'échange 3D.

Hypothèse 3 : La séquence animée créée par notre système permet de communiquer le sens d'une phrase simple.

1.9.3 Objectifs spécifiques

1. Déterminer le formalisme ou langage script à utiliser qui fasse abstraction de la scène animé 3D.
2. Déterminer le format d'échange 3D le plus approprié pour représenter le sens de n'importe quelle phrase sous la forme d'une animation 3D.
3. Proposer un système capable de lire le script décrivant la phrase et de le traduire dans un format 3D contenant l'animation qui représente la phrase initiale.
4. Valider que la séquence animée 3D générée par le système logiciel soit représentative du sens de la phrase initiale à partir de nos résultats.

CHAPITRE 2

MÉTHODOLOGIE

Cette section expose le cadre méthodologique retenu pour atteindre les objectifs du projet et permettre de vérifier les hypothèses posées en s'appuyant sur la revue des connaissances et de la littérature du chapitre précédent.

2.1 Déterminer les entrées du système

Cette première étape doit permettre d'assurer que nous avons l'information nécessaire et minimale pour la création d'une séquence animée 3D. Le système doit être capable de traiter cette information avec différents algorithmes afin de traduire et créer une animation 3D. Donc, toute l'information qui est utilisée pour construire l'animation doit être clairement définie pour permettre la bonne interprétation de celle-ci. Nous décrirons les différentes entrées du système que nous avons déterminées et définies dans les sous-sections suivantes.

2.1.1 Modèles et animations 3D utilisés dans le projet

Pour le développement de ce projet, nous avons eu besoin de plusieurs modèles et animations 3D avec lesquels nous avons travaillé au cours de cette recherche. La base de données des modèles 3D que nous avons créés est limitée et elle n'est pas suffisante pour la représentation de toute animation 3D, mais elle nous donne le support nécessaire pour démontrer nos hypothèses et atteindre nos objectifs. Cette base de données nous permet de créer plusieurs scénarios d'animation sur lesquels nous avons testé nos résultats. Tous les modèles 3D ont été créés en respectant une hiérarchie spécifique des nœuds et en ajoutant une couche sémantique qui permet de les identifier et de les manipuler plus facilement.

La spécification de la hiérarchie à respecter et la nomenclature à utiliser est un aspect important de cette recherche qui nous a permis de rester au niveau conceptuel et de faire abstraction des détails de la modélisation propres à l'infographie. Il faut noter que la modélisation et la création des objets et des animations 3D a été faite par un artiste 3D, Daniel Petels, qui a travaillé avec nous pendant un certain temps. Le travail de notre artiste 3D a consisté à la génération des modèles 3D et des animations 3D prédéfinies en fonction d'une hiérarchie que nous avons définie.

La figure 2.1 montre le monde virtuel créé. Elle montre une petite ville qui nous pouvons utiliser comme environnement pour nos scènes 3D. Elle est composée de différents objets,

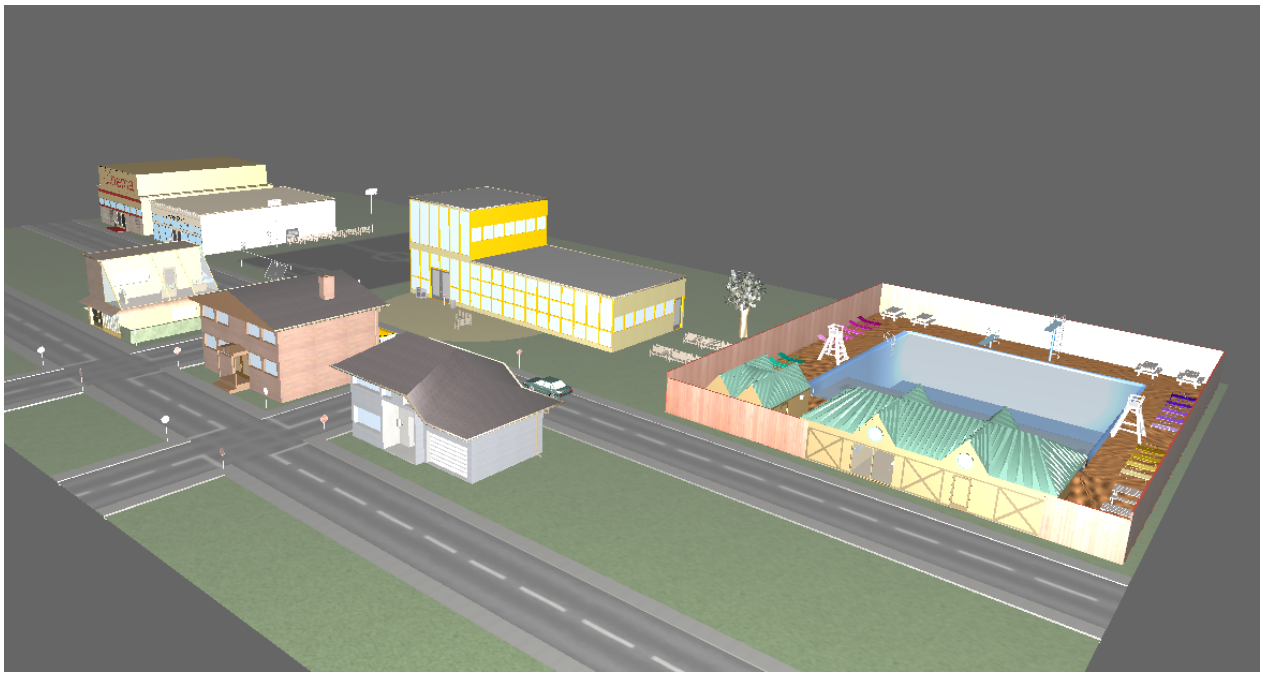

Figure 2.1 Modèle de la ville

structures, véhicules et personnages. La figure 2.2 montre mieux les modèles de la piscine, le cinéma, un magasin et une école avec son terrain de soccer.

Nous avons aussi différents types de maisons. La figure 2.3 montre les modèles de maisons que nous pouvons voir dans notre petit village.

De plus, nous avons à notre disposition l'intérieur de chaque structure ou maison pour créer nos scénarios. La figure 2.4 montre l'intérieur de quelques-unes d'entre elles.

Pour les véhicules, nous avons deux autos, un autobus et un avion. La figure 2.5 montre les modèles 3D des véhicules utilisés.

Finalement, nous avons créé différents personnages qui interagissent avec notre village

Figure 2.2 Modèles 3D dans la ville

Figure 2.3 Modèles 3D des maisons

Figure 2.4 Modèles 3D de l'intérieur

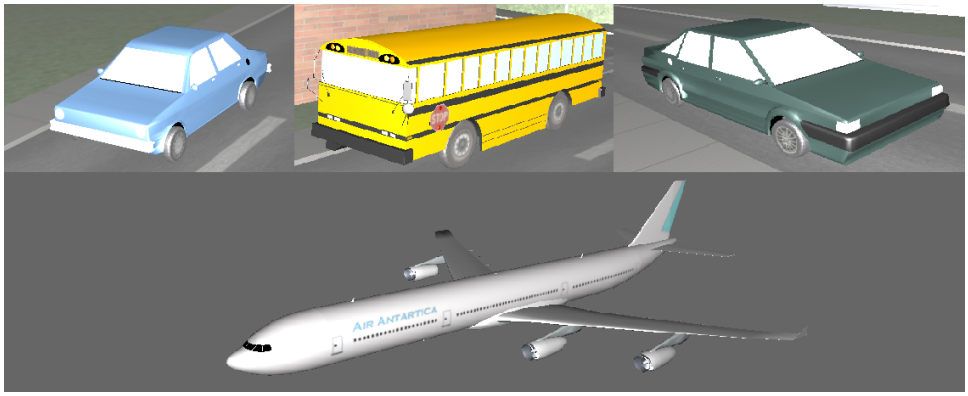

Figure 2.5 Modèles 3D des véhicules

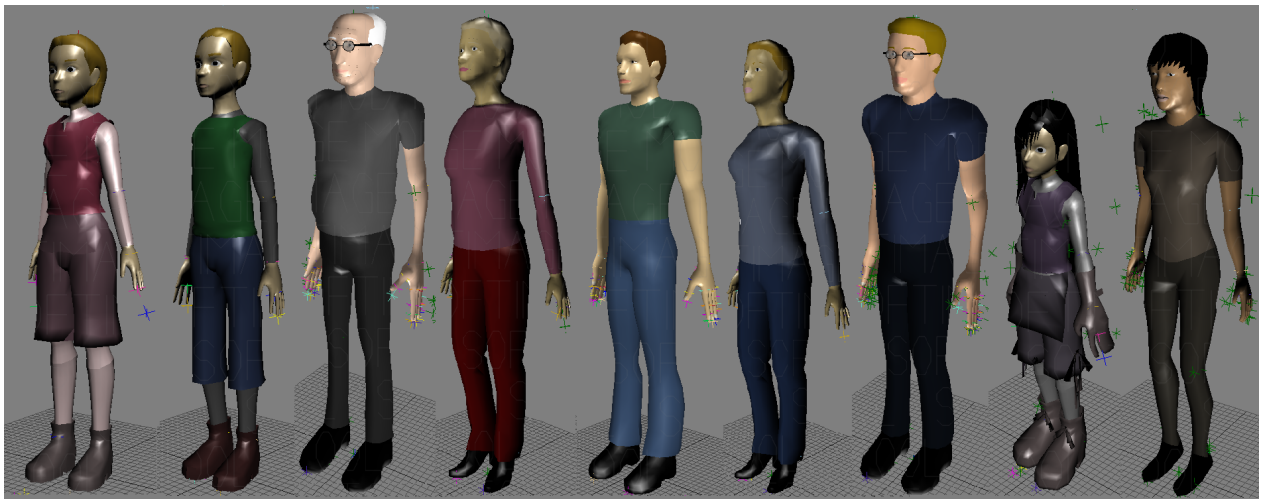

Figure 2.6 Modèles 3D des personnages

pour créer nos animations 3D. La figure 2.6 montre les différents types de personnages que nous avons créés.

Chaque modèle 3D a été créé avec une hiérarchie des nœuds que nous avons définis pour permettre l'introduction d'une couche sémantique. La figure 2.7 montre la hiérarchie utilisée.

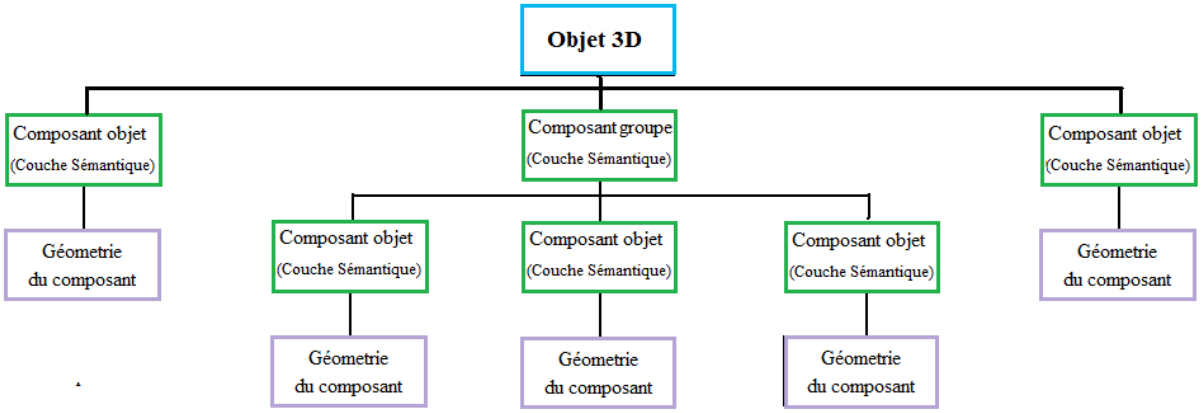

Figure 2.7 Hiérarchie des modèles 3D

La hiérarchie des modèles 3D que nous avons définie est composée par le nœud objet étiqueté avec le nom de l'objet 3D. Le nœud objet peut être décomposé en plusieurs nœuds, il existe deux types de nœuds qui permettent d'ajouter une couche sémantique aux composants de l'objet 3D : les nœuds composant groupe et les nœuds composant objet. On utilise des nœuds composant groupe pour regrouper plusieurs nœuds qui peuvent avoir une décom-

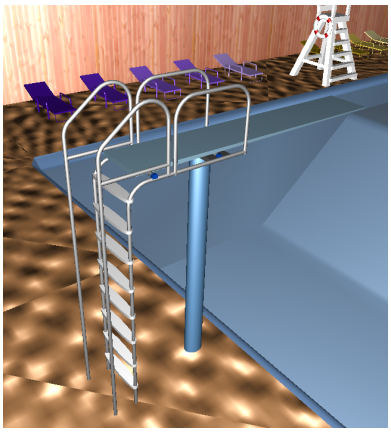

Figure 2.8 Modèles 3D du plongeoir

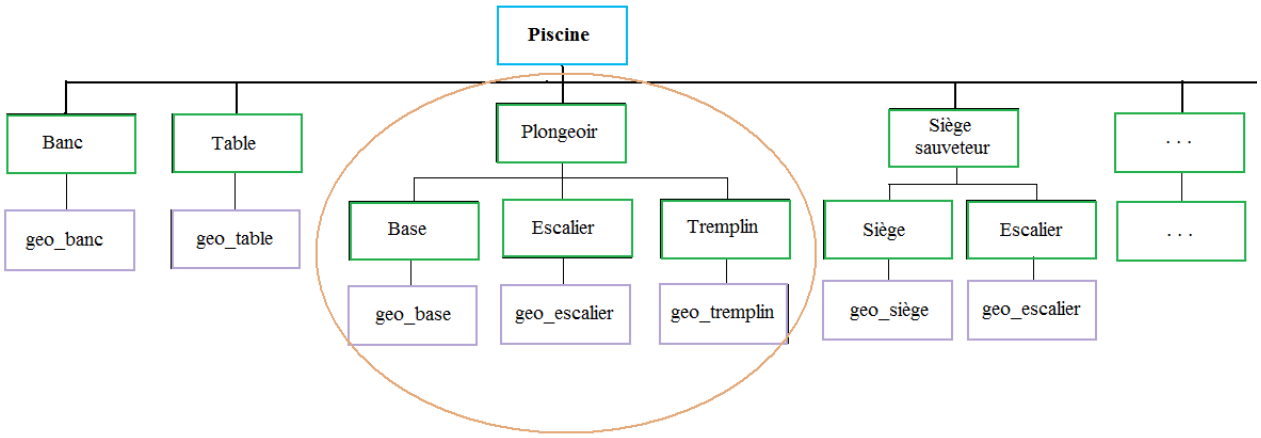

Figure 2.9 Hiérarchie du plongeoir

position sémantique et les nœuds objet pour identifier la composante du modèle 3D. Cette dernière contient la géométrie du composant. La figure 2.9 montre un exemple de l'utilisation de cette hiérarchie sur un modèle 3D.

La figure 2.8 montre le modèle du plongeoir dont la représentation hiérarchique est décrite à la figure 2.9. Nous pouvons remarquer que le plongeoir est composé de trois géométries lesquelles sont contenues dans des nœuds composants qui nous permettent d'identifier chaque partie du plongeoir soit la base cylindrique, l'escalier ou le tremplin. Cette décomposition hiérarchique de la piscine nous permet de manipuler chacun des composants qui pourraient avoir une valeur sémantique.

Pour accéder à un composant du plongeoir, nous utilisons les parents dans la hiérar-

chie pour nommer le composant. Donc, par exemple, si nous voulons accéder à l'escalier du plongeoir il faut écrire :

`Piscine.Plongeoir.Escalier`

De cette façon, nous pouvons manipuler des concepts et les distinguer les uns des autres. Dans cet exemple, nous pouvons distinguer l'escalier du plongeoir de l'escalier du siège du sauveteur.

Pour ce qui est des animations 3D prédéfinies, nous avons développé quelques animations qui sont appliquées sur les personnages 3D pour leur permettre se déplacer. Nous n'avons pas des postures ou des expressions faciales pour le moment. Nous jugeons qu'il est prioritaire d'implémenter des animations qui nous permettent de représenter le déplacement de nos personnages. Donc, pour chaque personnage nous pouvons représenter par exemple le cycle de marche ou le cycle de course et des sauts. La figure 2.10 montre une séquence d'images du cycle de marche de notre personnage.

2.1.2 Détermination du formalisme décrivant l'animation 3D

Pour la création d'une animation, nous avons besoin de définir les objets dans la scène, la relation entre eux dans le monde virtuel et les événements en fonction du temps. De plus, nous devons définir des paramètres pour identifier les objets et décrire ses caractéristiques (position, orientation, couleur, taille, etc.) en fonction du temps. Donc, pour satisfaire au premier objectif spécifique de notre recherche, il faut déterminer un formalisme qui peut nous permettre de décrire l'animation que nous voulons représenter en faisant abstraction des détails de bas niveau de l'animation 3D traditionnelle. Finalement, il faudra rester au niveau conceptuel de façon à manipuler l'information que nous pouvons extraire à partir d'une phrase.

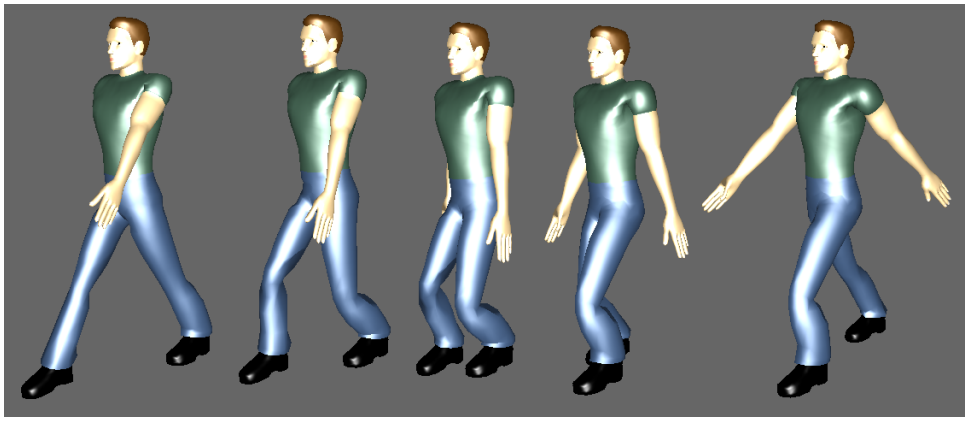

Figure 2.10 Séquence d'images du cycle de marche des personnages

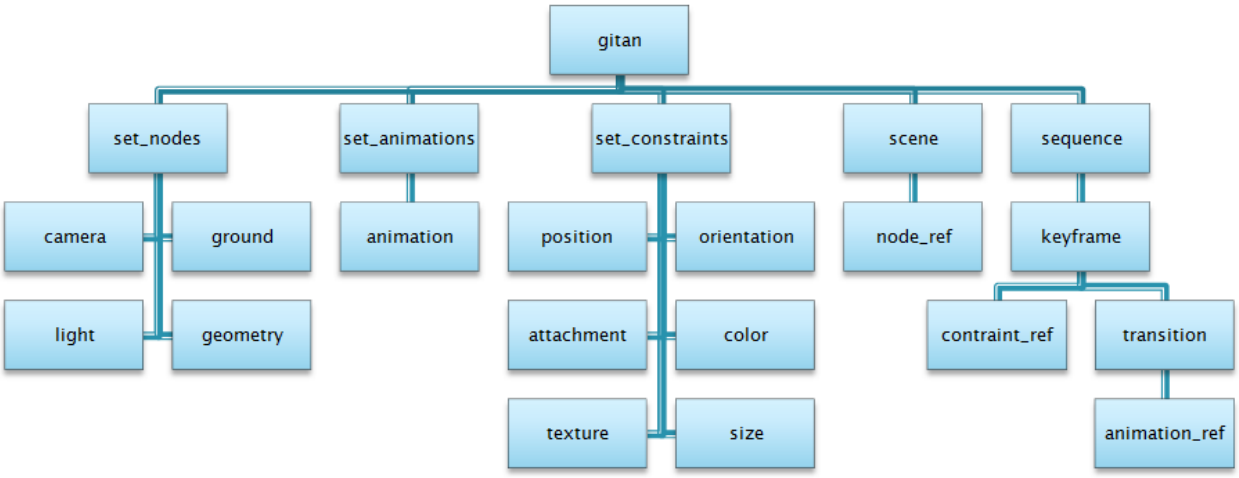

Figure 2.11 Formalisme de représentation des animations 3D

Pour ce faire, nous utiliserons le format XML pour encoder le formalisme que nous définirons. XML est un langage de balisage tout comme HTML. Il a été conçu pour transporter et stocker des données sous forme lisible par la machine. Les balises ne sont pas prédéfinies et il est conçu pour être autodescriptif(W3C, 2009). Il est très répandu et utilisé, c'est donc un bon moyen de construire notre formalisme de représentation d'une animation 3D au niveau conceptuel de la phrase.

La figure 2.11 montre les composantes générales du formalisme que nous avons spécifiées pour la représentation d'une scène animée 3D. Il est le format d'entrée de notre système qui spécifie l'animation à représenter.

Le schéma de cette figure présente le modèle de représentation qui nous permet de décrire une animation 3D. Il est composé de cinq groupes principaux : `set_nodes`, `set_animation`, `set_constraints`, `scene` et `sequence`.

- `set_nodes` :

Il permet de définir les différents nœuds de la scène soit les géométries, la caméra, la lumière ou le sol du monde virtuel. Pour ce projet de recherche, nous traitons que les géométries et nous utilisions des valeurs par défaut pour les autres nœuds, puisque la manipulation de la caméra ou les lumières pourraient faire partie d'autres projets de recherche.

Les nœuds des géométries qui sont définies dans cette section font référence à une base de données contenant des objets 3D en format COLLADA. Ces objets 3D possèdent une couche

sémantique qui nous permet de faire une recherche sur la base de données en utilisant les concepts identifiés dans les phrases. Nous avons vu à la section précédente la composition des objets 3D, donc, la couche sémantique avec laquelle nous manipulons nos modèles 3D nous permet de rester au niveau conceptuel de la phrase. De cette façon, nous pouvons faire abstraction de tous les détails de bas niveau de l'ensemble de polygones qui forme les modèles 3D.

Par exemple, si nous désirons représenter l'animation pour la phrase : « Une fille va au cinéma », la syntaxe XML pour déclarer les nœuds serait la suivante :

```
<set_nodes>
  <camera id="cam"/>
  <ground id="ground" />
  <light id="light" />
  <geometry id="cinema" scale="1.0" uri="C:\svn\modeles\cinema.dae"/>
  <geometry id="fille" scale="1.0" uri="C:\svn\animeeple\girl-walk.dae"/>
</set_nodes>
```

Nous devons spécifier les géométries avec ses attributs soit l'identificateur, la proportion de la taille et l'adresse URI du modèle 3D qui contient la géométrie. Dans cet exemple, nous avons déclaré deux nœuds : le cinéma et la fille.

- set_animation :

Il permet de définir les objets 3D qui ont des animations pour construire l'animation finale. Les animations qu'on peut définir sont de deux types : des animations d'activité et des animations de déplacement.

Le premier type, les animations d'activité, sont des animations prédéfinies qui nous permettent principalement de représenter des comportements, des postures, des signes ou des gestes des personnages et certains mouvements prédéfinis des objets comme ouvrir ou fermer une porte. Nous avons seulement implémenté les cycles de marche, les cycles de course et les sauts de nos personnages pour leur permettre de se déplacer, mais l'idée est d'avoir une base de données d'animations prédéfinies pour des gestes, des postures ou des expressions faciales ou corporelles.

Le second type, les animations de déplacement, sont des animations que nous générerons et appliquerons sur les modèles. Ces animations permettent de déplacer les objets d'un point à un autre suivant une trajectoire déterminée. La trajectoire peut être linéaire ou balistique.

Lorsque nous composons ces deux types d'animations, nous pouvons créer plusieurs scénarios d'animation. Nous pouvons définir le déplacement de nos objets et appliquer des animations prédéfinies comme le cycle de marche pour animer un personnage qui se déplace en marchant par exemple.

La syntaxe XML pour définir des animations est la suivante :

```
<set_animations>
  <animation id="fille-walk-cycle" looping="true" node="fille" type="activity">
    <param name="uri">C:\svn\animeeple\girl-walk.dae</param>
  </animation>
  <animation id="filleTranslate" looping="false" node="fille" type="displacement">
    <param name="trajectory">nodeAvoidance</param>
    <param name="speed">40</param>
  </animation>
</set_animations>
```

Lorsqu'on déclare des animations, il faut définir l'identificateur de l'animation, le mode de séquence soit une seule fois ou en boucle, le nœud sur lequel on applique l'animation et le type d'animation soit une activité ou un déplacement. Pour les animations d'activité, il faut spécifier l'uri de l'animation prédéfinie ; et pour les animations de déplacement, il faut spécifier la trajectoire à suivre soit linéaire ou en balistique et la vitesse du déplacement.

Dans notre exemple, nous avons spécifié deux types d'animation pour représenter que la fille se dirige vers le cinéma en marchant. La première définit l'animation de cycle de marche pour la fille et la deuxième définit le déplacement de la fille avec une trajectoire linéaire à une vitesse de 40 unités par seconde.

- set_constraints :

Nous avons développé un système de contraintes de différents types pour définir le positionnement, l'orientation, et le redimensionnement des objets 3D. Les contraintes sont appliquées sur les nœuds de géométries et sont activées par la suite sur chaque image clé.

La syntaxe XML pour définir les contraintes est la suivante :

```
<set_constraints>
  <position id="filleInfrontcinema" node="fille" reference="cinema"
    type="inFrontOf" worldProjection="horizontalPlane">
  </position>
  <position id="filleFarFromcinema" node="fille" reference="cinema"
    type="atDistance" worldProjection="horizontalPlane">
    <param name="distance">200</param>
  </position>
  <orientation id="filleTowardscinema" node="fille" reference="cinema"
      type="toward"/>
  </ orientation >
</set_constraints>
```

Pour la définition des contraintes, il y a plusieurs attributs qu'il faut définir pour chaque type de contraintes. Nous avons implémenté quatre types de contraintes :

- Contrainte de position : Elle permet de placer les objets en fonction d'un ou plusieurs objets dans le monde virtuel. La déclaration de cette contrainte doit spécifier l'identificateur, le nœud sur lequel on applique la contrainte, le nœud de référence, la projection (horizontale ou verticale) et finalement le type de position en fonction de la référence soit proche, loin, entre, à gauche, à droite, par dessus, par dessous, etc. La liste de toutes les valeurs possibles du type de position se trouve à l'annexe A.
- Contrainte d'orientation : Elle permet de fixer l'orientation de l'objet en fonction de l'orientation d'un autre objet de référence. Pour déclarer ce type de contrainte, il faut spécifier l'identificateur, le nœud sur lequel on applique la contrainte, le nœud de référence et le type d'orientation soit perpendiculaire, parallèle, dirigé vers ou contre la référence.
- Contrainte de redimensionnement : Elle permet de fixer la taille des objets. Elle définit la proportion de la dimension d'un objet par rapport à une référence. Les attributs à spécifier sont : le nœud à redimensionner, le nœud de référence, la proportion de redimensionnement et le type de redimensionnement.
- Contrainte d'attachement : Elle permet de fixer la position d'un objet en fonction d'un autre. L'objet qui est attaché suit la même trajectoire de déplacement que l'autre lorsque cette contrainte est définie.

Dans notre exemple, premièrement, nous avons déclaré deux contraintes de position pour placer la fille à deux positions soit loin du cinéma à une distance de 200 unités ou soit proche du cinéma juste devant celui-ci. Par la suite, nous avons déclaré une contrainte d'orientation pour placer la fille en regardant le cinéma. La dimension de nos modèles 3D garde une proportion normale, donc nous n'avons pas spécifié des contraintes de redimensionnement.

- scene :

La scène fait référence aux nœuds qui sont présents et visibles dans la scène. Nous pouvons déclarer des nœuds de géométries, mais s'ils ne sont pas présents dans cette section, ils ne seront pas visibles dans la scène générée. La syntaxe pour déclarer la scène est la suivante :

```
<scene>
  <!-- The lists of nodes used in the scene -->
  <node_ref node="fille"/>
  <node_ref node="cinema"/>
</scene>
```

Dans cet exemple, nous déclarons le nœud de référence de la fille et le cinéma qui sont les deux modèles qu'on verra dans la scène à générer.

- sequence :

Cette section permet de décrire l'animation à créer en utilisant les contraintes et les animations que nous avons déclarées. Pour ce faire, nous activons les contraintes déclarées sur chaque image clé et les animations à utiliser. Pour la création de notre scène animée, nous utilisons l'animation par image clé. La séquence est donc composée en images clés et en transitions comme nous pouvons voir à la figure 2.12.

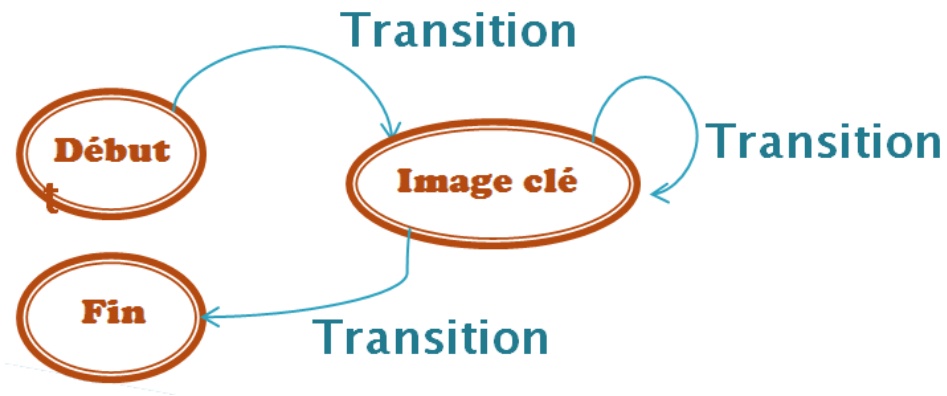

Figure 2.12 Séquence d'une animation

Cette figure montre une séquence d'images clés connectées par des transitions. Dans l'image clé, nous activons les contraintes qui permettent de modifier l'état des nœuds. Par la suite, dans les transitions, nous indiquons les animations prédéfinies qui sont jouées entre deux images clés, puis nous indiquons les animations de déplacement qui donnent le type de trajectoire à suivre. La syntaxe XML pour définir une séquence est la suivante :

```
<sequence>
  <keyframe>
    <toggle activate="true" constraint="filleInfrontcinema"/>
    <toggle activate="true" constraint="filleFarFromcinema"/>
    <toggle activate="true" constraint=" filleTowardscinema"/>
    <transition major="filleTranslate">
      <animation_ref animation="fille-walk-cycle"/>
      <animation_ref animation="filleTranslate"/>
      <keyframe>
<toggle activate="true" constraint="filleInfrontcinema"/>
<toggle activate="true" constraint=" filleTowardscinema"/>
      </keyframe>
    </transition>
  </keyframe>
</sequence>
```

Dans notre exemple, nous avons deux images clés. La première image active trois contraintes : une pour placer la fille juste devant le cinéma, l'autre contrainte éloigne la fille de 200 unités et finalement la dernière permet d'orienter la fille faisant face au cinéma. Par la suite, la transition fait référence à deux animations : la première active le cycle de marche du modèle de la fille et la deuxième permet de faire un déplacement de la fille afin de simuler la fille en train de marcher. Nous fixons l'animation de déplacement comme l'animation principale (major animation) pour permettre de synchroniser les autres animations en fonction de celle-ci. Finalement, dans la deuxième image clé, il y a une contrainte pour placer la fille devant le cinéma et l'autre pour qu'elle fasse face au cinéma. Donc, la fille est placée loin et en direction du cinéma et elle devrait finir juste devant le cinéma dans la même direction. Les positions intermédiaires seront calculées par interpolation par le système en fonction de la trajectoire spécifiée.

Le formalisme ou langage script que nous avons développé nous permet de faire abstraction de l'animation 3D et de représenter le sens d'une phrase simple. Ceci est une partie importante du projet, puisqu'elle nous permet de satisfaire le premier objectif spécifique de notre recherche.

2.1.3 Création des fichiers d'entrée du système

La dernière étape est de créer les fichiers d'entrée qui seront utilisés pour valider le système. Nous avons créé une série de fichiers XML pour représenter dix phrases avec le formalisme que nous avons développé. Ces fichiers sont la représentation de dix phrases qui permettront de créer dix animations qui garderont le sens de ces phrases. Nous avons créé d'autres phrases pour nos tests et pour le développement de notre système, mais nous avons basé notre validation sur ces dix fichiers. Les phrases pour lesquelles nous avons créé nos fichiers sont les suivantes :

1. John est sur le toit du cinéma.
2. John marche vers le cinéma.
3. John est début sur le banc du sauveteur.
4. John saute du plongeoir à la piscine.
5. John et Jane se lancent une balle dans le terrain de soccer.
6. Jane se dirige vers le cinéma.
7. Une pomme tombe de l'arbre.
8. John entre dans la piscine.
9. John regarde la piscine du plongeoir.
10. John entre au cinéma.

2.2 Analyse du format de la représentation 3D des animations

Une autre étape importante pour satisfaire notre deuxième objectif spécifique est d'analyser les différents formats de représentation 3D et déterminer le format le plus approprié dans le cadre de notre projet. Nous avons vu dans la littérature les différents formats de fichiers et les standards existants pour la représentation du contenu 3D. L'animation 3D que nous générerons devra être encodée dans le format de fichier plus adéquat. Nous ferons une analyse des formats existants afin de valider notre choix. Après avoir vu la littérature et parmi les principaux formats de représentation 3D, nous retrouvons les suivants : VRML, U3D, XAML, 3DXML, CITYGML, X3D et COLLADA. Pour notre projet, nous avons besoin d'un format riche dans la représentation 3D et qui soit supporté par la plupart des logiciels et outils 3D. Nous avons donc besoin d'un format avec les caractéristiques suivantes :

- Il doit être code ouvert et multiplateforme.
- Il doit être extensible et facile à manipuler.
- Il doit permettre de représenter les techniques de base et plus avancées d'infographie.
- Il doit être compatible avec les logiciels de modélisation et animation 3D les plus utilisés dans l'industrie comme 3ds Max, Maya, XSI, etc.
- Il doit être compatible avec les outils de visualisation 3D existants.

Nous analyserons les formats que nous avons présentés dans la littérature en fonction de nos besoins afin de bien déterminer le format le plus approprié pour notre projet.

VRML : C'est le prédécesseur de X3D. Il a été très populaire et très utilisé dans l'industrie vers la fin des années 90. Il est supporté par beaucoup de logiciels, mais toutes les améliorations ont été faites sur son successeur X3D, donc nous préférions utiliser X3D dans ce cas là.

U3D : C'est un standard utilisé surtout pour les données 3D utilisées dans les logiciels de CAO (conception assistée par ordinateur). C'est un format très simple, mais le principal problème est le support complet des animations 3D. Donc, nous jugeons que ce n'est pas un format approprié pour notre projet.

XAML : C'est un format spécifique à Windows, il n'est pas portable vers d'autres plateformes. Ce n'est donc pas un format qui nous intéresse.

3DXML : C'est un bon format qui se ressemble à X3D au niveau conceptuel. Il appartient à Dassault Systèmes et il est supporté uniquement par les produits développés par cette compagnie. Donc, il n'est pas dans notre intérêt d'utiliser ce format.

CITYGML : C'est un format spécialisé dans le domaine de la topographie. Il permet la représentation des objets 3D en milieu urbain. Il définit des relations pour les objets dans leurs propriétés géométriques, topologiques, sémantiques et d'apparence. Il est plus qu'un

simple format d'échange graphique, puisqu'il permet une analyse plus sophistiquée dans la simulation et l'exploration des données urbaines. Ce format, contrairement à U3D, est un format plus complexe avec une spécialité n'est pas exactement ce que nous recherchons.

X3D : C'est un format standard code ouvert spécialisé dans le rendu Web des graphiques 3D. Il est une version améliorée de VRML comme nous l'avons mentionné et il est devenu un standard 3D pour le Web. Il permet de représenter des environnements interactifs temps réel 3D et il supporte les techniques avancés d'infographie. Il est extensible et il existe plusieurs applications de modélisation et d'animation 3D qui le supportent. Pour ces raisons, ce format est donc un bon candidat pour les besoins de notre projet.

COLLADA : C'est un format de fichier ouvert pour les applications interactives 3D. Il est supporté par les plus grands logiciels de modélisation et d'animation 3D. Il supporte également les techniques avancées d'infographie. Il est facilement extensible et il a été conçu principalement pour être un format de contenu 3D intermédiaire ce qui est très avantageux pour notre projet.

D'après les principales caractéristiques que nous avons remarquées sur chaque format 3D, nous constatons que les formats X3D et COLLADA sont de bons candidats pour notre projet. Nous comparerons et analyserons ces deux formats plus en détail afin de choisir le format le plus approprié pour notre projet.

2.2.1 Comparaison entre COLLADA et X3D

Ces deux formats qui se sont développés en parallèle durant les dernières années nous permettent de représenter des données 3D codées en XML. Ils ont quelques similitudes, mais ils sont différents dans ses objectifs de conception et les usages prévus.

COLLADA se concentre principalement dans le pipeline de production qui permet d'utiliser du contenu 3D dans des applications 3D. C'est un format intermédiaire dont le but principal est de représenter du contenu riche 3D sur de multiples formes. De plus, il permet de transformer les données des applications de modélisation qui possèdent un haut niveau de description vers des applications qui ont besoin d'une description plus spécifique et optimisée.

X3D se concentre dans la visualisation de contenu 3D entre des applications. Il a été principalement conçu pour le Web. C'est un format de livraison destiné à contenir l'information nécessaire pour les applications interactives. X3D comporte un modèle spécifique qui permet la cueillette, la visualisation, la navigation, le scripting et qui fournit une interface de programmation pour manipuler le graphe de scène au moment de l'exécution.

La figure 2.13 montre la différence entre COLLADA et X3D. COLLADA est principalement intermédiaire, c'est le format de choix pour les outils de création de contenu numérique, pour les outils de conditionnement et l'archivage des données 3D. Tandis que X3D est le moyen

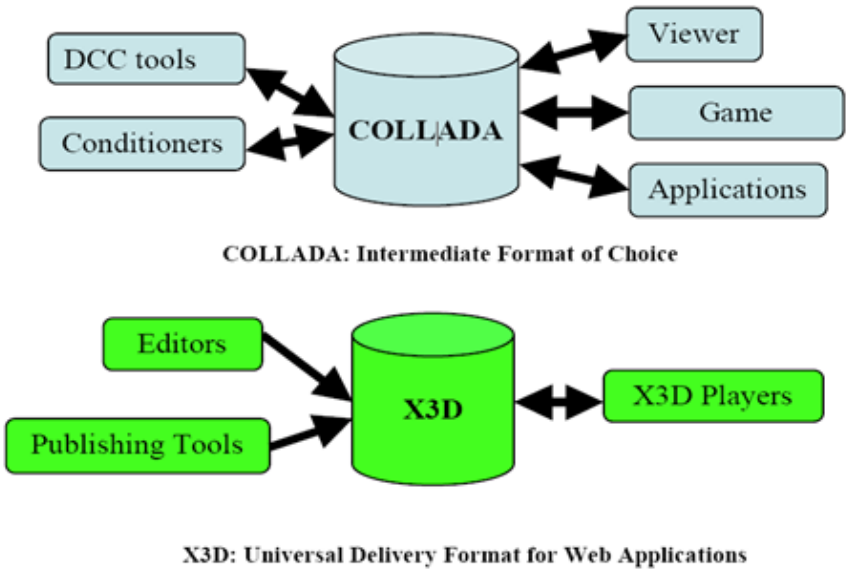

Figure 2.13 Collada vs X3D

de transmission des applications Web 3D interactives et des applications d'entreprises qui ont besoin une livraison en temps réel des données 3D.

X3D est utilisé plus tard dans le pipeline de production par rapport à COLLADA. X3D reprend là où COLLADA s'arrête, puisque COLLADA n'est pas défini à l'exécution contrairement à X3D. De plus, X3D est idéal pour le Web, puisqu'il possède une intégration complète avec des formats de données basés sur le Web.

Donc, pour les besoins de notre projet, les deux formats pourraient être utilisés, mais nous jugeons que COLLADA est le plus approprié pour notre projet. X3D possède plus de fonctionnalités que nous recherchons et c'est un format qui est plus orienté pour le Web. D'après cette analyse, nous pouvons dire que COLLADA est le format d'échange 3D le plus approprié pour notre projet ce qui nous permet de satisfaire le deuxième objectif spécifique de notre recherche.

2.3 Développement d'un système de création automatique des animations 3D COLLADA

Nous avons un formalisme de représentation 3D pour décrire des animations 3D et le format COLLADA pour représenter des animations 3D. Nous devons maintenant, d'après notre troisième objectif spécifique, transformer cette description vers l'animation finale en format COLLADA. Pour ce faire, nous avons développé un système capable de traduire le script qui décrit une phrase simple vers le fichier COLLADA qui contient l'animation 3D.

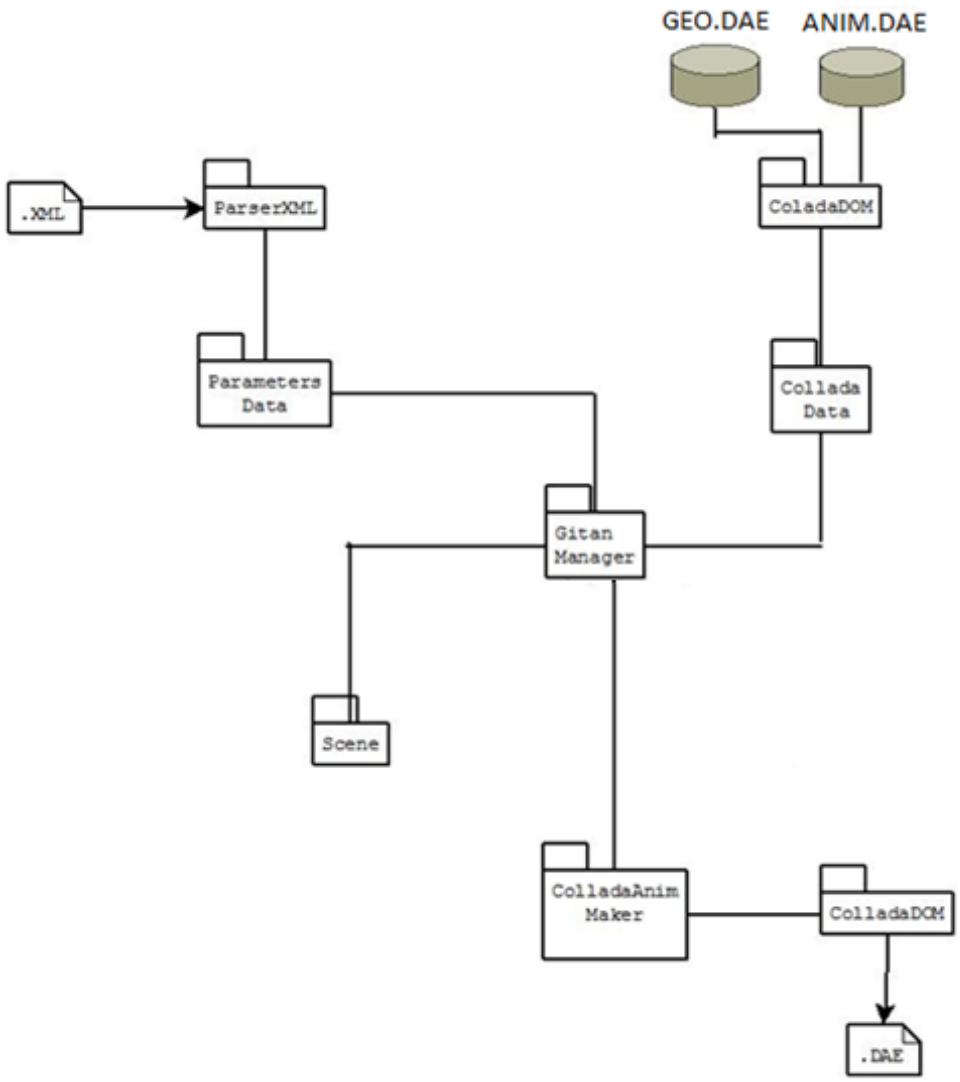

Figure 2.14 Architecture du Système

Dans cette section, nous présenterons le système que nous avons développé pour faire cette traduction. La figure 2.14 montre l'architecture du système.

Le système possède comme entrée trois éléments importants :

– le fichier XML avec la description de la scène,
– la base de données avec les géométries en format COLLADA et
– la base de données des animations prédéfinies 3D en format COLLADA.

À travers les différents modules, nous pouvons lire ces éléments d'entrée, nous traitons cette information pour le calcul des positions des objets dans la scène et nous traduisons cette nouvelle information vers le format COLLADA.

Nous expliquerons le traitement fait par chaque module et le raisonnement que nous avons

suivi pour la création du fichier COLLADA en sortie contenant notre scène animée 3D :

- module `ParserXML` :

Ce module a comme fonction de lire le contenu du fichier XML. Il nous permet de lire la description de la scène telle comme nous l'avons spécifié. Il utilise le parseur Xerces C++ pour parcourir le fichier XML et nous avons ajouté une couche de plus haut niveau pour faciliter l'extraction des éléments, des attributs et des paramètres existant dans le fichier XML. Son rôle est de faire l'extraction des données du fichier XML d'entrée.

- module `ColladaDOM` :

C'est une interface de programmation développée par Sony qui permet de lire et écrire sur un fichier COLLADA. Elle nous permet deux choses importantes : lire les géométries et les animations de la base de données et écrire le fichier COLLADA avec l'animation 3D finale. Son rôle est de lire les données des fichiers COLLADA d'entrée et d'écrire la scène résultante dans le fichier COLLADA de sortie.

- module `ParametersData` :

Ce module interagit avec le module ParserXML dans la lecture des données du fichier XML d'entrée. Ce module se charge de stocker toute l'information lue du fichier XML. Le module permet de sauvegarder les paramètres des nœuds, des animations, des contraintes et de la séquence des images clés qu'on utilise pour décrire l'animation 3D. Son rôle est simplement de stocker l'information sous forme des classes pour le traitement futur de cette information.

- module `ColladaData` :

Ce module se charge de deux choses importantes : l'extraction des données à partir des fichiers COLLADA d'entrée et le traitement de cette information pour le calcul des nouveaux paramètres nécessaires pour créer la scène 3D.

Premièrement, le module interagit avec ColladaDOM dans la lecture des géométries et des animations COLLADA à partir de la base de données. Il se charge d'extraire la position des sommets qui constituent la géométrie. Nous commencerons par expliquer l'extraction de ces données.

Pour expliquer clairement les données qui sont extraites du fichier COLLADA, nous devons comprendre la composition d'un fichier COLLADA. La figure 2.15 montre le contenu d'un fichier COLLADA en utilisant l'outil de visualisation XML Marker (LLC, 2010).

Le fichier COLLADA est composé d'une série de librairies qui permettent de définir la scène 3D. Parmi elles, nous retrouvons la librairie des géométries dont nous avons besoin. Par contre, lorsque nous déclarons une géométrie dans le fichier XML d'entrée, celle-ci est définie avec un identificateur qui ne correspond pas à l'identificateur de la géométrie en COLLADA, mais à l'identificateur d'un des nœuds de la libraire de scènes visuelles, puisque ce sont

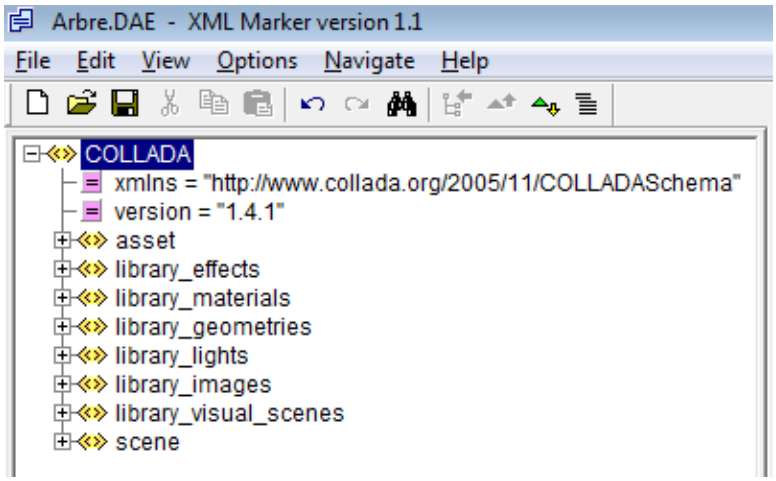

Figure 2.15 Librairies du fichier COLLADA

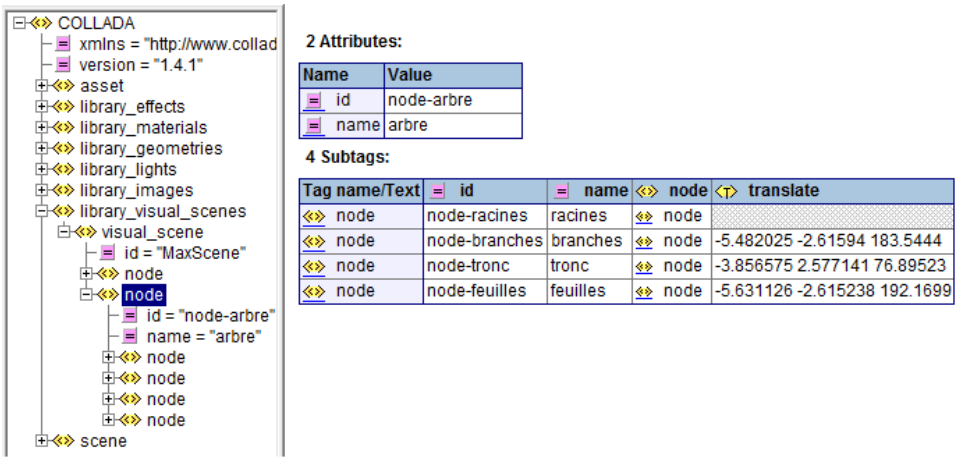

Figure 2.16 Contenu du nœud arbre dans le fichier COLLADA

ces nœuds qui contiennent la couche sémantique que nous avons définie précédemment. Par exemple, nous pouvons voir sur la figure 2.16 la composition du nœud « arbre » avec des nœuds enfants comme les racines, les branches, le tronc et les feuilles.

Avec cette méthodologie, nous pouvons manipuler n'importe laquelle sous-ensemble défini sur un modèle. Si nous désirons accéder seulement au tronc de l'arbre, comme nous l'avons spécifié précédemment dans le formalisme de représentation, il faut indiquer l'identificateur : « arbre.tronc ».

Par la suite, pour récupérer les géométries qui nous intéressent, il faut récupérer les instances de géométries contenues dans le nœud spécifié dans la scène visuelle. La figure 2.17 montre l'élément « instance geometry » contenu dans le nœud « node-tronc » qui nous permet accéder à la géométrie « geom-tronc » dans notre exemple.

```
<node id="node-tronc" name="tronc">
  <translate>-3.856575 2.577141 76.89523</translate>
  <node id="node-tronc_" name="tronc_">
    <translate>-11.91585 21.63097 -69.90898</translate>
    <rotate>0.388173 -0.3987943 0.8308338 -212.0745</rotate>
    <rotate>-0.07335842 0.9788784 0.1908284 -42.8815</rotate>
    <scale>0.6136841 0.6136837 0.4282382</scale>
    <rotate>-0.07335842 0.9788784 0.1908284 42.8815</rotate>
    <node>
      <matrix>1 0 0 201.2278 0 1 0 15.56237 0 0 1 246.3952 0 0 0 1</matrix>
      <instance_geometry url="#geom-tronc_">
        <bind_material>
          <technique_common>
            <instance_material symbol="Material__2" target="#Material__2-material">
              <bind_vertex_input semantic="CHANNEL1" input_semantic="TEXCOORD" input_set="1"/>
              <bind_vertex_input semantic="CHANNEL1" input_semantic="TEXCOORD" input_set="1"/>
            </instance_material>
          </technique_common>
        </bind_material>
      </instance_geometry>
    </node>
  </node>
</node>
```

Figure 2.17 Reférence vers la géométrie de l'arbre

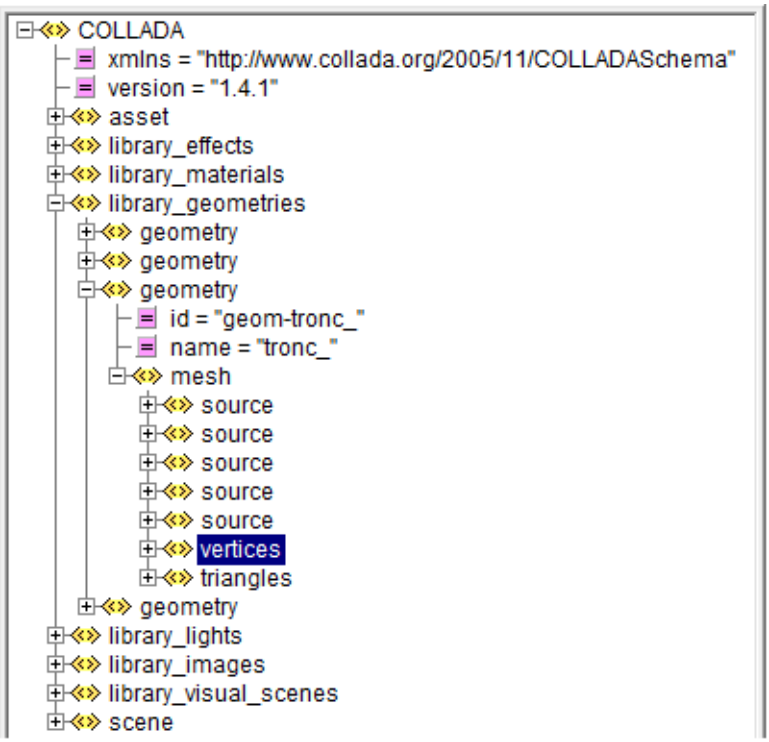

Figure 2.18 Sommets de la géométrie arbre

L'information qui nous intéresse de la géométrie, ce sont les positions des sommets des polygones qui constituent la géométrie. Donc, en utilisant l'URL de la géométrie, nous pouvons la rechercher dans librairie de géométries et stocker la position des sommets de celle-ci. La figure 2.18 montre la géométrie « geom-tronc » de notre exemple et les sommets que nous recherchons.

Par la suite, nous ferons le traitement des sommets afin de calculer des éléments importants pour la manipulation des géométries. Premièrement, nous calculerons les points extrêmes de toutes les géométries afin de calculer la boîte englobante de l'objet 3D. La boîte englobante nous permet de déterminer les points de référence de l'objet. Nous manipulons sept points de référence sur les modèles 3D : à gauche, à droite, en haut, en bas, devant, arrière et au centre. Nous avons déterminé à la création du modèle 3D que l'orientation de l'objet est toujours vers l'axe des Y positifs, donc en sachant les points de la boîte englobante, nous pouvons calculer les positions de référence de ces points. La figure 2.19 montre les positions sur le modèle de l'arbre. De cette façon, nous pouvons faire des relations spatiales avec d'autres objets en indiquant la position d'un objet par rapport à un autre.

De la même façon, nous extrairons les données des animations prédéfinies COLLADA. Ces animations nous permettent de représenter le cycle de marche de nos personnages comme nous l'avons expliqué précédemment.

Donc, l'animation COLLADA se trouve sous la librairie des animations *library_animations* et elle ne fait que modifier la position des géométries en fonction du temps. L'animation COLLADA est composée de trois éléments importants : les temps des images clés, les positions de sortie des géométries et le type d'interpolation appliqué. Ces données sont stockées et traitées dans ce module afin de récupérer ou modifier l'information à propos de l'animation que nous utiliserons dans la création de la scène animée. Alors, en traitant et en modifiant le temps des images clés, nous pouvons faire les actions suivantes :

- Déduire la durée de l'animation.
- Déduire le nombre d'images clés.
- Augmenter ou ralentir la vitesse d'exécution de l'animation.
- Couper l'animation à un temps précis.
- Étendre ou comprimer le temps d'animation.

- module `GitanManager` :

Ce module permet simplement la bonne communication et l'interaction entre les différents modules.

- module `Scene` :

Lorsque nous sommes rendus au module de la scène, nous avons déjà toutes les entrées prêtes pour la création de la scène animé 3D. Pour sa construction, nous nous basons sur

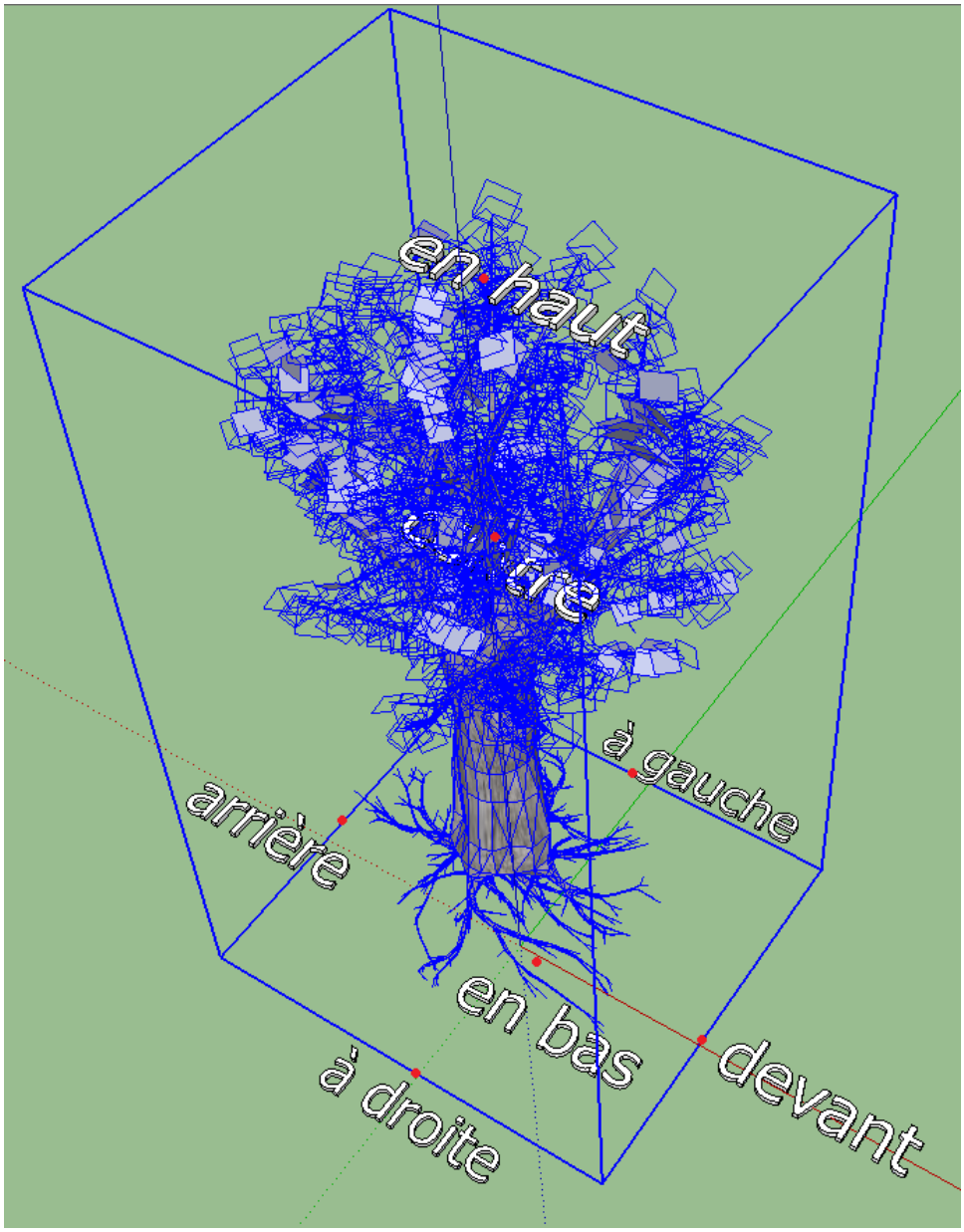

Figure 2.19 Points de référence sur le modèle

l'animation par images clés. Donc, ce module se charge principalement de calculer les positions des tous les objets de la scène à chaque image clé déclarée en appliquant les différentes contraintes possibles sur les objets.

Nous énumérerons maintenant les 6 étapes générales dans la construction de notre scène et par la suite nous détaillerons chaque étape afin de comprendre les algorithmes implantés dans le système.

1. Récupération de la position, de l'orientation et de la dimension initiale de tous les objets dans la scène si celles-ci ont été définies.
2. Application pour la première fois des contraintes de position.
3. Application des contraintes d'orientation.
4. Application des contraintes de dimensionnement.
5. Application des contraintes de position pour une deuxième fois.
6. Recommencer à l'étape 2 pour le reste d'images clés.

Dans la première étape, ce dont nous avons besoin comme information sont les transformations de base, c'est-à-dire la séquence de positions, d'orientations et de dimensions de tous les objets dans la scène pour chaque image clé. Pour ce faire, la première étape consiste à récupérer les transformations initiales existantes sur les objets déclarés dans la scène. Il est possible que certains objets possèdent une position, une orientation ou une dimension déjà déterminée. Ceci est possible si l'objet X appartient à un sous-ensemble de l'objet Y dans la scène. Donc, l'objet X aura déjà des transformations relatives en fonction d'Y.

Par exemple, si nous voulons représenter une personne qui se dirige au cinéma de la ville, il faut définir l'objet « ville » et l'objet « cinéma » (`id=ville.cinema`). Le modèle de la ville contient le cinéma et d'autres modèles qui sont déjà définis dans celle-ci comme nous pouvons voir à la figure 2.20. Donc, la ville nous permet d'avoir un environnement pour la scène et elle définit les transformations initiales des objets dans celle-ci. Pour cette raison, nous devons vérifier si les objets déclarés possèdent des transformations initiales pour fixer la position initiale des objets.

Par la suite, si l'objet est inanimé, les transformations rencontrées seront les mêmes pour chaque image clé, puisque ces objets resteront à la même position, orientation et taille pendant toute l'animation, vu qu'ils font seulement partie de l'environnement. Par opposition, les objets animés pourraient avoir des transformations différentes sur chaque image clé.

Pour la deuxième étape, il est nécessaire de calculer les transformations de déplacement pour chaque image clé dans la scène. Il faut appliquer les contraintes de position déclarées sur les objets. Pour appliquer ces contraintes, nous expliquerons premièrement l'algorithme que nous avons implanté pour décider l'ordre d'application des contraintes.

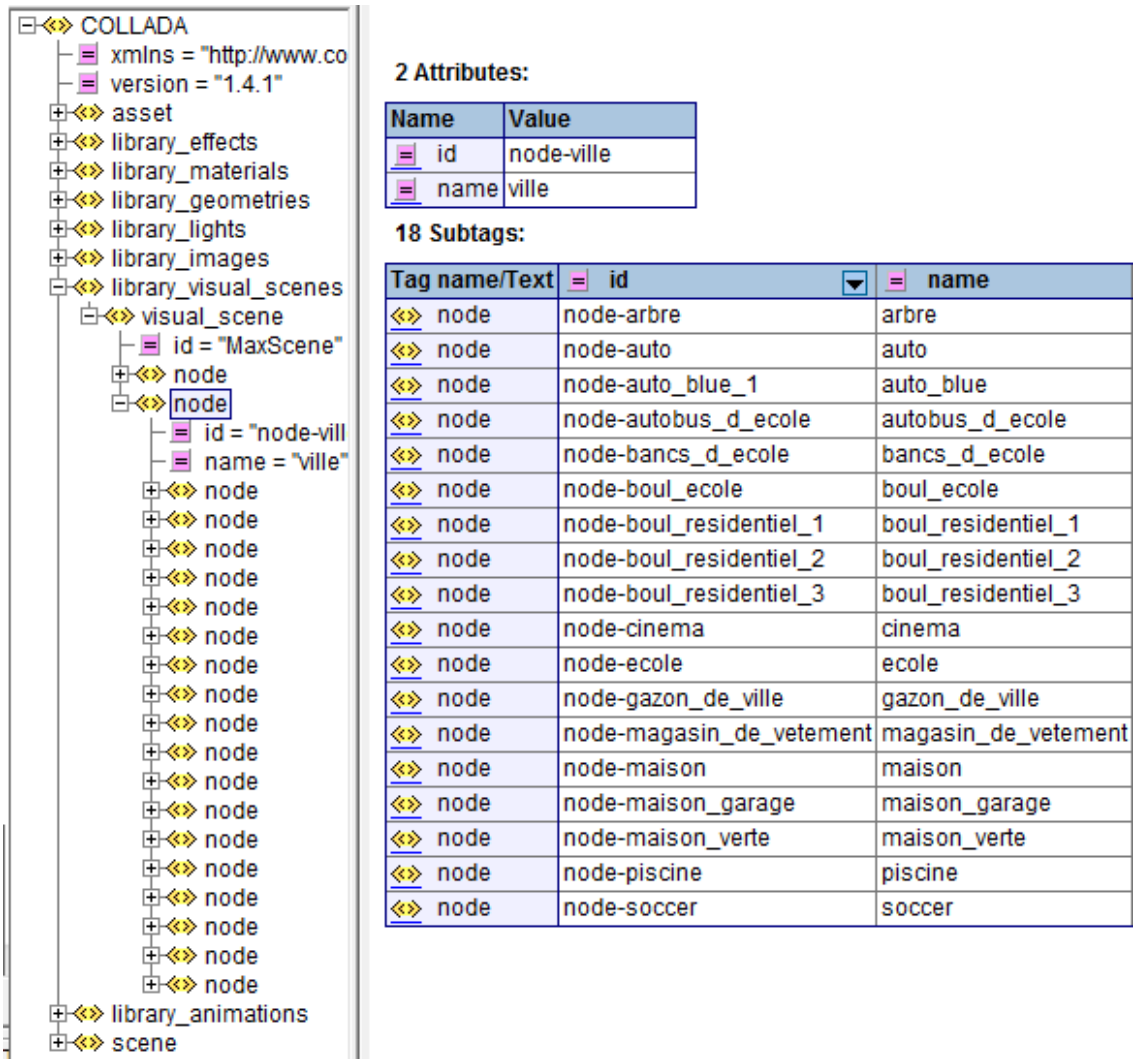

Figure 2.20 Nœuds de la ville

Comme nous l'avons expliqué, chaque contrainte possède comme attribut le nœud sur lequel on applique la contrainte et les nœuds de référence. De plus, pour appliquer une contrainte quelconque, il faut savoir la position de l'objet de référence afin de calculer la position de l'objet en fonction de celle-ci. Par exemple, la figure 2.21 montre des flèches qui représentent des contraintes qui partent de l'objet et qui sont dirigées vers des objets de référence. Donc, si la contrainte bleue fixe le nœud A en fonction de B et D, il est nécessaire d'appliquer les contraintes qui pourraient avoir B ou D en premier. Dans cet exemple, B a une autre contrainte en fonction de D, mais l'objet D n'a pas des contraintes, alors nous pouvons, dans ce cas, appliquer les contraintes qui ont D comme référence. Si les transformations de l'objet D ne sont pas initialisées, nous fixerons une position initiale par défaut à D à l'origine (0, 0, 0) et calculerons les positions des autres objets en fonction de celle-ci. Au besoin, nous

ferons de même pour l'objet C qui prend comme référence la position de A et B qui prend comme référence D.

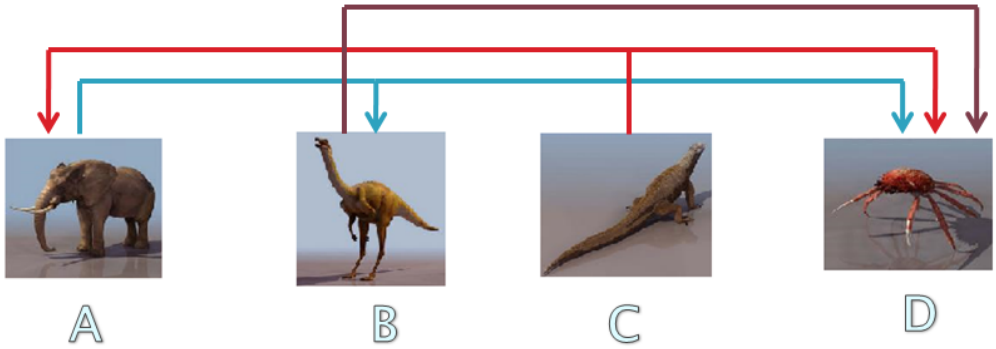

Figure 2.21 Contraintes sur les objets de la scène

Par la suite, lorsque nous connaissons la position du nœud de référence d'une contrainte, nous pouvons appliquer la contrainte de position sur l'objet.

La position d'un objet est calculée en fonction de trois vecteurs :

- Le vecteur position de la référence.
- Le vecteur décalage en fonction de la contrainte de position.
- Le vecteur distance en fonction de la contrainte de position.

Le premier vecteur est la position absolue de l'objet de référence. Si l'objet de référence possède un parent, sa position doit être additionnée à la position du parent pour avoir une position finale absolue. Par exemple, si l'objet de référence est le plongeoir d'une piscine, la position finale du plongeoir sera l'addition de sa position plus la position de la piscine, puisque la piscine pourrait être placée n'importe où dans la scène.

Le vecteur décalage est calculé en fonction du type de la contrainte de position. Les types de contraintes de position qui modifient le vecteur de décalage sont : *between*, *inFrontOf*, *behindOf*, *leftOf*, *rightOf*, *over*, *under* et *insideOf*. En fonction de ces types, nous fixons le vecteur décalage dont la valeur avait été calculée dans le module ColladaData comme nous l'avons expliqué précédemment à l'exception du type *between* dont la valeur est le point milieu entre deux objets de référence.

Le vecteur distance est également en fonction du type de la contrainte de position. Ce vecteur permet d'indiquer une distance qui sépare l'objet avec l'objet de référence. Les types possibles sont : *closeTo*, *awayFrom* et *atDistance*. Les deux premiers types possèdent le paramètre de pourcentage pour indiquer le pourcentage de distance qui indique si l'objet est

proche ou loin de l'objet référence. Le type *atDistance* possède le paramètre *distance* pour indiquer la distance précise en unités du monde virtuel qui sépare l'objet de la référence.

Finalement, lorsque les trois vecteurs sont calculés, la nouvelle position de l'objet est l'addition de ces trois vecteurs. Il est possible qu'il n'y ait pas de contrainte pour indiquer le vecteur de décalage, mais seulement des contraintes de distance. Dans ce cas précis, le décalage par défaut sera en fonction du type *infrontOf*.

La troisième étape consiste à appliquer les contraintes d'orientation. La procédure pour calculer l'ordre d'application des contraintes est la même que pour les contraintes de position. Il faut connaître l'orientation des objets de référence afin de calculer la nouvelle orientation des objets. Lorsque nous connaissons l'orientation de la référence, nous pouvons appliquer la contrainte d'orientation.

L'orientation d'un objet est déterminée par l'angle de rotation de l'objet et par l'axe de rotation. L'axe de rotation par défaut est Z. De plus, le calcul de cet angle dépend principalement du type de la contrainte d'orientation. Il existe quatre types d'orientation possibles : *toward*, *away*, *parallel* et *perpendicular*.

- Lorsque le type est *toward*, l'orientation de l'objet est dirigée vers l'objet de référence. L'angle de l'objet est donc l'angle de rotation du vecteur direction. Le vecteur direction est calculé de la façon suivante :
  `vecteurDirection = positionReference - positionObjet`
  Le calcul de ce vecteur nous oblige à appliquer les contraintes de position avant afin d'avoir les positions des objets.
  Pour calculer l'angle de ce vecteur, nous prendrons l'axe des Y comme référence et on utilise simplement le produit scalaire :
  `AB.BC = ||AB|| ||BC|| cos(ABC)`
  Alors :
  ⇒ `nouvelAngle = acos(vecteurDirectionUnitaire. (0, 1, 0))`
  La valeur de `nouvelAngle` devient l'angle de rotation de l'objet et il sera dirigé vers l'objet de référence.
- Lorsque le type d'orientation est *away*, l'orientation de l'objet sont vers le sens contraire de l'objet de référence. Donc, le calcul est le même que pour le type *away* sauf qu'il faut ajouter 180 degrés à l'angle de rotation de l'objet.
- Lorsque le type d'orientation est *parallel*, il faut simplement poser l'angle de rotation de l'objet égal à l'angle de rotation de l'objet de référence.
- Finalement, pour le type d'orientation *perpendicular*, l'angle de rotation de l'objet devient l'angle de rotation de la référence plus 90 degrés.

La quatrième étape consiste à l'application des contraintes de dimensionnement. La pro-

cédure pour déterminer l'ordre d'application de ce type de contrainte est la même que nous avons appliquée précédemment. La dimension de l'objet est déterminée par un vecteur de dimension fixé par défaut à (1,1,1). La nouvelle proportion de l'objet sera déterminée en fonction de l'objet de référence. De plus, la contrainte possède le paramètre *scale* qui contient la valeur de proportion en fonction de la référence. Si cette valeur est fixée à 1, l'objet aura la même dimension que l'objet de référence. Donc, la nouvelle dimension de l'objet dépend directement de la dimension de la référence et de la proportion passée comme paramètre.

Pour calculer cette nouvelle dimension, nous utiliserons la largeur de la boîte englobante de l'objet et celle de la référence. Alors, le calcul est le suivant :

```
newScale = scaling * refBox.largeur / objBox.largeur
```

Cette nouvelle proportion sera multipliée par le vecteur proportion de l'objet afin d'avoir le nouveau vecteur de dimension de l'objet.

La cinquième étape consiste à appliquer une deuxième fois les contraintes de positions, car l'application des contraintes d'orientation et de dimension pourrait avoir créé des collisions entre les objets. Le but de calculer les positions une nouvelle fois est d'éviter que les objets entrent en collision, puisque l'orientation et la dimension de certains objets auraient pu avoir changé. Par exemple, la figure 2.22 montre un bus devant le cinéma. Si nous déclarons une contrainte d'orientation pour indiquer que le bus est orienté vers le cinéma, il y aura une collision comme nous pouvons voir sur la figure. Pour cette raison, il est important de faire une deuxième passe des contraintes de position dans chaque image clé, puisque les transformations de rotation et de redimensionnement seront prises en compte dans le calcul de la nouvelle position.

Figure 2.22 Collision entre objets

Finalement, la dernière étape consiste à répéter les mêmes étapes d'application des contraintes

sur le reste d'images clés. Le calcul des transformations de position, de rotation et de redimensionnement est fait sur chaque image clé en utilisant la même procédure que pour la première image clé.

Après l'application de toutes ces étapes sur chaque image clé, nous connaissons les transformations de chaque objet à chaque image clé. Il est possible maintenant de créer l'animation 3D COLLADA au module suivant.

- module `ColladaAnimMaker` :

Ce module permet la création du fichier COLLADA avec les animations respectives. En effet, la dernière partie du système est la création de l'animation 3D et sa traduction vers le format COLLADA. Pour ce faire, nous utiliserons les transformations des objets afin de faire les calculs de synchronisation en fonction des types d'animation définis. Par la suite, on utilisera le module ColladaDOM pour écrire le fichier COLLADA contenant la scène et les animations créées.

Avant d'expliquer les étapes pour créer l'animation, nous expliquerons en quoi consiste une animation COLLADA pour comprendre l'information qu'il nous faut calculer. Les animations se trouvent dans la librairie d'animations comme nous pouvons voir à la figure 2.23.

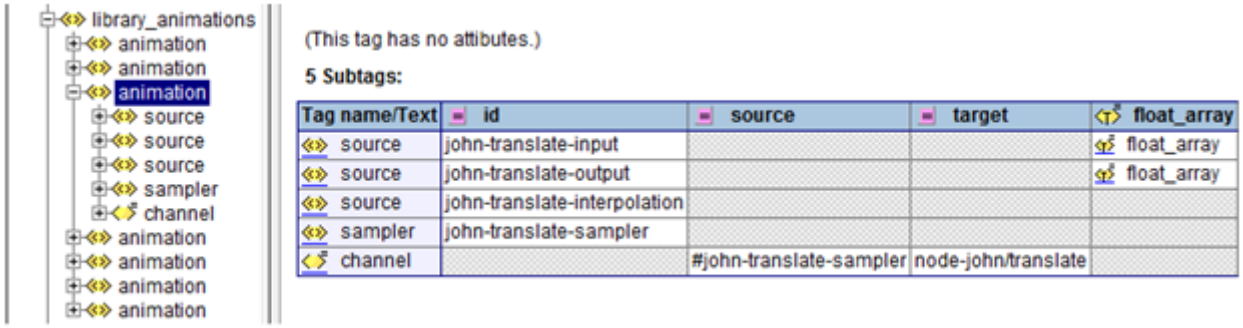

Figure 2.23 Librairie d'animations

L'animation COLLADA est composée de sources, d'un échantillon (*sampler*) et d'un canal (*channel*).

Il y a principalement trois sources qu'il faut définir pour créer une animation : la source des temps des images clés, la source des positions correspondant à chaque image clé et les types des interpolations à utiliser. La source des temps d'images clés sont les temps en secondes où chaque image clé sera affichée dans la scène. Donc, la première image clé est affichée à 0 seconde. La source des positions est simplement la liste des positions dans le monde virtuel d'un objet pour chaque image clé de l'animation. La source des interpolations est la liste des algorithmes d'interpolation utilisés pour passer d'une image clé à une autre. Les algorithmes

que COLLADA nous permet de spécifier sont les suivants : *LINEAR*, *BEZIER*, *BSPLINE* et *HERMITE*. L'échantillon permet simplement d'identifier et d'associer les types de sources avec celles déclarées. Finalement, le canal permet d'associer l'échantillon avec l'objet ou le nœud dans la scène sur lequel l'animation sera appliquée.

Nous expliquerons maintenant les différentes étapes pour bien comprendre la procédure pour générer nos animations. Donc, premièrement, il est nécessaire de calculer le temps nécessaire pour passer d'une image clé vers la suivante de façon à fixer le temps où chaque image clé est affichée. La durée de la transition entre les images clés dépend de la distance à parcourir par l'objet animé et de la vitesse à laquelle l'objet se déplace. C'est une simple équation physique :

$$temps = distance/vitesse \tag{2.1}$$

La distance est calculée en utilisant les transformations des objets, puisqu'elles nous permettent de savoir la position exacte de l'objet pour chaque image clé. La vitesse est donnée en entrée lorsqu'on déclare une animation de déplacement. Avec ces informations, nous pouvons calculer pour chaque objet animé les temps de transition en sachant que la première image clé est affichée au temps 0 seconde.

Deuxièmement, il faut faire la synchronisation des animations d'activité. Nous avons des cycles de marche et de cycles de course qui sont des animations COLLADA prédéfinies. Ces animations permettent de simuler l'action de marcher ou de courir de nos personnages. Des animations prédéfinies peuvent être utilisées pour des gestes, des postures, des expressions faciales, etc., mais nous avons seulement utilisé des animations pour simuler les actions de marche, de course et de saut. Ces animations doivent être synchronisées avec les temps des images clés. Dans la première étape, nous avons calculé le temps pour passer d'une image clé vers une autre. Cette deuxième étape permet de fixer la durée des animations d'activité. Par exemple, si un cycle de marche dure 1 seconde et nous savons que la deuxième image clé est au seconde 4, donc, il faudrait créer une activité de marche qui est composée par quatre cycles de marche de 0 à 4 secondes. Pour ce faire, il faut modifier les trois sources : les temps des images clés, la source de positions et la source des interpolations comme nous avons vu à la figure 2.23. Il faut donc reproduire le cycle de marche quatre fois pour la durée de la transition (4 secondes) pour simuler la marche d'un personnage.

La troisième étape consiste à créer les animations de déplacement. À l'étape 1, nous avons calculé les temps pour chaque image clé de chaque objet. Nous savons donc la position initiale et la position finale entre deux images clés. Par contre, il faut déterminer la trajectoire à suivre de chaque déplacement. Notre système nous permet de déterminer que des trajectoires linéaires et des trajectoires balistiques. Le type de trajectoire est défini lors de la déclaration d'une animation de déplacement. Dans le format COLLADA, la trajectoire est représentée

par le type d'interpolation entre deux images clés. Les types d'interpolation possibles sont : *Linéaire*, *Bézier*, *Hermite*, *Step* et *BSpline*. Par contre, le logiciel que nous utilisons pour jouer les animations COLLADA ne supporte que les interpolations linéaires. Donc, nous avons implémenté nous même la trajectoire balistique en utilisant la courbe de Bézier pour simuler des paraboles. Nous utilisons la courbe de Bézier de degré 1 en utilisant la position de chaque image clé. Si P_0 est la position initiale et P_1 est la position finale, alors la courbe de Bézier de degré 1 est définie par l'équation 2.2 :

$$B(t) = (1-t)P_0 + t(P_1), t = [0,1] \tag{2.2}$$

Chaque nouveau point de la courbe devient une nouvelle position dans l'animation de transition. Nous avons ainsi ajouté de nouvelles images clés intermédiaires entre les deux images clés qu'on veut interpoler. Ceci permettra de dessiner la trajectoire balistique que nous voulons définir.

La quatrième étape consiste à copier les données des fichiers COLLADA des objets déclarés comme entrée vers le fichier COLLADA de sortie qui contient l'animation COLLADA finale. Nous avons utilisé ColladaDOM pour manipuler les librairies à copier. Ces librairies contiennent principalement les objets, les géométries qui seront utilisées dans la scène et toute autre information utilisée par celles-ci, donc, les librairies à copier sont les suivantes : librairie des géométries, des nœuds, des matériaux, des effets, de contrôleurs et des images.

La cinquième étape consiste à copier les animations d'activité et de déplacement que nous avons modifiées à l'étape 2 et 3 respectivement vers le fichier COLLADA de sortie. Nous avons procédé de la même façon en utilisant ColladaDOM.

La dernière étape consiste simplement à sauvegarder le fichier COLLADA de sortie contenant l'animation finale à l'endroit spécifié par l'usager en utilisant ColladaDOM.

Toutes ces étapes permettent la création du fichier COLLADA dans ce module. Le fichier COLLADA de sortie sera lu par le logiciel de visualisation qui permettra de voir l'animation finale résultante. Ce système nous permet de lire le script d'entrée et de le traduire vers le fichier COLLADA qui contient la scène 3D. Nous pouvons donc dire que nous avons satisfait le troisième objectif spécifique de notre recherche.

2.4 Méthodologie de la validation des résultats

Nous n'avons pas développé des scènes très réalistes comme pour le cinéma ou pour les jeux vidéo, puisque nous jugeons que ce qui est important est de créer des animations simples qui soient capables de transmettre un message sans forcement avoir besoin d'une animation très réaliste ou d'un scénario très complexe. Afin de satisfaire le quatrième objectif spécifique

de notre recherche, une partie importante du projet est de valider si l'animation produite par le système est capable de transmettre un message et si le message transmis est correct. De plus, le système est capable de produire une scène animée ou une scène statique, donc il est important de valider que les scènes animées permettent de transporter le message mieux que les scènes statiques. Nous avons ainsi développé un sondage qui permettra de valider notre système et de vérifier le message compris par l'observateur.

2.4.1 Le sondage

Le sondage consiste à valider la construction correcte de la scène et la communication du message que nous voulons transmettre avec des scénarios statiques et dynamiques. Pour ce faire, nous avons développé divers scénarios en utilisant notre système. Ces animations seront vues par un échantillon de 30 personnes qui nous permettront de vérifier le pourcentage de personnes qui comprennent correctement le message transmis. L'échantillon correspond à des personnes de 15 à 65 ans. Chaque personne devra écrire, après avoir vu la scène, deux phrases pour décrire ce qu'il vient de voir. Une première phrase qui décrit exactement ce qu'ils regardent dans la scène sans faire de déductions et avec un point de vue objectif. L'autre phrase décrit la scène plutôt d'un point de vue subjectif, donc ils peuvent imaginer le scénario le plus logique et le plus probable d'après leur jugement. La première phrase nous permettra de valider la construction de la scène. Cette phrase étant objective, on s'attend qu'elle soit précise et correcte dans la plupart des cas. La deuxième phrase nous permet de valider la compréhension du message transmis et à quel point l'environnement de la scène peut influencer sur le message transmis vers l'observateur.

Par exemple, à la figure 2.24, nous avons une scène statique sur laquelle nous pouvons voir une scène 3D avec différents composants comme une piscine, une personne, une petite cabane, des chaises, etc. La première phrase raconte principalement l'état de la scène. Dans ce cas précis, la description serait : « Il y a une personne debout sur une chaise à côté de la piscine ». Par contre, pour la deuxième phrase, le message transmis par la scène n'est plus quelque chose d'objectif, puisqu'il faut déduire ce que la scène représente pour soi-même en tenant en compte tous les composants de la scène. Pour cette phrase, nous voulons savoir si la composition de tous les composants de la scène permet aux observateurs de déduire le message que nous voulons transmettre. Pour cet exemple, la réponse serait : « Une personne regarde la piscine debout sur une chaise ». La deuxième phrase décrit plutôt l'action de regarder ce qu'on recherche à transmettre comme message. De plus, certains scénarios serviront à valider l'influence de l'environnement de la scène dans la compréhension du message.

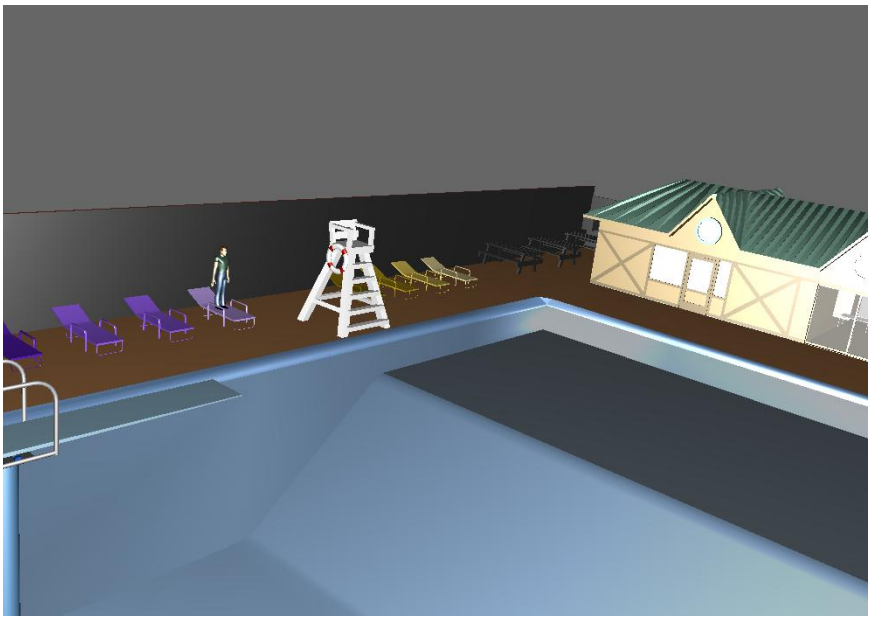

Figure 2.24 Exemple de scène statique

2.4.2 Les scénarios

Dix scénarios ont été construits avec notre système. Chaque scénario nous permettra d'analyser trois points importants :

1. La construction de la scène : ceci sera analysé avec la première phrase des scènes statiques, puisqu'on demande une phrase qui essaie de décrire l'état de la scène, l'observateur décrira ce qu'il juge important dans celle-ci. Si la scène est bien construite, l'observateur sera capable de distinguer les éléments importants de la scène et leur position.
2. La communication du message : ceci sera analysé avec la deuxième phrase, puisqu'on demande d'écrire une phrase subjective qui décrit ce qui se passe dans la scène, afin de vérifier si le message a été compris par l'observateur. De plus, nous pourrons analyser la différence entre les scènes statiques et les scènes animées dans la communication du message.
3. Influence de l'environnement : ceci sera analysé avec la deuxième phrase également. Nous avons généré des scènes qui transmettent le même message, mais le scénario n'est pas le même. Nous avons placé d'autres objets pour créer l'environnement de la scène et nous voulons vérifier si le message est transmis de la même façon pour les deux types de scène.

L'analyse de ces trois points nous permettra d'avoir une idée plus claire de la communication des messages par les scènes statiques ou animées. De plus, nous pourrons valider notre hypothèse et constater si les scènes animées permettent de communiquer le sens d'une phrase simple.

2.4.3 Les types de scénarios

1. **Scène statique simple sans environnement** : Pour la première scène, à la figure 2.25, nous montrons un cinéma et un personnage. Nous n'avons pas exposé d'autres éléments, parce que nous voulons montrer l'état de positionnement d'une personne qui est sur le toit d'un cinéma. Cette scène statique est simple et précise, puisqu'elle ne montre pas d'autres éléments que ceux qui sont pertinents et elle ne montre pas un d'environnement spécifique.

Figure 2.25 Scène 1

2. **Scène animée simple sans environnement** : La deuxième scène montre une personne qui marche en direction du cinéma. La figure 2.26 montre le premier et dernier cadre de l'animation. Cette scène nous permettra de comparer si le message « d'aller au cinéma » est plus facile à passer avec une scène qui n'a pas d'environnement ou avec une scène avec plus d'objets qui forment l'environnement comme à la scène 6 du sondage.

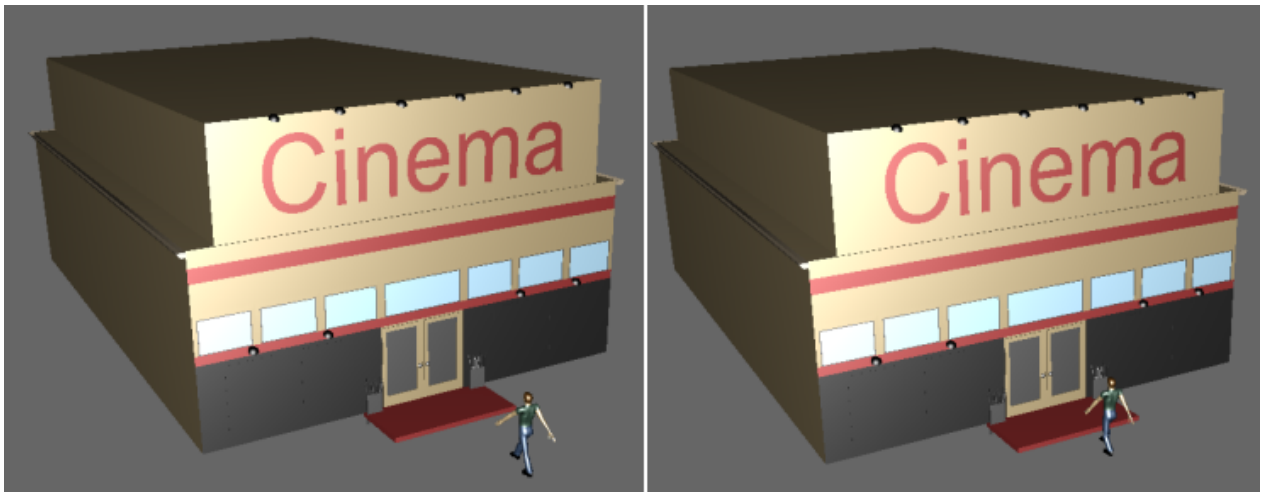

Figure 2.26 Scène 2

3. **Scène statique simple avec environnement** : La figure 2.27 montre la troisième scène. C'est une scène statique où il y a une personne debout sur le banc du sauveteur à côté d'une piscine. Dans cette scène, il y a plusieurs éléments visibles qui font seulement partie de l'environnement qui pourrait influencer la communication du message et qu'il soit mal compris.

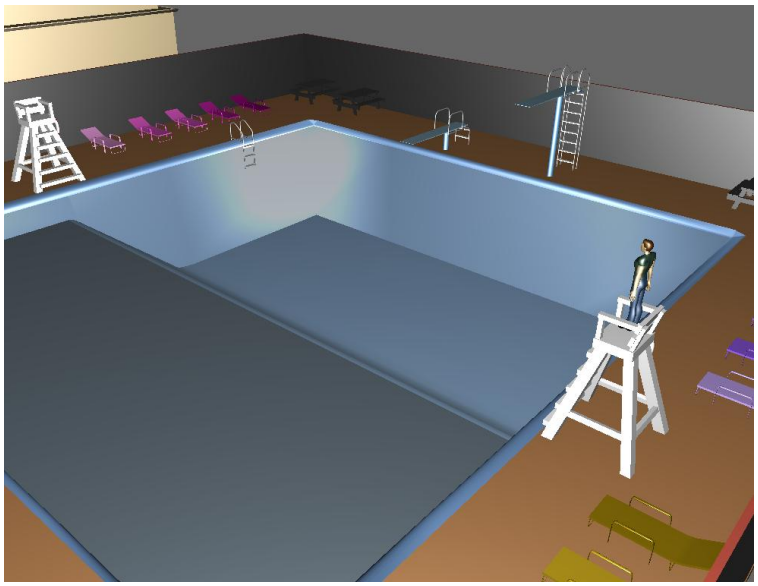

Figure 2.27 Scène 3

4. **Scène animée simple avec environnement** : La quatrième scène est une animation qui montre une personne en train de sauter d'un plongeoir vers une piscine. À la figure 2.28 nous pouvons voir les trois images qui montrent l'instant du début, du milieu et de la fin de l'animation.

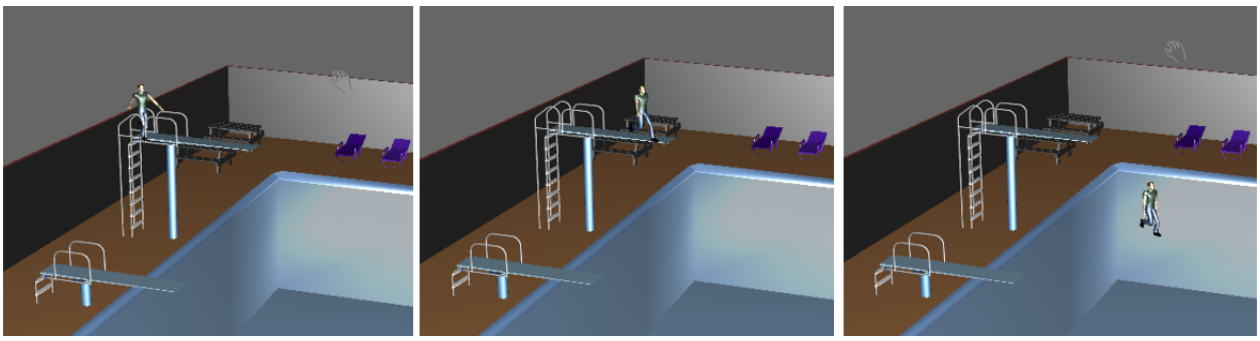

Figure 2.28 Scène 4

5. **Scène animée simple avec environnement** : Pour la cinquième scène, nous avons une animation montrant deux personnes en train de jouer avec une balle. La figure 2.29 montre trois cadres à différents moments de l'animation. Le message est simple, puisque nous voulons seulement représenter l'action de jouer avec un ballon dans l'environnement d'un terrain de soccer.

Figure 2.29 Scène 5

6. **Scène animée simple avec environnement** : La figure 2.30 montre deux cadres de notre sixième scène. Cette scène animée montre un cinéma à côté d'un boulevard où il y a des autos et une personne qui marche sur le trottoir en direction du cinéma. Cette scène nous permettra de valider si une scène qui contient plusieurs éléments est capable de passer le même message que la scène 2 qui montre seulement le cinéma et la personne.

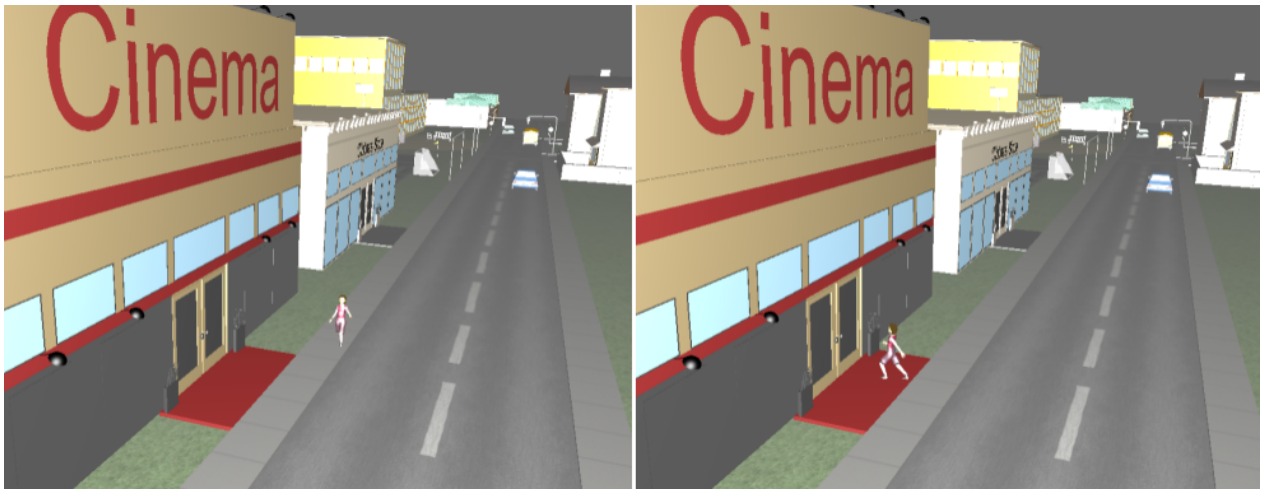

Figure 2.30 Scène 6

7. **Scène animée simple avec environnement** : La septième scène à la figure 2.31 montre une pomme qui tombe d'un arbre. La scène est toujours dans le contexte de la ville que nous avons construite, donc l'arbre est à côté d'une maison et de la piscine.

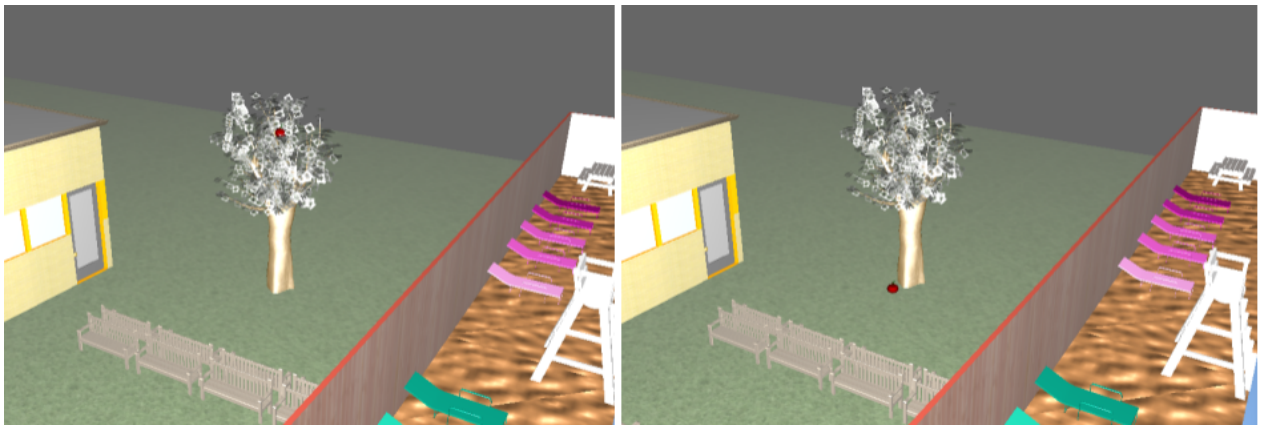

Figure 2.31 Scène 7

8. **Scène animée complexe avec environnement** : La huitième scène, à la figure 2.32, montre une personne qui entre dans une piscine et marche vers le centre. Cette scène animée n'est pas très simple puisqu'elle montre différentes actions qui se déroulent consécutivement. De plus, la scène présente un environnement de plusieurs éléments, l'interprétation du message pourrait donc prendre diverses significations à cause de cela.

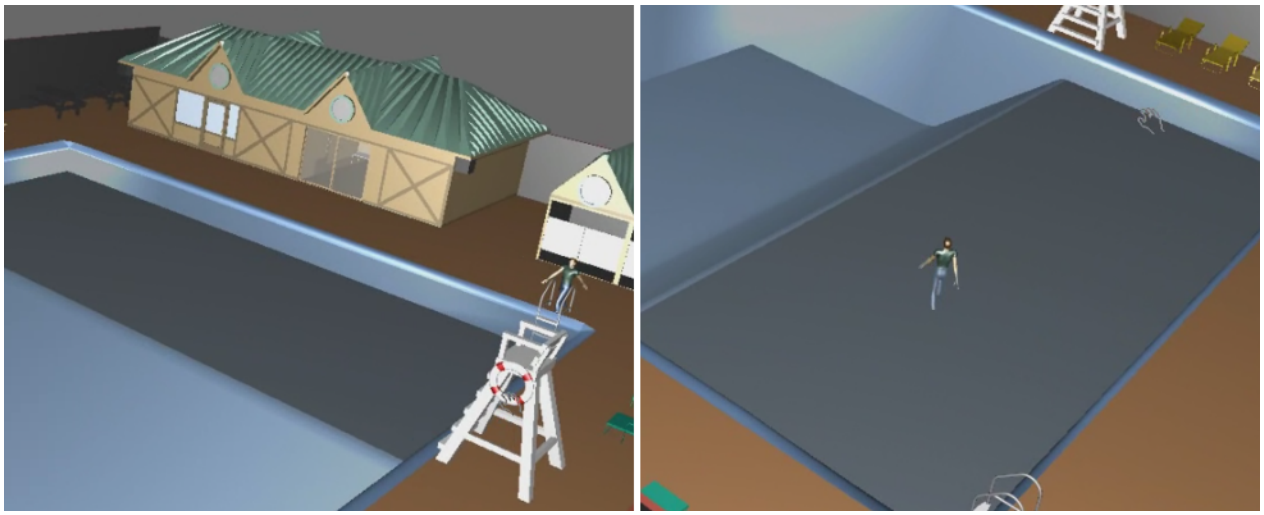

Figure 2.32 Scène 8

9. **Scène statique complexe avec environnement** : La figure 2.33 est la neuvième scène qui montre une personne debout sur un plongeoir devant une piscine. Cette scène statique nous permettra de vérifier si le message pour les scènes statiques est transmis correctement. Elle nous permettra de valider la capacité de communiquer des messages.

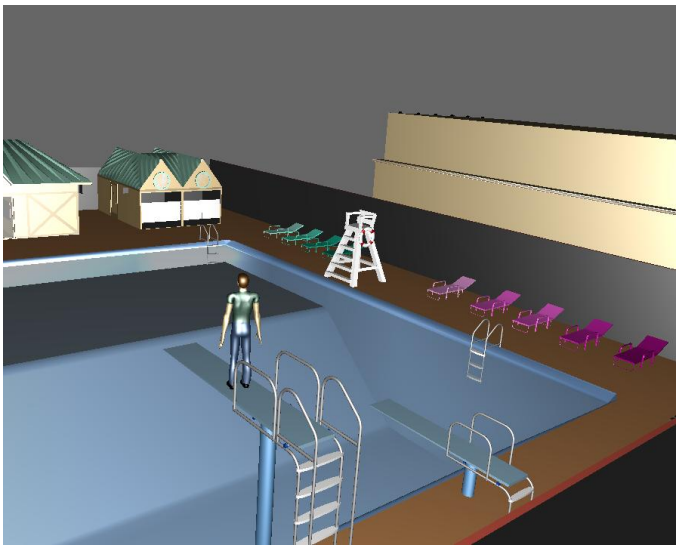

Figure 2.33 Scène 9

10. **Scène animée simple sans environnement** : La dernière scène à la figure 2.34 nous montre l'animation d'une personne qui marche en direction du cinéma et qui entre par la porte vers l'intérieur. Cette scène a été créée pour vérifier si dans la première phrase

l'action de « entrer » est comprise à différence des autres scènes où le personnage n'entre pas vers l'intérieur.

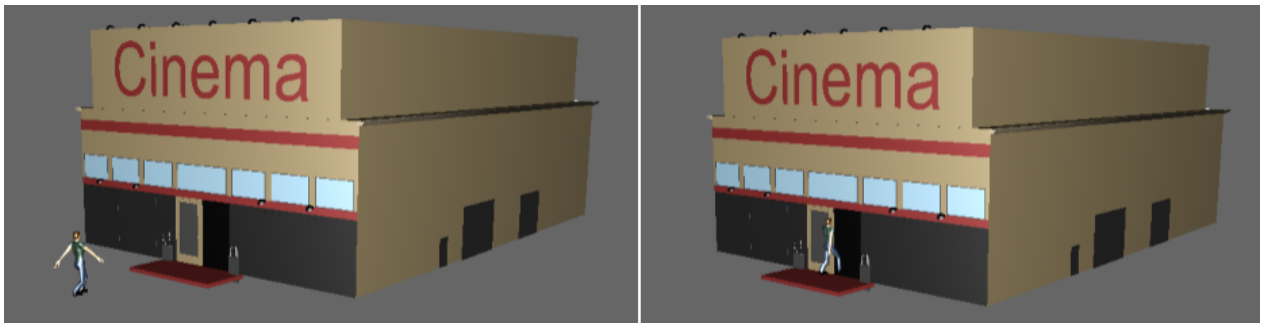

Figure 2.34 Scène 10

L'analyse des réponses du sondage nous permettra de valider notre système et nos hypothèses. Dans le chapitre suivant, nous verrons les résultats du sondage et leur analyse.

CHAPITRE 3

RÉSULTATS

Le système que nous avons développé nous permet de créer des scènes 3D statiques et dynamiques en utilisant le formalisme que nous avons implanté. Les scènes que nous avons obtenues ne prétendent pas être très réalistes comme dans les animations 3D des films ou des jeux vidéo. Notre objectif est de permettre de créer de façon simple et automatique des scènes 3D qui soient capables de transmettre des messages. La scène 3D doit montrer les éléments principaux de la scène et permettre aux observateurs d'identifier facilement ces éléments afin que l'animation soit interprétée correctement. Dans cette section, nous ferons la compilation des résultats que nous avons obtenus avec notre système. Nous présenterons la performance obtenue avec notre système, les fichiers de sortie COLLADA que notre système est capable de générer et finalement nous montrerons les résultats du sondage.

3.1 Performance du système

Le système permet de produire différents types de scène de complexité variée. La performance pour générer une scène 3D soit statique ou dynamique dépend principalement de sa complexité. Pour une scène d'environ 15 objets et 5 animations la génération prend 3 secondes approximativement sur un ordinateur Core Duo Centrino de 2.6 GHz et 3 Go de mémoire vive. L'optimisation du système n'est pas un aspect principal pour cette recherche, mais nous jugeons que le temps d'exécution pour générer des animations 3D est satisfaisant.

3.2 Fichiers COLLADA générés

Notre système permet de générer des fichiers COLLADA qui contiennent une scène 3D. Il y a deux types de scènes que le système permet de générer : des scènes statiques et des scènes animées. Les scènes statiques permettent de montrer l'état des objets dans la scène. Nous montrerons les résultats que nous obtenons pour générer chaque type de scène. Nous présenterons le script utilisé pour générer la scène et le fichier COLLADA généré.

3.2.1 Scènes statiques

Ce type de scènes permet de représenter des phrases qui contiennent des verbes d'état comme « être » ou « rester ». Par exemple, le script suivant nous permet de créer la scène

de la figure 3.1 pour montrer une personne devant le cinéma. La section `set_nodes` permet de déclarer les géométries pour le cinéma et pour le personnage « John ». La section `set_constraints` déclare les contraintes pour créer des relations spatiales entre les objets de la scène. Nous pouvons voir la déclaration de la contrainte *inFrontOf* appliquée au nœud « John » qui fait référence au nœud « cinema ». Cette contrainte positionne le personnage « John » devant le cinéma. Finalement, la section `keyframe` active la contrainte sur la seule image clé, puisque nous traitons une scène statique.

```
<set_nodes>
  <ground id="ground"/>
  <geometry id="cinema" scale="1.0" uri="C:\svn\cinema\cinema.dae"/>
  <geometry id="john" scale="1.0" uri="C:\svn\john\John_immobile.dae"/>
</set_nodes>
<set_constraints>
  <!-- The lists of constraints used by the sequence -->
  <position id="JohnInFrontOfCinema" node="john" reference="cinema"
    type="inFrontOf" worldProjection="horizontalPlane">
  </position>
</set_constraints>
<sequence >
  <!-- The sequence of keyframes.
       A keyframe is a list of constraints toggled on or off. -->
  <keyframe>
    <toggle activate="true" constraint="JohnOverCinema"/>
  </keyframe>
</sequence>
```

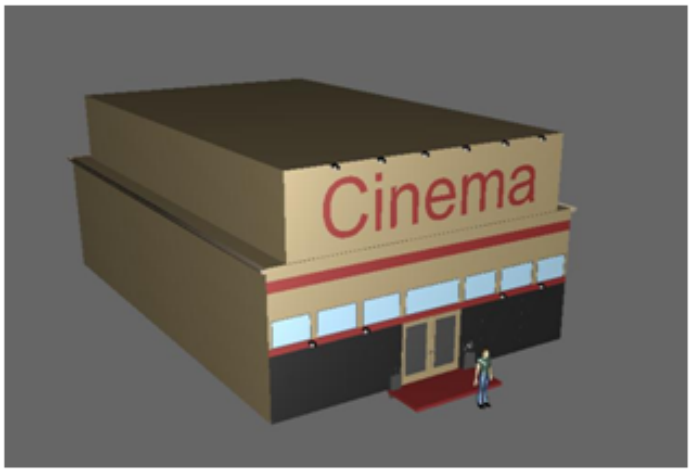

Figure 3.1 Exemple de scène Statique

Cette scène veut montrer une personne devant le cinéma. Nous n'avons pas besoin d'une scène animée pour cela, une simple scène statique est capable de montrer l'état des objets. Notre système permet de construire des scènes 3D en spécifiant des relations entre des objets semblables au projet WordsEye(Coyne et Sproat, 2001).

Le fichier COLLADA généré permet de représenter facilement des scènes statiques et des scènes animées. La figure 3.2 montre le contenu du fichier COLLADA généré par notre système qui représente la scène statique de la figure 3.1.

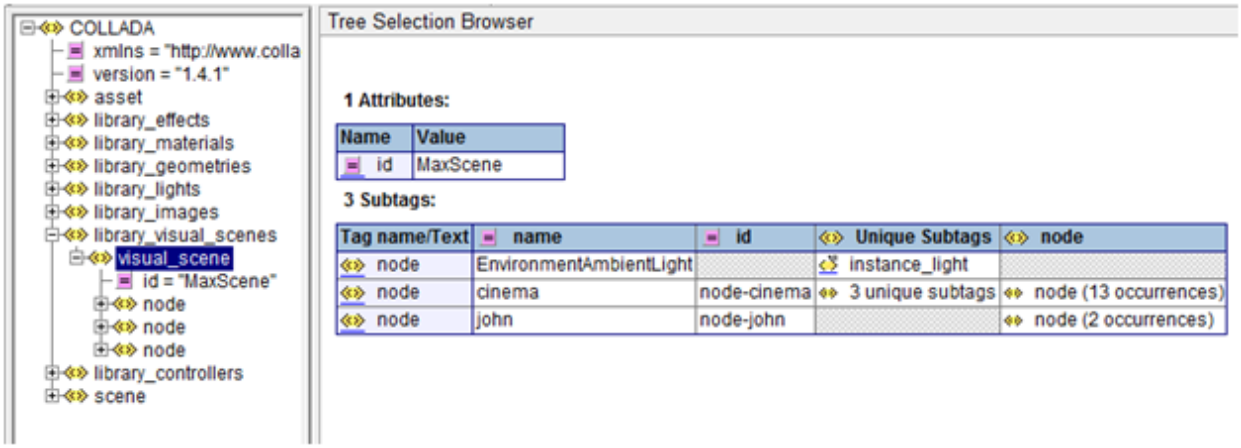

Figure 3.2 Contenu du fichier COLLADA

Nous voyons qu'il y a trois éléments dans la section `visual_scene` : le nœud « cinema » et le nœud « John » et le nœud « environnementAmbientLight ». La définition de ces objets est spécifiée à la construction de la scène. À la gauche dans la figure, nous pouvons voir toutes les librairies existantes où la librairie d'animations est absente, puisque c'est une scène statique. La figure 3.3 montre la librairie de géométries, dans cette image nous pouvons voir les géométries générées qui font partie du cinéma et du personnage « John ».

Si nous voulons exprimer l'état des objets dans une scène, il est facile de faire ceci avec une seule image clé, puisqu'il n'y a pas de mouvement. Par contre, pour représenter une action, une animation 3D est plus efficace pour communiquer le message.

3.2.2 Scènes animées

Si nous voulons exprimer une action, nous ne pouvons pas utiliser des scènes statiques, puisqu'une action demande normalement du mouvement dans la scène. Nous avons introduit des animations de déplacement et des activités pour exprimer des actions. Par exemple, le script suivant montre la description de la scène animée de la figure 2.26 utilisée dans le sondage.

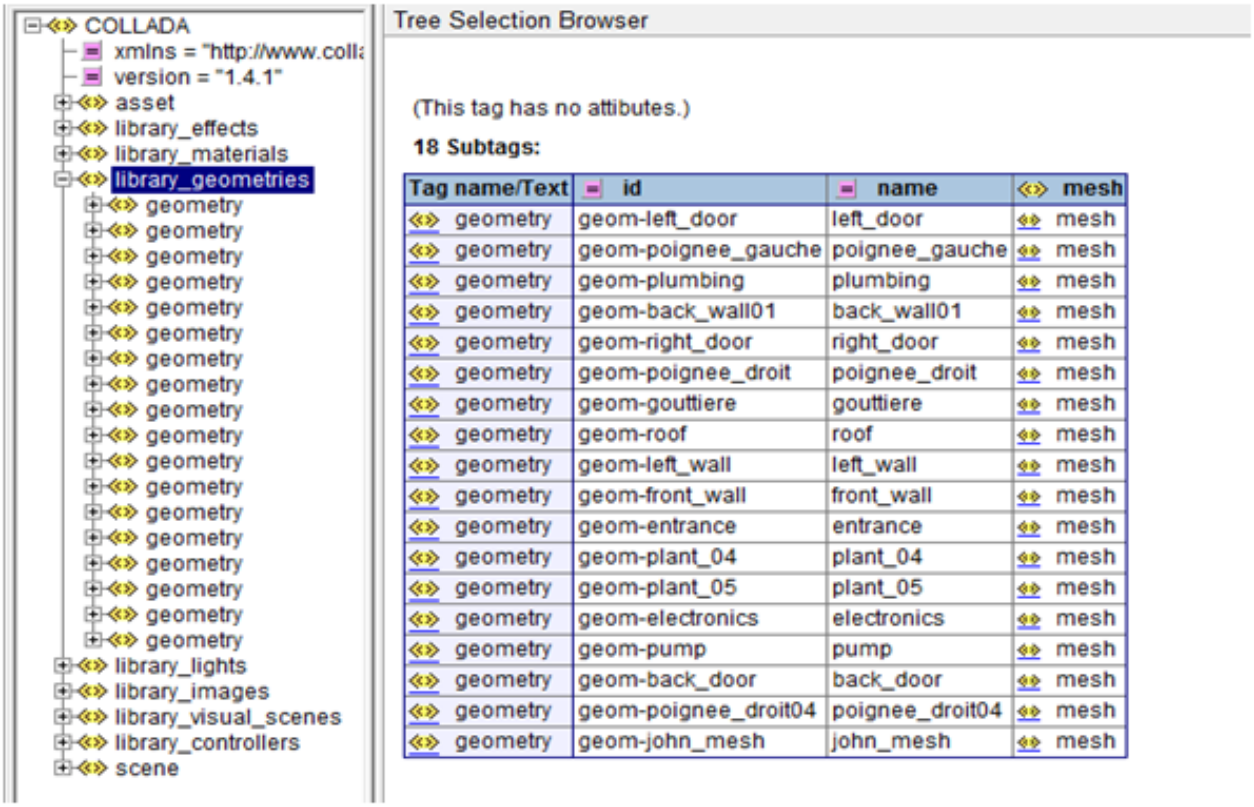

Figure 3.3 Geométries de la scène

```
<set_nodes>
  <geometry id="cinema" scale="1.0" uri="C:\svn\cinema\Cinema.dae"/>
  <geometry id="john" scale="1.0" uri="C:\svn\john\John_boucle_de_marche.dae"/>
</set_nodes>
<set_animations>
  <animation id="john-walk-cycle" looping="true" node="john" type="activity">
    <param name="uri">C:\svn\john\John_boucle_de_marche.dae</param>
  </animation>
  <animation id="johnTranslate" looping="false" node="john" type="displacement">
    <param name="trajectory">lineaire</param><param name="speed">40</param>
  </animation>
</set_animations>
<set_constraints>
  <position id="1johnInfrontCinema" node="john" reference="cinema"
    type="inFrontOf" worldProjection="horizontalPlane" />
  <position id="2johnFarFromCinema" node="john" reference="cinema"
    type="atDistance" worldProjection="horizontalPlane">
    <param name="distance">100</param>
```

```
    </position>
    <orientation id="johnTowardsCinema" node="john" reference="cinema"  type="toward"/>
  </set_constraints>
  <scene>    <node_ref node="john"/>     </scene>
  <sequence >
    <keyframe>
      <toggle activate="true" constraint="1johnInfrontCinema"/>
      <toggle activate="true" constraint="2johnFarFromCinema"/>
      <toggle activate="true" constraint="johnTowardsCinema"/>
      <transition major="johnTranslate">
        <animation_ref animation="john-walk-cycle"/>
        <animation_ref animation="johnTranslate"/>
        <keyframe>
          <toggle activate="true" constraint="1johnInfrontCinema"/>
          <toggle activate="true" constraint="johnTowardsCinema"/>
        </keyframe>
      </transition>
    </keyframe>
  </sequence>
```

Nous voyons dans le script que deux géométries sont définies, une pour le cinéma et une pour le personnage « John ». Par la suite, on déclare les animations d'activité et de déplacement pour déplacer le personnage. On déclare les contraintes de positionnement et d'orientation et, finalement, on déclare deux images clés qui déterminent la position initiale et la position finale de l'animation en activant les contraintes. Le premier cadre le personnage est loin du cinéma et dans le dernier cadre il est juste devant de celui-ci. La figure 2.26 montre ces deux instants de l'animation. Le personnage se déplace entre ces deux points avec un cycle de marche comme nous l'avons déclaré dans la transition entre les deux images clés.

Pour la génération d'une scène animée, il faut créer la libraire d'animations contenant les animations d'activité et de déplacement. À la figure 3.4, nous pouvons voir le fichier COLLADA généré correspondant à l'animation de la figure 2.26. Sur cette figure, nous pouvons voir les animations créées dans la librairie d'animations qui permettent de faire l'activité de marche et les cinq autres animations qui correspondent au déplacement, à la rotation sur les trois axes ainsi qu'au redimensionnement du personnage. Alors, d'après ces résultats nous pouvons constater que notre système est capable de traduire le script de description de la phrase vers l'animation 3D en format COLLADA, ce qui nous permet de valider notre deuxième hypothèse.

Les scènes 3D générées par notre système peuvent être statiques ou animées lorsque nous voulons communiquer un message. Pour valider le message transmis et satisfaire le quatrième objectif spécifique, nous avons réalisé un sondage comme nous l'avons expliqué précédemment.

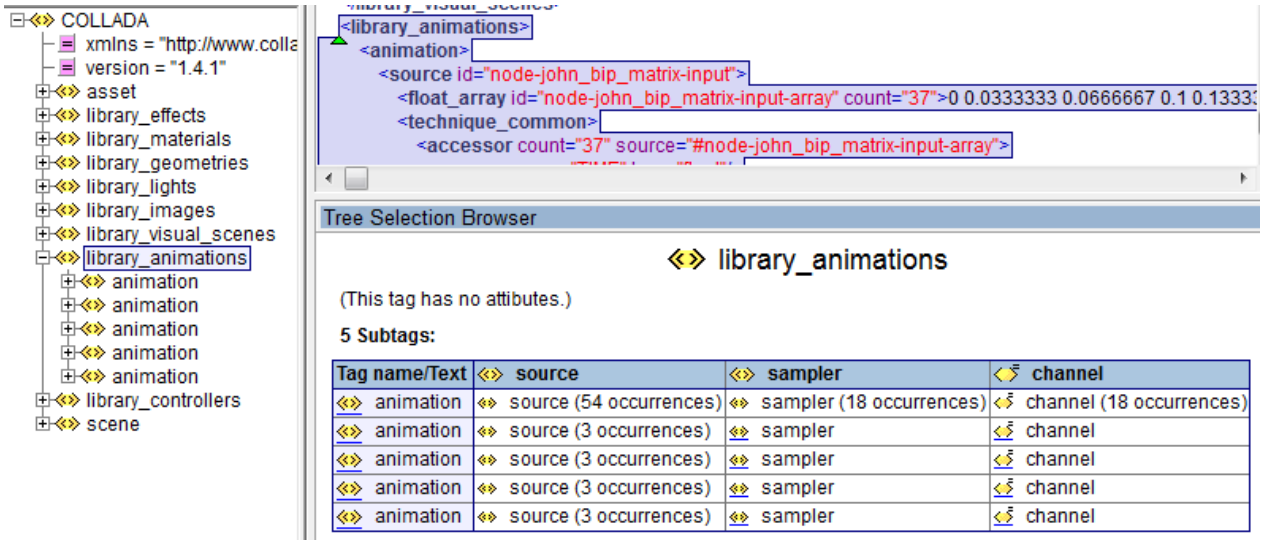

Figure 3.4 Animations générées

Dans la section suivante, nous montrerons le résultat attendu ainsi que le résultat du sondage.

3.3 Résultat attendu

Rappelons que le sondage que nous avons effectué nous permettra d'analyser trois points importants :

1. **La génération des scènes :** Nous jugerons si notre système est capable de construire correctement des scènes 3D de façon automatique. La construction de la scène sera analysée avec les réponses de la première phrase des scènes statiques. L'analyse de ces résultats nous permettra de vérifier que les observateurs sont capables de reconnaître les scènes et ses composants.
2. **Interprétation du message :** Nous jugerons si les scénarios produits par notre système sont capables de communiquer le message transmis par la scène 3D. Ceci sera analysé avec les réponses de la deuxième phrase. L'analyse de ces résultats nous permettra de constater à quel point les différentes scènes peuvent transmettre un message. De plus, nous analyserons si les scènes animées permettent une communication plus efficace du message que les scènes statiques et ainsi valider notre troisième hypothèse.
3. **Influence de l'environnement de la scène :** Nous jugerons si les scènes qui contiennent plusieurs éléments peuvent influencer la compréhension de la scène. Si les scènes sont construites avec un environnement de plusieurs objets, la communication du message pourrait être influencée soit positivement ou négativement. Pour analyser cet effet,

les scènes 2 et 6 du sondage transmettent le même message, mais avec des scénarios différents. Nous comparerons les résultats de ces deux types de scènes.

Les deux phrases attendues pour chaque scénario, que nous appellerons phrase A et phrase B, sont les suivantes :

1a : Un homme est sur le toit du cinéma.
1b : Un homme est debout sur le toit du cinéma.
2a : Un homme marche en direction vers le cinéma.
2b : Un homme s'en va au cinéma.
3a : Un homme est debout sur le banc du sauveteur.
3b : Un homme regarde la piscine debout sur un banc.
4a : Un homme saute du plongeoir vers une piscine en courant.
4b : Un homme saute vers une piscine du plongeoir.
5a : Deux personnes se lancent un ballon.
5b : Deux personnes jouent avec un ballon.
6a : Une fille marche vers le cinéma.
6b : Une fille s'en va au cinéma.
7a : Une pomme tombe de l'arbre.
7b : Une pomme tombe de l'arbre.
8a : Un homme entre et marche dans la piscine.
8b : Un homme va se baigner à la piscine.
9a : Un homme est debout sur le plongeoir devant une piscine.
9b : Un homme regarde la piscine du plongeoir.
10a : Un homme entre au cinéma.
10b : Un homme entre au cinéma.

Pour évaluer les réponses données par les observateurs, nous avons utilisé les consignes de correction suivantes pour les deux phrases :

- Phrase A

Les phrases A nous permettront de valider l'exactitude de la création de la scène. Ces phrases font une description de la scène très précise et elles permettent d'identifier les éléments importants de la scène. Donc, la phrase sera acceptée seulement si la description donnée identifie les éléments importants de la scène. Pour les scènes statiques, il est important de pouvoir identifier l'état des objets. Tandis que pour les scènes animées, c'est l'action qu'il faut identifier.

Par exemple, pour la première scène statique du sondage montrée à la figure 2.25, nous considérons une réponse correcte si elle fait référence au cinéma, à la personne et au fait que la personne est sur le cinéma. Exemple : « Il y a une personne qui se tient debout sur le toit

du cinéma ». Par contre, si un de ces éléments est manquant, nous considérons la réponse incorrecte. Exemple : « Il y a une personne debout et un cinéma ». La description mentionne les objets, mais ne pas l'état entre eux, elle est donc incorrecte.

Pour une scène animée, ce qui est important est d'identifier également les éléments principaux de la scène et les actions qui se déroulent. Par exemple, pour la scène 2 animée du sondage montrée à la figure 2.26. Une phrase correcte serait : « Une personne marche vers le cinéma ». Par contre, une réponse incorrecte serait : « Une personne marche devant le cinéma ». Pour certaines scènes, principalement les scènes animées, les phrases A et B peuvent être les mêmes ou très similaires. De plus, si les phrases données ajoutent d'autres adjectifs ou des compléments circonstanciels, nous considérons la phrase correcte si les éléments principaux mentionnés sont présents.

Pour l'analyse de la construction de la scène, nous n'utiliserons que les scènes statiques, puisque nous voulons valider si les observateurs sont capables d'identifier les objets de la scène et leur état. Les scènes animées représentent surtout des actions qui s'approchent plus du message transmis par la scène.

- Phrase B

Les phrases B peuvent varier en fonction du contexte ou de l'environnement de la scène. Ces phrases nous serviront à mesurer à quel point le message a été bien transmis par la scène générée, à comparer les scènes statiques des scènes animées et à vérifier si l'environnement peut influencer la compréhension du message. Nous considérons une réponse correcte si elle fait référence à l'événement principal et aux acteurs principaux de la scène. Par exemple, pour la scène 3 du sondage montrée à la figure 2.27, une réponse correcte serait : « l'homme regarde la piscine debout sur un banc ». Par contre, une réponse incorrecte serait : « une personne veut sauter du banc vers la piscine ». Ce qui nous désirons retrouver dans la réponse est principalement le verbe « regarder » ou autre verbe avec le même sens, et les éléments principaux de la scène qui sont le personnage et la piscine.

3.4 Résultats du sondage

Le tableau 3.1 montre les résultats du sondage pour nos dix scénarios. Sur celui-ci, nous pouvons voir pour chaque scène le nombre de réponses incorrectes, le nombre de réponses correctes et le pourcentage de réponses correctes pour chacune de phrases. Nous utiliserons ce tableau pour analyser et valider notre système.

Tableau 3.1 Résultats du sondage pour les deux phrases demandées

No Scénario	**Phrase A**			**Phrase B**		
	Incorrecte	**Correcte**	**%**	**Incorrecte**	**Correcte**	**%**
1	0	30	100.0%	19	11	34.0%
2	5	25	83.0%	2	28	93.0%
3	7	23	77.0%	12	18	60.0%
4	5	25	83.0%	6	24	80.0%
5	8	22	73.0%	4	26	87.0%
6	6	24	80.0%	5	25	83.0%
7	6	24	80.0%	4	26	87.0%
8	12	18	60.0%	11	19	63.0%
9	3	27	90.0%	27	3	10.0%
10	9	21	70.0%	6	24	80.0%
Total	**61**	**239**	**80.0%**	**96**	**204**	**68.0%**

3.5 Analyse des résultats

1. **Génération des scènes** : Pour valider que les scènes soient générées correctement, il faut analyser les résultats obtenus pour la première phrase. Si la scène est bien créée, l'observateur sera capable de décrire les aspects importants de la scène. Les pourcentages obtenus de la phrase A varient en fonction du type de scène. Si nous regardons les pourcentages les plus élevés, nous pouvons constater que ce sont les scènes statiques qui ont le pourcentage le plus élevé. Nous pouvons voir dans le tableau 3.2 les pourcentages de bonnes réponses pour les scènes statiques et les scènes animées.

Tableau 3.2 Description de l'état des scènes statiques et animées

Types de Scènes	**Phrase A**		
	Incorrecte	**Correcte**	**%**
Statiques	10	80	**89.0%**
Animées	51	159	**76.0%**

Nous avons obtenu 89% de bonnes réponses pour les scènes statiques et 76% pour les scènes animées. Nous pouvons donc dire qu'il semble plus facile de décrire l'état dans une scène statique que dans une scène animée. Les scènes statiques nous permettent de mieux juger la reconnaissance des éléments importants de la scène, puisque les phrases A ne font pas de référence à l'action qui se déroule, mais à l'état de la scène. Nous pouvons remarquer que les scènes statiques construites par notre système ont été reconnues à

89%.

2. **Communication du message** : Un deuxième point très important est de valider si les scènes produites par notre système sont capables de communiquer le sens d'une phrase simple. Si nous regardons les résultats, nous pouvons voir que les scènes qui ont les pourcentages les plus élevés ce sont les scènes animées. Le tableau 3.3 montre les résultats de la phrase B pour les scènes statiques et animées.

Tableau 3.3 Communication du message avec des scènes statiques et animées

Types de Scènes	**Phrase B**		
	Incorrecte	**Correcte**	**%**
Statiques	58	32	**35.5%**
Animées	38	172	**82.0%**

Dans ce tableau, nous pouvons voir clairement une grande différence entre ces deux types de scènes. Les scènes statiques semblent être moins efficaces pour communiquer de messages qu'une scène animée. Nous avons obtenu 82% de bonnes réponses pour les scènes animées, tandis que pour les scènes statiques nous avons obtenu que 35.5%. Ceci montre que les scènes animées permettent plus facilement de communiquer de l'information et à mieux comprendre le sens d'une phrase simple

En regardant les résultats, nous pouvons faire une remarque importante. Si bien les scènes statiques ont eu le pourcentage plus bas, par contre, pour la première phrase, elles ont eu le pourcentage plus élevé comme nous avons vu dans le tableau 3.2.

Donc, pour la représentation des états, les scènes statiques démontrent une supériorité à 89% de bonnes réponses, tandis que pour la représentation du sens d'une phrase simple les scènes animées sont beaucoup plus appropriées pour communiquer des messages à 82% de bonnes réponses.

Ces résultats nous permettent de constater que les scènes animées générées par notre système permettent de communiquer le message d'une phrase simple ce qui nous permet de valider notre troisième hypothèse.

3. **Environnement de la scène** : Les scènes 3D que nous avons générées sont composées par différents objets. Nous pensons que cette composition pourrait influencer la communication du message. Si la scène est très peuplée, elle pourrait changer le sens du message. Pour valider ceci, nous avons construit avec notre système deux types de scènes : une première contenant plusieurs objets et une autre ne contenant que les objets important de la scène. Nous avons fait cette comparaison pour deux scènes animées et pour deux scènes statiques.

Pour les scènes animées, la scène 2 est une scène qui possède deux objets, tandis que la scène 6 contient plusieurs objets pour recréer l'environnement d'une ville. Lorsqu'on les compare, nous pouvons voir dans le tableau 3.4 que la scène 2 a un pourcentage plus élevé de réponses correctes que la scène 6.

Tableau 3.4 Influence de l'environnement sur des scènes animées

Scène animée	**Phrase B**		
	Incorrecte	**Correcte**	**%**
Scène 2 (sans environnement)	2	28	**93.3%**
Scène 6 (avec environnement)	5	25	**83.3%**

Donc, nous pouvons voir dans les résultats que 10% du temps, l'environnement n'a pas permis aux personnes de bien comprendre le message que nous voulions transmettre, même si les deux scènes animées communiquaient exactement le même message.

Pour les scènes statiques, la scène 1 est une scène simple qui montre seulement deux objets contrairement à la scène 3 et à la scène 9 qui montrent l'environnement d'une piscine. Nous pouvons voir dans le tableau 3.5 la comparaison de ces scènes.

Tableau 3.5 Influence de l'environnement sur des scènes statiques

Scène Statique	**Phrase B**		
	Incorrecte	**Correcte**	**%**
Scène 1 (sans environnement)	19	11	**34.0%**
Scène 3 (avec environnement)	12	18	**60.0%**
Scène 9 (avec environnement)	27	3	**10.0%**

D'après ces résultats, nous ne pouvons pas affirmer que l'environnement à une influence sur la compréhension de la scène, puisque les résultats sont très variables. Cela peut être dû au fait que ce sont des scènes statiques et qu'il est déjà plus difficile à communiquer des messages avec ce type de scènes comme nous l'avons montré précédemment, donc il n'est pas possible de juger si le nombre d'objets dans la scène pourrait influencer la compréhension des messages d'une phrase simple sur les scènes statiques.

Note : Toutes les réponses données au sondage se trouvent à l'annexe B.

Conclusions de l'analyse

Les résultats montrent que les observateurs ont été capables d'identifier le message des scènes animées 46.5% de plus que les scènes statiques avec 82% (tableau 3.3) des scènes identifiées correctement. De plus, nous avons pu constater que les scènes statiques nous

permettent de communiquer des états plus facilement. Ceci est logique puisque les verbes d'état comme « être » ou « rester » sont très bien représentés par ces types de scènes. En effet, d'après les résultats, les scènes statiques que nous avons générées permettent de communiquer des états à 89% (tableau 3.2). Lorsque nous représentons un message, que ce soit avec des verbes d'action ou des verbes d'état, notre système est capable de construire une scène 3D qui reflète le sens de la phrase. En effet, en nous basant sur ces résultats, nous pouvons dire que si nous utilisons des scènes statiques ou des scènes animées au besoin, elles peuvent communiquer des messages à 85.5%. Si nous prenons en compte les moyens de communication existants, soit verbaux ou non verbaux, ce n'est jamais à 100% qu'on réussit à communiquer lorsqu'on parle ou lorsqu'on lit un texte. Il y a des facteurs qui existent, comme par exemple les différences de culture, qui ne permettent pas aux interlocuteurs de communiquer à 100%. Pour cette raison, nous jugeons que nous avons obtenu de très bons résultats et nos animations 3D sont effectivement capables de communiquer des messages. Finalement, le dernier point à soulever par rapport au sondage est au sujet de l'influence de l'environnement sur le message. Avec ces résultats, nous avons pu constater que la scène animée qui n'avait pas d'environnement permettait de communiquer 10% de mieux le sens du message(tableau 3.4). Ce n'est pas une grande différence, mais nous constatons qu'il y a des personnes qui sont influencées par l'environnement. Pour ce qui est des scènes statiques, les résultats sont très variés. De plus, vu qu'il est difficile de communiquer des messages avec ce type de scènes, nous ne pouvons pas repérer une influence de l'environnement. Nous avons donc constaté que l'environnement n'a apporté aucune amélioration à la communication du message. D'autre part, si l'environnement n'est pas généré dans la scène, la génération devient moins complexe. Donc, pour cette raison nous jugeons qu'il est préférable de construire les scènes sans environnement pour permettre une meilleure communication du message.

CHAPITRE 4

DISCUSSION

Ce chapitre discute des résultats que nous avons obtenus avec la méthode que nous avons proposée et énonce les différentes limites de notre méthode. De plus, nous discuterons de la méthode de validation proposée ainsi que de l'analyse du sondage et ce qui nous a permis de valider. Finalement, nous discuterons des systèmes semblables au nôtre, de ses caractéristiques et ses différences.

4.1 Sondage

Le sondage que nous avons réalisé nous a servi premièrement à avoir une rétroaction externe des animations générées pour les différents points que nous avons expliqués. De cette façon, nous pouvons au moins vérifier, pour l'échantillon que nous avons pris pour le sondage, que nous n'avons pas eu de résultats négatifs ou contradictoires. Au contraire, nous avons pu constater que les résultats étaient très encourageants. Deuxièment, il est important de remarquer que l'échantillon que nous avons pris ne nous permet pas de faire de conclusions statistiquement valides, car l'échantillonnage ne l'est pas scientifiquement. Par contre, les résultats obtenus nous permettent de constater qu'il est possible de faire une étude plus poussée de nos scènes générées avec un échantillon beaucoup plus grand et statistiquement valide.

4.2 Automatisation et l'aspect visuel de la scène 3D

Nous avons proposé un système simple, efficace et extensible pour générer des scènes animées de façon automatique. Notre solution propose une vue de haut niveau, afin de simplifier le processus de production de celles-ci. En effet, notre système faisait partie d'une recherche dans la conversion du texte vers l'animation 3D. Il a été conçu dans le but de créer des animations simples qui permettent de communiquer des messages. Nous ne nous sommes pas intéressés à produire des animations très réalistes comme dans les jeux de vidéo ou les films d'animation, ce qui a influencé les choix que nous avons faits. Nous nous sommes intéressés spécialement à l'automatisation de la génération des scènes 3D et, en même temps, qu'elles puissent communiquer le message désiré. Nous étions obligés à innover les techniques d'animation traditionnels et nous avons proposé un système capable de générer des animations 3D de façon automatique qui fait abstraction des difficultés de bas niveau qu'on retrouve

en infographie. Les modèles3D que nous retrouvons dans les scènes générées ont été développés par nous même en spécifiant une hiérarchie des nœuds spécifique afin de manipuler des concepts et de faire abstraction de la géométrie des objets, mais ce choix nous impose certaines limites dont nous discuterons plus tard. Les animations prédéfinies nous permettent principalement d'ajouter le réalisme minimal de la scène 3D, puisque l'idée principale est de communiquer un message et d'automatiser la génération de la scène. Les animations prédéfinies nous permettent d'ajouter le réalisme suffisant sur les actions qu'on appelle « activités » pour ajouter des mouvements humanoïdes à nos personnages lorsque nous avons besoin de simuler la marche ou la course par exemple. Les modèles et les animations prédéfinies ont été développés par un artiste 3D qui a créé une petite base de données simple suffisante pour valider nos hypothèses, puisque l'aspect visuel de la scène dépend beaucoup de l'automatisation. Par contre, il est possible de remplacer les animations prédéfinies et les modèles 3D par d'autres plus réalistes, mais ce n'est pas une priorité pour notre projet.

Un autre aspect important de l'automatisation est le schéma XML qui décrit la scène et son contenu. C'était notre premier objectif spécifique de bien définir le formalisme qui nous a permis de représenter le sens d'une phrase simple. Les langages script qui ont été développés par d'autres systèmes pour décrire une scène sont souvent très orientés vers le domaine d'application. Donc, nous avons jugé qu'il était plus intéressant de développer notre propre schéma que d'utiliser un autre déjà existant. Nous sommes très satisfaits du formalisme que nous avons développé, puisqu'il est très efficace et simple. Il peut être facilement extensible pour ajouter d'autres caractéristiques sur la scène ou sur les objets ce qui rend le système extensible. Nous avons utilisé une description par contrainte pour fixer la position, l'orientation et la taille des objets dans la scène. Par contre, les aspects comme le son, les gestes corporelles ou faciales n'ont pas été développés, puisqu'ils n'étaient pas pertinents pour cette recherche.

4.3 Fichier de sortie COLLADA

Le format de fichier de sortie qu'on génère pour représenter nos animations 3D est un aspect important et un de nos objectifs spécifiques. Notre système ne fait pas l'affichage de l'animation, il se charge uniquement de générer le fichier COLLADA. Le choix de ce format est indépendant des résultats que nous avons obtenus, même si le système génère uniquement la scène sous ce format. En effet, il est possible que si nous avions utilisé un autre format comme X3D nous aurions eu des résultats semblables. Le format à utiliser devient simplement comme le moule qu'il faut remplir avec la scène qu'on désire générer. Par conséquent, la simplicité du format détermine la complexité de notre système pour générer l'animation. C'est une

des raisons pour laquelle nous avons décidé d'utiliser COLLADA comme format de fichier intermédiaire. Il est assez simple, il est très complet, il est supporté par la plupart des logiciels de modélisation et de visualisation 3D, et il est bien connu de la communauté de l'infographie. Pour cette raison COLLADA semblait le plus indiqué dans tous les niveaux et nous sommes satisfaits des résultats obtenus avec ce format même s'il nous a imposé certaines limites dans la visualisation. Nous discuterons ceci plus en détail lorsque nous discuterons des limites du système.

4.4 Performance du système

Pour ce qui est du système qui nous permet de générer les scènes 3D, nous sommes très satisfaits de la performance, en prenant en compte que l'optimisation du système n'a pas été une priorité. Notre système génère une scène 3D de 15 objets en 3 secondes sur un ordinateur Core Duo Centrino de 2.6 GHz et 3 Go de mémoire vive. Les scènes générées sont très précises, par contre, le fichier XML qui décrit la scène est ardu à créer pour une scène complexe de plusieurs objets. Nous n'avons pas créé un système d'édition pour générer ce fichier, puisque ce n'est pas une priorité pour cette recherche, mais nous pensons qu'il aurait été intéressant d'avoir dans le système un module permettant de générer ce fichier qui représente la scène en manipulant les objets et les contraintes qui la décrivent.

4.5 Limites du système

Nous avons été capables de transformer une description de la scène avec le formalisme créé vers une scène animée 3D sous format COLLADA qui permet de communiquer le message d'une phrase simple. Par contre, le système comporte certaines limites que nous avons identifiées. Les limites rencontrées se trouvent dans le fichier de description de la scène, dans les modèles 3D et dans les logiciels de visualisation. En effet, le script de description de la scène est créé manuellement ce qui peut être très laborieux lorsqu'il y a plusieurs scènes ou lorsqu'elle est trop complexe. Ce fichier devait être créé par un autre module du projet GITAN lorsque la phrase textuelle est traitée, donc nous n'avons pas développé d'éditeur pour générer ce fichier ce qui aurait été très intéressant pour générer nos scènes plus rapidement indépendamment de GITAN. De plus, les modèles 3D que nous avons développés nous limitent en matière des animations 3D qu'on peut représenter. En effet, la base de données des modèles 3D et d'animations prédéfinies n'est pas très grande. Par contre, les scénarios que nous sommes capables de générer sont suffisants pour valider nos hypothèses et le principe de notre recherche. De plus, nos modèles 3D sont construits en respectant une hiérarchie des nœuds prédéfinie qui ajoute une couche sémantique aux modèles 3D ce qui ne nous permet

pas d'utiliser les modèles qu'on rencontre sur Internet. Cependant, notre système supporte les modèles de Google SketchUp si la couche sémantique est ajoutée manuellement. Finalement, les limites existantes en matière des visualiseurs COLLADA ne nous permettent pas une libre visualisation des animations 3D. En effet, COLLADA est un format d'échange 3D, il est très flexible et il nous permet d'ajouter des spécifications propres à l'application à l'aide des éléments « Extra ». Cette flexibilité fait que les différents systèmes qui l'utilisent ne le fassent pas de la même façon ce qui peut parfois causer des conflits. Alors, vu que nous n'avons pas développé notre propre visualiseur, nous avons utilisé « Scenix Viewer 6.0 » pour la visualisation de nos scènes 3D. Ce visualiseur nous permet de voir une scène complexe de plusieurs objets correctement comparés à d'autres systèmes, par contre il nous limite sur ce qu'on peut animer, puisque le visualiseur ne peut pas afficher plusieurs activités ou animations prédéfinies en même temps. Ceci n'est pas très grave, puisque pour les besoins du projet, les animations ne sont pas trop complexes, mais si nous désirons créer des animations plus complexes, le visualiseur ne nous le permet pas.

4.6 Autres systèmes de génération des scènes 3D

Finalement, nous discuterons des systèmes semblables au nôtre dans la génération automatique des scènes 3D. Ces types de systèmes sont souvent orientés à un domaine d'application spécifique comme CAMEO qui fait des animations pour la cinématographie immersive. Nous avons vu aussi BEHAVIOR3D qui permet de créer des comportements en ajoutant des nœuds prédéfinis qui ont été ajoutés dans le standard X3D. Ce type de système permet de créer des animations automatiquement en utilisant X3D pour prédéfinir des nœuds qui permettent d'animer des objets en fonction du comportement assigné. De plus, grâce à cette façon de faire de BEHAVIOR3D, il permet de définir des interactions avec les usagers et la possibilité de créer des animations sur le Web. Donc, les objectifs de ce projet ne sont pas les mêmes que les nôtres, puisqu'ils s'intéressent surtout au comportement des objets. D'autre part, BEHAVIOR3D est plus orienté pour le Web, puisqu'il utilise X3D pour représenter ses animations. Par contre, nous nous sommes inspirés de ce projet, parce que nous nous sommes basés sur COLLADA pour représenter nos animations 3D, mais dans un autre contexte et avec d'autres objectifs. L'adaptation de COLLADA a facilité la génération des animations et la création de la scène 3D. Également, nous avons analysé d'autres systèmes avec des objectifs semblables et capables de générer des scènes 3D de façon automatique comme WorldEye et PUT. En effet, WorldEye est un système qui permet de traiter du texte descriptif pour le représenter avec une scène 3D statique. Il possède une riche base de données de modèles 3D, mais il utilise des étiquettes sur l'objet lui-même pour déterminer le devant, l'arrière, le haut et tous les autres

points de référence de l'objet qui permettent de faire de relations spatiales entre eux. Notre système ne possède pas une grande base de données de modèles. Par contre, les points de référence sont calculés automatiquement à la création de la scène. D'autre part, leur système utilise des postures prédéfinies pour donner des actions aux personnages. Par exemple, des positions pour saluer, courir, marcher, nager, etc., sont des postures que peuvent adopter les personnages pour simuler une action. En ce qui nous concerne, nous utilisons des animations prédéfinies pour ajouter des activités aux personnages pour représenter les actions. WorldEye est un système plus mature qui a misé sur des scènes statiques avec de postures prédéfinies, contrairement aux scènes animées que nous utilisons. Pour ce qui est du système PUT, il est très semblable à WorldEye, puisqu'il recherche à générer des scènes statiques en utilisant le langage naturel, mais il se concentre sur le placement des objets et le calcul des zones de contact entre eux.

CHAPITRE 5

CONCLUSION

Ce mémoire a proposé un système capable de générer des animations 3D de façon automatique qui représentent le sens d'une phrase simple. Le système que nous proposons utilise les concepts et les actions qu'on retrouve dans une phrase pour générer une scène animée. Ceci permet de faire abstraction de la procédure de création de l'animation 3D traditionnelle, puisqu'elle peut devenir très complexe spécialement pour les objectifs de ce projet. La solution que nous avons proposée nous permet de générer des animations plus facilement et de communiquer des messages aux observateurs. La méthodologie que nous avons retenue nous a permis de bien cerner les difficultés que nous devions surmonter pour la génération de l'animation 3D, l'automatisation de cette génération et la validation du message transmis afin de satisfaire nos objectifs.

Le premier objectif spécifique était de déterminer le formalisme à utiliser pour décrire la scène animée de façon à faire abstraction de l'animation 3D traditionnelle. La section 2.1 de la méthodologie montre les différents éléments du formalisme que nous avons implanté pour décrire les animations 3D à générer. Ce formalisme nous permet de déclarer des objets et de faire des relations spatiales entre eux en utilisant des contraintes. Nous utilisons l'animation par image clé pour représenter nos animations, alors une séquence d'images clés peut être déclarée pour générer une scène animée. Le formalisme permet de spécifier les composants qui définissent l'animation en fonction de noeuds, de contraintes, d'activités, de déplacements et d'images clés. Le système n'utilise ainsi que la description de la scène basée sur ce formalisme pour générer automatiquement l'animation 3D ce qui atteste que cet objectif a été atteint.

Le deuxième objectif spécifique était de déterminer le format d'échange 3D le plus approprié pour représenter le sens d'une phrase sous la forme d'une animation 3D. La section 2.2 de la méthodologie fait l'analyse des différents formats de représentation 3D. Nous avons discuté brièvement des caractéristiques principales de quelques-uns d'entre eux et nous avons comparé les deux formats, X3D et COLLADA, qui étaient les meilleurs candidats pour représenter nos animations. Nous avons jugé qu'il était possible d'utiliser soit un ou l'autre, mais le plus approprié a été COLLADA. Les animations 3D que notre système a pu générer sous format COLLADA attestent que cet objectif a été satisfait correctement.

Le troisième objectif spécifique consistait à créer le système logiciel qui permet de traduire la description de la scène en utilisant notre formalisme vers l'animation 3D sous format COLLADA. Nous avons présenté l'architecture du système et les différents modules qui la

composent à la section 2.3 de la méthodologie. Ces modules permettent de lire le fichier XML contenant la description de la scène, ils se chargent de lire les modèles 3D et les animations prédéfinies de la base de données, et ils permettent de traiter toute cette information pour générer la scène animée de façon automatique sous format COLLADA. De plus, les différents modules utilisent des algorithmes pour traiter les contraints, positionner les objets dans la scène et synchroniser les animations. Les résultats que nous avons obtenus nous ont permis de constater que notre système était capable de traduire la description de la scène vers la scène animée 3D ce qui permet de satisfaire le troisième objectif spécifique.

Le dernier objectif spécifique était la validation de la scène animée dans la communication du message. La scène animée doit être représentative du sens de la phrase initiale. Pour ce faire, le sondage que nous avons effectué nous a permis d'analyser les scènes que nous avons générées. Nous avons vérifié la construction de la scène, la communication du message et l'influence de l'environnement. La section 2.4 de la méthodologie décrit le sondage plus en détail et décrit les différents scénarios qui ont été créés pour valider la communication des messages. L'analyse des résultats a permis de constater que les scènes animées permettent de communiquer mieux le message que les scènes statiques. De plus, les scènes statiques permettent de communiquer des états plus facilement. Par conséquent, si les scènes statiques et les scènes animées sont générées au besoin lorsqu'il faut représenter un état ou une action respectivement, le message est communiqué plus facilement à l'observateur. Ces résultats nous permettent de satisfaire ce dernier objectif spécifique.

La réalisation de ces objectifs spécifiques nous a permis de valider les hypothèses qui ont été à la base de cette recherche. En effet, la première hypothèse énonçait qu'il est possible de représenter une phrase simple avec un langage script qui décrit le sens de la phrase. Lors du premier objectif spécifique, nous avons créé le formalisme qui permet de représenter la description de la phrase, par conséquent, nous pouvons valider cette hypothèse. La deuxième hypothèse énonçait qu'il est possible de traduire une description de la phrase sous le formalisme implanté vers une l'animation 3D dans un format d'échange 3D. Nous avons été capables de valider cette hypothèse avec le deuxième et troisième objectif spécifique. Nous avons montré que COLLADA est le format le plus approprié pour représenter nos animations 3D et nous avons présenté le système qui permet de faire cette traduction vers l'animation en format COLLADA. Finalement, la troisième hypothèse énonçait que l'animation 3D générée par le système permet de communiquer le message de la phrase initiale. Alors, cette hypothèse a été validée avec notre quatrième objectif spécifique. Nous avons pu constater grâce au sondage que les scènes 3D générées par le système sont capables de transmettre le message si les scènes animées et les scènes statiques sont utilisées correctement lorsque nous voulons représenter des actions ou des états respectivement.

Les contributions de ce travail de recherche se situent à plusieurs niveaux. Tout d'abord, nous avons implanté un formalisme qui se base sur l'animation par image clé qui nous permet de décrire le sens d'une phrase simple. Ce formalisme est simple, extensible et facile à comprendre. Nous nous sommes inspirés de différents systèmes de conversion du texte vers une animation 3D, mais les formalismes qu'ils utilisent sont souvent orientés vers un domaine d'application. Le formalisme que nous avons proposé est plus général et nous pouvons facilement ajouter d'autres caractéristiques à la scène ou aux personnages. De plus, le système que nous avons proposé nous permet de générer l'animation 3D sous format COLLADA. Les systèmes que nous avons étudiés génèrent directement l'animation 3D pour être visualisés. Notre système nous permet de générer le fichier en format COLLADA contenant l'animation 3D et la visualisation est traité séparément. D'autres systèmes comme BEHAVIOR3D ont fait quelque chose de semblable en utilisant X3D, mais personne ne s'est basé sur COLLADA comme format de sortie. Finalement, nous avons fait l'effort d'analyser les scènes 3D générées par notre système avec le sondage que nous avons réalisé. Grâce au sondage, nous avons pu constater les meilleures façons pour générer nos scènes 3D dans le but de transmettre le message correct aux observateurs.

Pour ce qui est des améliorations et des recommandations, nous pensons qu'il serait intéressant d'implémenter un éditeur qui permettrait de créer le fichier XML contenant la description de la scène. Actuellement, cette description est faite manuellement, puisque ce fichier devait être créé par un autre module du projet GITAN. Ce n'était pas une priorité pour notre recherche, mais cela permettrait de faire évoluer notre système indépendamment de GITAN. Notre système pourrait très bien fonctionner de façon autonome et il pourrait être utilisé pour d'autres fins. Par exemple, nous pourrions manipuler des objets, des contraintes et des images clés pour générer des animations facilement et pour créer des scènes complexes plus rapidement qui peuvent être réutilisées et modifiées dans d'autres systèmes. Nous ne générons pas l'affichage de l'animation 3D, nous générons l'animation sous format COLLADA comme fichier de sortie. Ce qui nous permet d'importer et de modifier ou de transformer l'animation que notre système génère, puisque COLLADA est supporté par plusieurs logiciels de traitement 3D. De cette façon, ce système pourrait évoluer et devenir un outil intéressant dans la création d'animations 3D de haut niveau.

RÉFÉRENCES

AKERBERG, O., SVENSSON, H., SCHULZ, B. et NUGUES, P. (2003). Carsim : an automatic 3d text-to-scene conversion system applied to road accident reports. *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, EACL '03, 191–194.

ARAFA, Y., KAMYAB, K., MAMDANI, E., KSHIRSAGAR, S., MAGNENAT-THALMANN, N., CUI, M., GUYE-VUILLÈME, A. et THALMANN, D. (2002). Two approaches to scripting character animation. *Embodied conversational agents – Let's specify and evaluate them !* 21–25.

ARNAUD, R. et PARISI, T. (2010). Developing web applications with collada and x3d. Consulté en janvier 2010, https ://www.khronos.org/collada/presentations/Developing_Web_Applications_with_COLLADA_and_X3D.pdf.

BARES, W. H. et LESTER, J. C. (1999). Intelligent multi-shot visualization interfaces for dynamic 3d worlds. *Proceedings of the 4th international conference on Intelligent user interfaces*. ACM, New York NY - USA, IUI '99, 119–126.

BILASCO, I. M. (2007). *Une approche sémantique pour la réutilisation et l'adaptation de donnée 3D*. Thèse de doctorat, Université Joseph-Fourier - Grenoble I.

CARSON, G. S., PUK, R. F. et CAREY, R. (1999). Developing the vrml 97 international standard. *IEEE Comput. Graph. Appl.*, 19, 52–58.

CASSELL, J., VILHJÁLMSSON, H. H. et BICKMORE, T. (2001). Beat : the behavior expression animation toolkit. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, SIGGRAPH '01, 477–486.

CHOE, B., LEE, H. et KO, H.-S. (2001). Performance-driven muscle-based facial animation. *The Journal of Visualization and Computer Animation*, 12, 67–79.

CHRIS, M. et BRADEN, M. (2010). Openvrml. Consulté en mai 2010, tiré de http ://open-vrml.org.

CLAY, S. et WILHELMS, J. (1996). Put : language-based interactive manipulation of objects. *Computer Graphics and Applications, IEEE*, 16, 31 –39.

COYNE, B. et SPROAT, R. (2001). Wordseye : an automatic text-to-scene conversion system. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, SIGGRAPH '01, 487–496.

DACHSELT, R. et RUKZIO, E. (2003). Behavior3d : an xml-based framework for 3d graphics behavior. *Proceedings of the eighth international conference on 3D Web technology*. ACM, New York, NY, USA, Web3D '03, 101–ff.

DALY, L. et BRUTZMAN, D. (2008). X3d : extensible 3d graphics standard. *ACM SIGGRAPH ASIA 2008 courses*. ACM, New York, NY, USA, SIGGRAPH Asia '08, 22 :1–22 :6.

DARKEN, R., MCDOWELL, P. et JOHNSON, E. (2005). Projects in vr : the delta3d open source game engine. *Computer Graphics and Applications, IEEE*, 25, 10–12.

DING, L., BALL, A., MATTHEWS, J., MCMAHON, C. A. et PATEL, M. (2007). Product representation in lightweight formats for product lifecycle management (plm).

DIRK, R., GERRIT, V. et JOHANNES, B. (2002). Opensg : Basic concepts. *In 1. OpenSG Symposium OpenSG*. 200–2.

EKMAN, P. et ROSENBERG, E. (2005). The study of spontaneous facial expressions in psychology. *What the face reveals : basic and applied studies of spontaneous expression using the facial action coding system (FACS)*. Oxford University Press, Oxford, England, 2, 3–18.

FRAMEFORGE (2009). Unity script language. Consulté en avril 2010, tiré de http ://www.frameforge3D.com/.

FREEVR (2010). Freevr : Virtual reality integration library. Consulté en avril 2010, tiré de http ://freevr.org.

GEER, D. (2005). Making 3d technology universal. *Computer*, 38, 23 – 25.

GETTO, P. et BREEN, D. (1990). An object-oriented architecture for a computer animation system. *The Visual Computer*, 6, 79–92.

GLASS, K. et BANGAY, S. (2007). Constraint-based conversion of fiction text to a time-based graphical representation. *Proceedings of the 2007 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*. ACM, New York, NY, USA, SAICSIT '07, 19–28.

HERRLICH, M., HOLLE, H. et MALAKA, R. (2010). *Integration of CityGML and Collada for High-Quality Geographic Data Visualization on the PC and Xbox 360*, vol. 6243 de *Lecture Notes in Computer Science*. Springer Berlin - Heidelberg.

HORPRASERT, T., HARITAOGLU, I., WREN, C., HARWOOD, D., DAVIS, L. et PENTLAND, A. (1998). Real-time 3d motion capture. *In Workshop on Perceptual User Interfaces*. 87–90.

HUANG, Z., ELIENS, A. et VISSER, C. (2003). Implementation of a scripting language for vrml-x3d-based embodied agents. *Proceedings of the eighth international conference on 3D Web technology*. ACM, New York NY - USA, Web3D '03, 91–100.

IZANI, M., AISHAH, ESHAQ, A. et NORZAIHA (2003). Keyframe animation and motion capture for creating animation : a survey and perception from industry people. *Research and Development, 2003. SCORED 2003. Proceedings. Student Conference on*. 154 – 159.

JUNKER, G. (2006). *Pro OGRE 3D Programming (Pro)*. Apress., Berkely CA - USA.

KANEKO, K. et OKADA, Y. (2008). Skeleton based 3d model morphing using barycentric map. *Computer Graphics, Imaging and Visualisation, 2008. CGIV '08. Fifth International Conference on*. 132 –137.

KIM, J. O., LEE, B. R. et CHUNG, C. H. (2002). The inductive inverse kinematics algorithm using uniform posture map. *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on.* vol. 1, 697 – 700 vol.1.

KISS, S. (2002). Computer animation for articulated 3d characters. *CTIT technical reports series.* Consulté en mai 2010, tiré de http ://doc.utwente.nl/38232/1/000000aa.pdf.

KWON, J.-Y. et LEE, I.-K. (2008). Determination of camera parameters for character motions using motion area. *The Visual Computer*, 24, 475–483.

LI, T.-Y. et CHENG, C.-C. (2008). Real-time camera planning for navigation in virtual environments. A. Butz, B. Fisher, A. Krüger, P. Olivier et M. Christie, éditeurs, *Smart Graphics*, Springer Berlin / Heidelberg, vol. 5166 de *Lecture Notes in Computer Science.* 118–129.

LLC, S. C. S. (2010). Xml marker easily browse and edit xml files. Consulté en janvier 2010, tiré de http ://symbolclick.com/index.htm.

MACVITTIE, L. A. (2006). *XAML in a Nutshell (In a Nutshell (O'Reilly)).* O'Reilly Media, Inc.

MAGNENAT-THALMANN, N. et THALMANN, D. (1991). Complex models for animating synthetic actors. *Computer Graphics and Applications, IEEE*, 11, 32 –44.

MENTORSYSTEMS (2009). The mentor system talking head. Consulté en mai 2010, tiré de http ://www.mentor.computing.edu.au/.

MICROSYSTEMS, S. (2010). Project wonderland. Consulté en mai 2010, tiré de http ://java.net/projects/wonderland/.

MIYAHARA, K. et OKADA, Y. (2009). Collada-based file format supporting various attributes of realistic objects for vr applications. *Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09. International Conference on.* 971 –976.

NADEAU, D. (1999). Building virtual worlds with vrml. *Computer Graphics and Applications, IEEE*, 19, 18 –29.

PAUL, M. (2007). A quick introduction to the cross-platform open source scene graph api. *OpenSceneGraph Quick Start Guide*. 1–5.

PREDA, M., ARSOV, I. et MORAN BURGOS, F. (2010). Collada + mpeg-4 or x3d + mpeg-4. *IEEE Vehicular Technology Magazine*, 5, 39–47. 10158.

REINERS, D. (2002). Scene Graph Rendering. *OpenSG Forums*, vol. 9, pp. 95–102.

REYNOLDS, C. W. (1987). Flocks, herds and schools : A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21, 25–34.

RUSSELL, K. et BLUMBERG, B. (1999). Behavior-friendly graphics. *Computer Graphics International, 1999. Proceedings*. 44 –50, 241.

SCHOOL, A. M. et MARRIOTT, A. (2007). Vhml - the virtual human markup language. Consulté en mai 2010, tiré de http ://www.miv.t.u-tokyo.ac.jp/pricai02-LAA/papers/andrew-marriott-abstract.pdf.

SHIM, H. et KANG, B. G. (2008). Cameo - camera, audio and motion with emotion orchestration for immersive cinematography. *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*. ACM, New York, NY, USA, ACE '08, 115–118.

SMITH, D., KAY, A., RAAB, A. et REED, D. (2003). Croquet - a collaboration system architecture. *Creating, Connecting and Collaborating Through Computing, 2003. C5 2003. Proceedings. First Conference on*. 2–9.

SOWIZRAL, K., RUSHFORTH, K. et SOWIZRAL, H. (1997). *The Java 3D API Specification*. Addison-Wesley Longman Publishing Co. Inc., Boston MA - USA, première édition.

SPACE, C. (2010). Crystal space. Consulté en mai 2010, tiré de http ://www.crystalspace3d.org.

THALMANN, D. (1996). Physical, behavioral, and sensor-based animation. St. Petersburg Russia, 214–221. Consulté en ligne, avril 2010, tiré de http ://ligwww.epfl.ch/ thalmann/papers.dir/Graphicon96.pdf.

TRENHOLME, D. et SMITH, S. (2008). Computer game engines for developing first-person virtual environments. *Virtual Reality*, 12, 181–187.

UNI-VERSE.ORG (2010). Uni-verse 3d engine. Consulté en mai 2010, tiré de http ://www.uni-verse.org/.

W3C (2009). Xml schema. Consulté en mai 2010, tiré de XML Schema : http ://www.w3.org/XML/Schema.

WEB3D, C. (2010). X3d tool kit. Consulté en mai 2010, tiré de http ://artis.imag.fr/Software/X3D.

WERNECKE, J. (1993). *The Inventor Mentor : Programming Object-Oriented 3d Graphics with Open Inventor, Release 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, première édition.

ANNEXE A

Contraintes de position

Types de contraintes

Nom	***Description***
closeTo	Proche de la référence.
awayFrom	Loin de la référence.
atDistance	Le noeud est à une certaine distance de la référence.
between	Entre deux références.
inFrontOf	Devant la référence.
behindOf	En arrière de la référence.
leftOf	À gauche de la référence
rightOf	À droite de la référence
over	La contrainte force le noeud à être sur la référence.
under	La contrainte force le noeud à être en dessous de la référence.
insideOf	Place le noeud à l'intérieur de la référence.
onTopOf	Place le noeud à la position de la référence

ANNEXE B

Réponses au sondage

1.

[R1a] => Jean est sur le toit du cinéma

[R1b] => Jean est au cinéma

[R2a] => Jean va au cinéma

[R2b] => Jean va au cinéma

[R3a] => Jean, tout habillé, est monté sur la chaise du maître-nageur et regarde la piscine vide et sans eau

[R3b] => Le maître-nageur surveille la piscine

[R4a] => Jean, tout habillé, saute dans la piscine vidée de son eau

[R4b] => Jean saute dans la piscine du plus haut plongeoir

[R5a] => Un ballon passe de la tête de Jean à celle de Sophie

[R5b] => Jean et Sophie jouent au ballon dans un gymnase

[R6a] => Sophie court sur le trottoir, et décide soudainement d'entrer dans le cinéma

[R6b] => Sophie va au cinéma

[R7a] => Une pomme tombe d'un arbre aux feuilles étranges, près de la piscine

[R7b] => Une pomme tombe d'un arbre, près de la piscine

[R8a] => Jean, tout habillé, marche dans la piscine vidée de son eau, sans doute pour l'inspecter

[R8b] => Jean descend dans la piscine et nage

[R9a] => Jean, tout habillé, regarde la piscine vidée de son eau du haut du plongeoir

[R9b] => Jean s'apprête à sauter dans la piscine du haut du plongeoir

[R10a] => Jean se dirige vers le cinéma et y entre

[R10b] => Jean se dirige vers le cinéma et y entre

[sexe] => F

[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x		x
4		x		x
5		x		x
6		x		x
7		x		x
8	x			x
9	x		x	
10		x		x
Total	2	8	2	8

2.

[R1a] => Bonhomme sur le cinéma

[R1b] => La petite amie de Bob vient de le planter là, il veut se suicider en sautant en bas du toit du cinéma

[R2a] => Quelqu'un marche vers le cinéma

[R2b] => La personne est en avance, elle marche lentement.

[R3a] => Quelqu'un est debout sur une des chaises de sauveteur

[R3b] => La piscine est actuellement fermée

[R4a] => Quelqu'un court sur le tremplin

[R4b] => La personne ne connaît pas les règles de sécurité et ne sais pas plonger.

[R5a] => Deux personnes se trouvent sur un terrain de soccer.

[R5b] => Les deux se lancent un ballon

[R6a] => Quelqu'un marche sur le trottoir pour finalement tourner en direction du cinéma.

[R6b] => La petite vieille marche pas vite, elle va être en retard à la représentation de l'âge d'or.

[R7a] => Quelque chose tombe de l'arbre

[R7b] => Le fruit est mur

[R8a] => Quelqu'un marche sur l'eau dans la piscine.

[R8b] => Jésus a probablement ressuscité, parce que l'eau ne semble pas gelé.

[R9a] => Il y a quelqu'un sur le plus haut des 2 tremplins

[R9b] => Bob a la chienne de sauter du tremplin de 3 mètres

[R10a] => La porte du cinéma est ouverte.
[R10b] => Quelqu'un s'engouffre dans le cinéma.
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x	x	
4		x		x
5		x		x
6		x		x
7		x		x
8	x			x
9	x		x	
10		x		x
Total	2	8	3	7

3.
[R1a] => Un homme est sur le toit d'un cinéma.
[R1b] => Un employé fait de l'entretien sur un toit.
[R2a] => Une personne marche vers un cinéma.
[R2b] => Une personne va écouter un film au cinéma
[R3a] => Une personne est sur une chaise de surveillance
[R3b] => Un sauveteur surveille la piscine
[R4a] => Une personne court sur un tremplin puis saute
[R4b] => une personne plonge dans la piscine
[R5a] => Un ballon flotte bizarrement entre deux personnes sur un terrain de soccer.
[R5b] => Deux personnes jouent au ballon dehors, sur un terrain de soccer.
[R6a] => Une personne marche sur la rue puis vers un cinéma
[R6b] => Une personne va au cinéma pour y écouter un film
[R7a] => Un objet rouge se dirige vers le sol depuis le haut d'un arbre
[R7b] => Une pomme tombe d'un arbre
[R8a] => Une personne marche dans une piscine vide
[R8b] => Une personne marche dans une piscine vide... pour une raison obscure
[R9a] => Une personne est debout sur un tremplin

[R9b] => Une personne veut plonger dans une piscine vide et s'y fera mal
[R10a] => une personne rentre dans un cinéma
[R10b] => une personne rentre dans un cinéma pour y écouter un film
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x		x
4		x		x
5	x			x
6		x		x
7	x			x
8		x		x
9		x	x	
10		x		x
Total	2	8	2	8

4.
[R1a] => C'est une personne sur le cinéma
[R1b] => Il y a quelqu'un sur le toit du cinéma.
[R2a] => C'est un cinéma
[R2b] => Quelqu'un marche vers le cinéma
[R3a] => Il y a une piscine
[R3b] => Un sauveteur regarde une piscine
[R4a] => Il y a quelqu'un qui saute du tremplin
[R4b] => Quelqu'un saute d'un tremplin.
[R5a] => Des personnes se lancent un ballon.
[R5b] => Un adulte et une fillette se lance le ballon
[R6a] => quelqu'un marche sur une rue
[R6b] => une fillette se dirige vers le cinéma
[R7a] => Il y a une pomme qui tombe d'un arbre
[R7b] => une pomme tombe d'un arbre
[R8a] => Il y a une piscine
[R8b] => quelqu'un rentre dans la piscine

[R9a] => Il y a quelque chose sur le tremplin
[R9b] => Quelqu'un se prépare à plonger
[R10a] => Il y a un cinéma
[R10b] => Quelqu'un rentre dans le cinéma
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3	x			x
4		x		x
5		x		x
6	x			x
7		x		x
8	x			x
9	x		x	
10	x			x
Total	5	5	1	9

5.
[R1a] => L'homme se tient sur le toit du cinéma.
[R1b] => L'homme déblaie la neige sur le toit du cinéma.
[R2a] => L'homme se dirige vers le cinéma.
[R2b] => L'homme va écouter un film au cinéma.
[R3a] => L'homme se tient sur la chaise du sauveteur de la piscine.
[R3b] => Le sauveteur surveille la piscine.
[R4a] => L'homme cours sur le tremplin et saute dans la piscine.
[R4b] => L'homme plonge dans la piscine.
[R5a] => Les personnes se lance le ballon.
[R5b] => Les personnes jouent au soccer.
[R6a] => La femme marche vers le cinéma.
[R6b] => La femme s'en va regarder un film au cinéma.
[R7a] => Une pomme tombe de l'arbre.
[R7b] => Une pomme tombe de l'arbre
[R8a] => L'homme descend dans la piscine.

[R8b] => L'homme va se baigner.
[R9a] => L'homme se tient sur le tremplin.
[R9b] => L'homme s'apprête à plonger.
[R10a] => L'homme entre dans le cinéma.
[R10b] => L'homme s'en va regarder un film.
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x		x
4		x		x
5		x	x	
6		x		x
7		x		x
8		x		x
9		x	x	
10		x		x
Total	0	10	2	8

6.
[R1a] => Une personne est sur le toit du cinéma.
[R1b] => Une personne veut se suicider en sautant en bas du toit du cinéma.
[R2a] => Une personne marche vers le cinéma.
[R2b] => Une personne va voir un film.
[R3a] => Une personne est debout sur une chaise de sauveteur.
[R3b] => Un sauveteur surveille la piscine.
[R4a] => Une personne court sur un tremplin et saute.
[R4b] => Une personne exécute un plongeon du tremplin.
[R5a] => Une ballon flotte au-dessus d'un terrain se soccer.
[R5b] => Deux personnes s'échangent le ballon en faisant des têtes.
[R6a] => Une personne marche vers le cinéma.
[R6b] => Une fille va voir un film.
[R7a] => Une pomme tombe d'un arbre.
[R7b] => Une pomme tombe d'un pommier.

[R8a] => Une personne marche autour et dans la piscine.

[R8b] => Une personne va se baigner.

[R9a] => Une personne est debout sur un tremplin.

[R9b] => Une personne s'apprête à sauter du tremplin.

[R10a] => Une personne marche vers le cinéma

[R10b] => Une personne va voir un film.

[sexe] => M

[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x	x	
3		x		x
4		x		x
5	x			x
6		x	x	
7		x		x
8	x			x
9		x	x	
10		x	x	
Total	**2**	**8**	**5**	**5**

7.

[R1a] => La personne est sur le toit du cinéma.

[R1b] => La personne est montée sur le toit du cinéma pour vérifier l'état du toit.

[R2a] => La personne marche vers la porte du cinéma.

[R2b] => La personne va voir un film au cinéma.

[R3a] => La personne est debout sur la chaise de sauveteur à côté des chaises jaunes et violettes.

[R3b] => La personne surveille la piscine.

[R4a] => La personne court sur le tremplin et saute de celui-ci dans la piscine.

[R4b] => La personne saute du tremplin.

[R5a] => Les personnes envoient le ballon l'un à l'autre.

[R5b] => Ils jouent à un jeu avec le ballon.

[R6a] => La personne marche sur le trottoir et tourne en direction des portes du cinéma une fois vis-à -vis.

[R6b] => La personne va au cinéma regarder un film.

[R7a] => Une pomme tombe d'un arbre.

[R7b] => Une pomme tombe d'un arbre.

[R8a] => La personne marche sur la partie de la piscine peu profonde qui n'a pas d'eau.

[R8b] => La personne va se baigner dans la partie creuse de la piscine.

[R9a] => La personne se tient sur le tremplin.

[R9b] => La personne va plonger dans la piscine.

[R10a] => La personne entre dans le cinéma.

[R10b] => La personne rentre voir un film au cinéma.

[sexe] => M

[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x		x
4		x		x
5		x		x
6		x		x
7		x		x
8		x		x
9		x	x	
10		x		x
Total	**0**	**10**	**2**	**8**

8.

[R1a] => Il y a une personne debout sur le toit d'un cinéma.

[R1b] => Un homme se tient debout sur le toit de ce cinéma.

[R2a] => Une personne marche lentement vers l'entrée du cinéma.

[R2b] => Cet homme va au cinéma.

[R3a] => Il y a une personne debout sur le poste de surveillance de la piscine.

[R3b] => Un homme se tient debout sur la chaise de sauveteur de la piscine.

[R4a] => Une personne a sauté du tremplin le plus haut de la piscine.

[R4b] => Un homme a sauté du tremplin le plus haut dans une piscine vide.

[R5a] => Un homme et une fillette frappent à coup de tête un ballon de plage dans un terrain de soccer.

[R5b] => Sur le terrain de soccer, père et fille jouent à donner des coups de tête au ballon.

[R6a] => Une personne se dirige toute seule tranquillement vers le cinéma la nuit et s'arrête devant l'entrée.

[R6b] => Une fille va au cinéma la nuit.

[R7a] => Quelque chose est tombée du haut de l'arbre dans la cours à coté de la piscine.

[R7b] => Un gros fruit est tombé de l'arbre.

[R8a] => Une personne est allée dans une piscine vide pendant la nuit.

[R8b] => Un homme marche dans la piscine pendant la nuit.

[R9a] => Une personne est debout sur le haut tremplin de la piscine.

[R9b] => Un homme regarde du haut du tremplin de 3 mètres la piscine qui est vide.

[R10a] => Une personne rentre dans le cinéma.

[R10b] => Un homme va au cinéma.

[sexe] => F

[age] => 35-45

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3		x		x
4		x		x
5		x		x
6		x		x
7	x			x
8	x		x	
9		x		x
10		x		x
Total	**2**	**8**	**1**	**9**

9.

[R1a] => Le sujet est au-dessus du cinéma

[R1b] => Employé d'entretien au-dessus du cinéma pour effectuer une certaine tâche

[R2a] => Sujet marche vers l'entrée du cinéma

[R2b] => Sujet va entrer au cinéma

[R3a] => Sujet debout sur la chaise d'observation du maître-nageur

[R3b] => Maître-nageur qui s'entraîne

[R4a] => Sujet qui court sur le plongeoir sans s'arrêter et tombe

[R4b] => Sujet qui effectue un plongeon

[R5a] => Un homme et une fillette qui se passent un ballon dans un terrain de football (soccer)

[R5b] => Un père et sa fille qui jouent dans un terrain de football

[R6a] => Sujet qui marche vers la porte du cinéma

[R6b] => Sujet qui entre au cinéma

[R7a] => Pomme qui tombe

[R7b] => Pomme qui tombe d'un arbre

[R8a] => Sujet qui marche dans une piscine vide

[R8b] => Un homme qui explore le fond de la piscine lorsqu'elle est vide

[R9a] => Sujet debout sur le plongeoir

[R9b] => Un homme qui s'apprête à plonger

[R10a] => Sujet s'éloignant du cinéma

[R10b] => Un homme qui sort du cinéma

[sexe] => M

[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3		x	x	
4.		x		x
5		x		x
6		x	x	
7		x		x
8		x	x	
9		x	x	
10	x		x	
Total	**1**	**9**	**5**	**5**

10.

[R1a] => L'homme est sur le cinéma.

[R1b] => L'homme est sur le toit du cinéma.

[R2a] => L'homme se dirige vers le cinéma.

[R2b] => L'homme va au cinéma.

[R3a] => L'homme est sur la chaise du sauveteur.

[R3b] => Le sauveteur surveille la piscine.

[R4a] => Le bonhomme court et tombe en bas du tremplin.

[R4b] => Le bonhomme fait un plongeon.

[R5a] => Un gars et une fille sont sur un terrain de soccer et un ballon de plage rebondi entre eux.

[R5b] => Un gars et une fille se passent un ballon de plage sur un terrain de soccer.

[R6a] => Une madame marche sur le trottoir et tourne en direction du cinéma.

[R6b] => Une madame se dirige vers au cinéma en marchant.

[R7a] => Une pomme tombe d'un arbre se trouvant entre une maison et une piscine.

[R7b] => Une pomme tombe du pommier entre la maison et la piscine.

[R8a] => Le bonhomme va marcher dans la piscine.

[R8b] => Le bonhomme va se baigner.

[R9a] => Le bonhomme se tient sur le tremplin

[R9b] => Le bonhomme va faire un plongeon.

[R10a] => Le bonhomme entre dans le cinéma.

[R10b] => Le bonhomme entre dans le cinéma.

[sexe] => M

[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3		x		x
4		x		x
5		x		x
6		x		x
7		x		x
8	x			x
9		x	x	
10		x		x
Total	**1**	**9**	**1**	**9**

11.

[R1a] => L'homme est à une hauteur égale au toit du cinéma, et est situé dans son centre.

[R1b] => L'homme est sur le cinéma.

[R2a] => L'homme se déplace vers l'entrée du cinéma.

[R2b] => L'homme va au cinéma

[R3a] => L'homme est sur une chaise de surveillance, au bord d'une piscine (intérieur mais il y a des chaises longues, ce qui ne fait pas de sens)

[R3b] => Le sauveteur surveille la piscine depuis sa chaise.

[R4a] => L'homme est sur le plongeon.

[R4b] => Le baigneur veut plonger.

[R5a] => L'homme est sur le terrain de basket.

[R5b] => L'homme joue au basket (tout seul).

[R6a] => (je ne peux pas ouvrir les animations)...La femme se rend au cinéma

[R6b] => La femme se rends au cinéma

[R7a] => La piscine extérieure...a des chaises magenta et roses...

[R7b] => la piscine publique est à l'extérieur... ? ? ?

[R8a] => La piscine publique est vide.

[R8b] => La piscine publique est vide.

[R9a] => L'homme est sur le tremplin.

[R9b] => L'homme s'apprête à plonger dans une piscine vide et il va se tuer.

[R10a] => Il y a un cinéma dans l'image.

[R10b] => L'homme (quitte ou va) au cinéma (je ne peux pas ouvrir les animations).

[sexe] => M

[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3		x		x
4	x		x	
5	x		x‘	
6		x		x
7	x		x	
8	x		x	
9		x	x	
10	x			x
Total	**5**	**5**	**5**	**5**

12.

[R1a] => L'homme est sur le cinéma

[R1b] => L'homme est au cinéma
[R2a] => L'homme marche vers le cinéma
[R2b] => L'homme va au cinéma
[R3a] => L'homme est sur le plongeoir de la piscine
[R3b] => L'homme se prépare à plonger du plongeoir de la piscine
[R4a] => L'homme rate son plongeon
[R4b] => L'homme plonge dans la piscine
[R5a] => un gars et une fille se passent le ballon avec la tête
[R5b] => un gars et une fille jouent au ballon sur un terrain
[R6a] => Une femme marche vers le cinéma
[R6b] => Une femme va au cinéma
[R7a] => Une pomme tombe de l'arbre du jardin
[R7b] => Une pomme tombe de l'arbre du jardin
[R8a] => un homme entre et marche dans la piscine
[R8b] => un homme entre dans la piscine
[R9a] => L'homme est sur le plongeon de la piscine extérieure
[R9b] => L'homme se prépare à plonger du plongeoir de la piscine extérieure
[R10a] => un homme entre au cinéma
[R10b] => un homme va au cinéma
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3	x		x	
4		x		x
5		x		x
6		x		x
7		x		x
8		x		x
9		x	x	
10		x		x
Total	**1**	**9**	**3**	**7**

13.

[R1a] => L'individu est au-dessus du cinéma
[R1b] => je suis sur le toit du cinéma
[R2a] => L'individu marche vers le cinéma
[R2b] => je vais au cinéma
[R3a] => l'individu est sur la chaise de sauveteur
[R3b] => je suis sauveteur
[R4a] => l'individu marche vers la piscine
[R4b] => je saute du tremplin
[R5a] => le ballon de volleyball passe d'un terrain à l'autre
[R5b] => nous jouons au volleyball sur un terrain de soccer
[R6a] => l'individu marche devant le cinéma, puis vers le cinéma
[R6b] => je vais au cinéma
[R7a] => la pomme tombe de l'arbre
[R7b] => la gravité
[R8a] => l'individu marche vers la piscine
[R8b] => heu, john enters pool ?
[R9a] => l'individu est sur le tremplin
[R9b] => je me prépare à plonger
[R10a] => l'individu rentre dans le cinéma, par la porte
[R10b] => je vais au cinéma.
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3		x	x	
4		x		x
5	x			x
6		x		x
7		x	x	
8	x			x
9		x	x	
10		x		x
Total	**2**	**8**	**3**	**7**

14.
[R1a] => guardián encima del cine
[R1b] => cine en funcionamiento
[R2a] => Saliendo del cine
[R2b] => No le gusto la película
[R3a] => Piscina sin agua
[R3b] => Calculando la altura de la piscina en la zona mas alejada del trampolín
[R4a] => Parado en el trampolín de la piscina sin agua
[R4b] => Calculando la altura en la zona donde esta el trampolín
[R5a] => Dos personas jugando con un balón
[R5b] => Dos personas jugando soccer
[R6a] => Cine muy bien ubicado
[R6b] => Fuera de hora de funcionamiento del cine
[R7a] => Manzana cayendo del árbol
[R7b] => Manzana madura cayendo del árbol
[R8a] => Club o casa con piscina, pero sin agua
[R8b] => No es temporada de verano
[R9a] => Hombre observando piscina sin agua
[R9b] => No es temporada para uso de piscina
[R10a] => Entrando al cine
[R10b] => Ir al cine
[sexe] => M
[age] => 55-65

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2	x		x	
3	x		x	
4	x		x	
5		x		x
6	x		x	
7		x		x
8	x		x	
9		x	x	
10		x		x
Total	**5**	**5**	**7**	**3**

15.
[R1a] => Un hombre encima de un edificio, mas o menos en el centro del techo
[R1b] => Un hombre que puede estar pensando en matarse
[R2a] => El hombre sale del edificio caminando
[R2b] => Se arrepintió de quitarse la vida
[R3a] => Un hombre observando el fondo de la piscina vacina, encima de un banco
[R3b] => Estar pensando en lanzarse al vació de la piscina
[R4a] => Dos trampolines uno alto y uno bajo y hamacas a los costados.
[R4b] => Un sillón muy grande
[R5a] => Una cancha de fútbol con dos personas en ambos lados
[R5b] => Un grupo de personas corriendo en la margen izquierda con una banderola
[R6a] => Una autopista que pasa frente a un cine al fondo pasan carros
[R6b] => Una persona caminando hacia el fondo para llegar al mar
[R7a] => Una caseta amarilla frente a una cerca donde atrasse encuentran hamacas
[R7b] => La cerca se esta cayendo y tuiene parantes que la sujetan
[R8a] => Una piscina y una caseta donde se cambian para bañarse
[R8b] => Unas casitas frente a un tobogán gigante
[R9a] => Un hombre parado encima de un trampolín
[R9b] => Subió las escaleras y tiene que detenerse para volver a subir y no detenerse
[R10a] => Una persona agachada frente a un cinema
[R10b] => Se esta alejando feliz después de haber cisto una película
[sexe] => F
[age] => 45-55

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2	x		x	
3		x	x	
4	x		x	
5	x		x	
6	x		x	
7	x		x	
8	x		x	
9		x	x	
10	x		x	
Total	**7**	**3**	**10**	**0**

16.

[R1a] => Il y a un personnage sur le toit d'un cinéma

[R1b] => Pour une raison x le personnage est grimpé sur le toit

[R2a] => Un personnage marche vers les portes du cinéma

[R2b] => Le personnage va au cinéma

[R3a] => Il y a un personnage debout sur une chaise de surveillant sauveteur

[R3b] => Il surveille la piscine

[R4a] => Le personnage court sur le tremplin et saute dans la piscine

[R4b] => Le personnage veut sauter dans la piscine

[R5a] => Il y a un ballon qui fait des allers retours entre les 2 personnages

[R5b] => les personnages se lancent le ballon

[R6a] => Il y a un personnage qui marche devant le cinéma et tourne en direction des portes

[R6b] => Le personnage s'apprête à aller au cinéma

[R7a] => Il y a une pomme qui tombe d'un arbre

[R7b] => La pomme est trop mûre et elle tombe toute seule

[R8a] => Un personnage marche dans la partie peu profonde d'un piscine vide

[R8b] => Il va vérifier s'il n'y a pas de bris

[R9a] => Il y a un personnage debout sur un tremplin

[R9b] => Le personnage s'apprête à sauter dans la piscine

[R10a] => Un personnage entre par la porte ouverte du cinéma

[R10b] => Le personnage va voir un film

[sexe] => F

[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x		x
4		x		x
5		x		x
6		x		x
7		x		x
8		x	x	
9		x	x	
10		x	x	
Total	**0**	**10**	**4**	**6**

17.

[R1a] => La personne est sur le toit d'un cinéma

[R1b] => La personne veut regarder la vue du toit d'un cinéma

[R2a] => Quelqu'un marche vers l'entrée du cinéma

[R2b] => Quelqu'un marche lentement pour aller voir un film au cinéma

[R3a] => Quelqu'un est installer sur la chaise de surveillance d'une piscine

[R3b] => Un sauveteur surveille les gens dans la piscine

[R4a] => quelqu'un marche sur un plongeon et tombe

[R4b] => un homme fait un plongeon.

[R5a] =>

[R5b] =>

[R6a] => deux personnes se lancent un ballon

[R6b] => deux personnes jouent au volleyball

[R7a] => une pomme tombe de l'arbre

[R7b] => un coup de vent fait tomber la pomme de l'arbre

[R8a] => quelqu'un utilise l'échelle pour aller dans l'eau

[R8b] => un homme décide d'aller nager en se dirigeant vers la partie la plus profonde de la piscine

[R9a] => un homme est sur le plongeon

[R9b] => un homme se prépare a faire un plongeon

[R10a] => un homme marche

[R10b] => un homme se rend dans un cinéma à la marche.

[sexe] => F
[age] => 15-2

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x		x
4		x		x
5	x		x	
6		x		x
7		x		x
8		x		x
9		x	x	
10	x			x
Total	**2**	**8**	**3**	**7**

18.
[R1a] => Une personne sur le toit d'un cinéma
[R1b] => Une personne qui s'apprête à réparer quelque chose sur le toit du cinéma
[R2a] => Une personne qui entre dans un cinéma
[R2b] => Une personne qui entre dans un cinéma pour regarder un film
[R3a] => Une personne debout sur une grande chaise à coté d'une piscine.
[R3b] => Un sauveteur dans une piscine debout sur sa chaise qui s'apprête à plonger
[R4a] => Une personne qui saute d'un plongeoir dans une piscine.
[R4b] => Un plongeur professionnel qui s'entraîne dans une piscine.
[R5a] => Deux personnes qui se lancent un ballon sur un stade de Basketball
[R5b] => Un papa qui joue avec sa fille au ballon sur un stade de Bascketball
[R6a] => Une fille qui part au cinéma
[R6b] => Une fille qui part regarder un film dans un cinéma
[R7a] => Une pomme qui tombe d'un arbre
[R7b] => Une pomme trop mûre qui tombe d'un arbre
[R8a] => Une personne qui marche dans une piscine vide
[R8b] => Un employé qui inspecte une piscine vide
[R9a] => Une personne debout sur un petit plongeoir.
[R9b] => Une personne qui s'apprête à plonger d'un petit plongeoir dans une piscine.
[R10a] => Une personne qui entre dans un cinéma

[R10b] => Une personne qui entre dans un cinéma pour regarder un film
[sexe] => M
[age] => 25-35

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2	x		x	
3		x	x	
4		x	x	
5		x		x
6		x		x
7		x		x
8		x	x	
9		x	x	
10		x		x
Total	**1**	**9**	**6**	**4**

19.
[R1a] => il est sur le toit
[R1b] => il va sauter du toit
[R2a] => il se dirige vers le cinéma
[R2b] => il se dirige vers le cinéma
[R3a] => il regarde vers la piscine
[R3b] => il regarde vers la piscine
[R4a] => il va en courant et tombe dans la piscine
[R4b] => il veut sauter dans la piscine
[R5a] => la balle va du bonhomme vers la fille et le contraire aussi
[R5b] => ils jouent au ballon
[R6a] => elle marche vers le cinéma
[R6b] => elle marche vers le cinéma
[R7a] => la pomme tombe de l'arbre
[R7b] => la pomme tombe de l'arbre
[R8a] => il marche dans la piscine
[R8b] => il marche dans la piscine et se dirige vers l'eau profonde
[R9a] => il est sur le trampoline
[R9b] => il va sauter du trampoline

[R10a] => il est entré au cinéma
[R10b] => il est entré au cinéma
[sexe] => F
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3	x			x
4		x		x
5		x		x
6		x		x
7		x		x
8		x		x
9		x	x	
10		x		x
Total	**1**	**9**	**2**	**8**

20.
[R1a] => une personne debout sur le toit d'un cinéma
[R1b] => une personne marche sur le toit d'un cinéma
[R2a] => une personne marche en direction d'un cinéma
[R2b] => une personne va entrer dans un cinéma
[R3a] => une personne debout sur une chaise devant une piscine
[R3b] => une personne va sauter dans la piscine
[R4a] => une personne coure et saute d'un plongeoir
[R4b] => une personne plonge dans une piscine
[R5a] => un ballon se déplace dans les airs
[R5b] => deux personnes jouent au ballon
[R6a] => une personne marche sur le trottoir et tourne au niveau d'un cinéma
[R6b] => une personne va au cinéma après avoir magasiné toute la journée.
[R7a] => une pomme tombe d'un arbre
[R7b] => une pomme tombe d'un arbre et vient s'écraser au sol
[R8a] => un homme marche dans une piscine
[R8b] => une homme nage dans une piscine
[R9a] => une personne debout sur un plongeoir

[R9b] => une personne s'apprête à sauter d'un plongeoir
[R10a] => une personne marche et entre dans un cinéma
[R10b] => une personne va au cinéma
[sexe] => F
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x	x	
4		x		x
5	x			x
6	x			x
7		x		x
8		x	x	
9		x	x	
10		x		x
Total	**2**	**8**	**4**	**6**

21.
[R1a] => Un homme est sur le toit d'un cinéma.
[R1b] => Un homme va se jeter en bas du toit d'un cinéma.
[R2a] => Un homme marche à l'extérieur d'un cinéma.
[R2b] => Un homme s'en va d'un film.
[R3a] => Une femme est debout sur une chaise de sauveteur.
[R3b] => Une femme surveille la piscine.
[R4a] => Quelqu'un est sur le plongeon.
[R4b] => Quelqu'un va sauter du plongeon.
[R5a] => Deux personnes sont sur un terrain de soccer.
[R5b] => Deux personnes jouent au soccer.
[R6a] => Quelqu'un marche devant un cinéma.
[R6b] => Quelqu'un s'en va voir un film.
[R7a] => Il y a une maison avec des lumières allumées alors qu'il fait nuit.
[R7b] => Les lumières vont s'éteindre.
[R8a] => Il y a une piscine vide.
[R8b] => La piscine va se remplir.

[R9a] => Il y a quelqu'un sur un plongeon.
[R9b] => Il va sauter.
[R10a] => Quelqu'un marche devant un cinéma.
[R10b] => Il n'ira pas au cinéma.
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2	x		x	
3		x		x
4		x		x
5	x			x
6	x		x	
7	x		x	
8	x		x	
9		x	x	
10	x		x	
Total	**6**	**4**	**7**	**3**

22.
[R1a] => Un homme sur le toit d'un cinéma
[R1b] => Un homme qui veut probablement se suicider.
[R2a] => Quelqu'un qui se dirige vers la porte d'un cinéma.
[R2b] => Quelqu'un qui va voir un film au cinéma.
[R3a] => Quelqu'un debout sur une chaise de sauveteur
[R3b] => Quelqu'un qui va plonger dans la piscine.
[R4a] => Quelqu'un qui cours sur un plongeon et saute.
[R4b] => Quelqu'un qui va plonger dans la piscine.
[R5a] => Deux personnes qui se lancent un ballon
[R5b] => Deux personnes qui jouent au soccer
[R6a] => Quelqu'un qui marche vers le cinéma
[R6b] => Quelqu'un qui va voir une film au cinéma.
[R7a] => Un objet rouge qui tombe d'un arbre
[R7b] => Une pomme qui tombe d'un arbre
[R8a] => Quelqu'un qui entre dans la piscine

[R8b] => Quelqu'un qui va se baigner

[R9a] => Quelqu'un debout sur un plongeon

[R9b] => Quelqu'un qui va plonger

[R10a] => Quelqu'un qui entre dans un cinéma

[R10b] => Quelqu'un qui va voir un film

[sexe] => F

[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x	x	
4		x		x
5		x		x
6		x		x
7		x		x
8		x		x
9		x	x	
10		x	x	
Total	**0**	**10**	**4**	**6**

23.

[R1a] => Hay un hombre parado en el techo del cinema

[R1b] => Persona en espera de algo. tecnico q subio a revisar debido a falla e cine

[R2a] => Hay un hombre parado frente al cinema

[R2b] => El hombre esta ingresando al cinema

[R3a] => Una persona esta observando la piscina vacía

[R3b] => Nadador novato observa como lanzarse/.El juez de natación esta chequeando las medidas reglamentarias

[R4a] => Persona sube al trampolín para realizar practicas. La piscina esta vacía

[R4b] => La persona camina sobre el trampolín y se resbala . Cae y ...

[R5a] => Hay una pareja en el campo/ La pareja esta cerca del centro del campo listos para jugar

[R5b] => La pareja juega cabezeando la pelota./ El hombre cabecea y la alienígena responde..

[R6a] => La mujer viene con dirección al cinema

[R6b] => La mujer ingresa al cine./ Ingresa ella al escuchar ruido de personas en el cine
[R7a] => Un árbol cerca de la casa colindante con la piscina
[R7b] => Se desprende una manzana del árbol.
[R8a] => Una piscina a desniveles sin agua
[R8b] => John ingresa al nivel superior de la piscina dirigido por una mano.
[R9a] => Un hombre esta parado en el trampolín .
[R9b] => Un hombre esta observando como lanzarse del trampolín
[R10a] => Un hombre esta frente al cinema con los brazos abiertos
[R10b] => Hombre atlético esta ingresando al cinema
[sexe] => F
[age] => 45-55

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2	x			x
3	x		x	
4	x		x	
5		x		x
6		x		x
7	x			x
8	x		x	
9		x	x	
10	x			x
Total	**6**	**4**	**5**	**5**

24.
[R1a] => Un homme sur le cinéma
[R1b] => Un homme qui attend sur le toit.
[R2a] => Quelqu'un qui se marche vers la porte du cinéma.
[R2b] => Quelqu'un qui va voir un film.
[R3a] => Quelqu'un debout sur la chaise de sauveteur
[R3b] => Quelqu'un qui regarde la piscine.
[R4a] => Quelqu'un qui cours sur un plongeon.
[R4b] => Quelqu'un qui tombe dans la piscine.
[R5a] => Deux personnes qui se lancent un ballon
[R5b] => Deux personnes qui jouent avec un ballon

[R6a] => Quelqu'un qui marche vers le cinéma
[R6b] => Quelqu'un qui va voir un film au cinéma.
[R7a] => Une pomme tombe d'un arbre
[R7b] => Une pomme qui tombe d'un arbre
[R8a] => Quelqu'un qui entre dans la piscine
[R8b] => Quelqu'un qui va se baigner
[R9a] => Quelqu'un debout sur un plongeon
[R9b] => Quelqu'un qui va plonger
[R10a] => Quelqu'un qui entre dans un cinéma
[R10b] => Quelqu'un qui va voir un film
[sexe] => F
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3		x		x
4	x		x	
5		x		x
6		x		x
7		x		x
8		x		x
9		x	x	
10		x		x
Total	**1**	**9**	**2**	**8**

25.
[R1a] => Une personne est sur le toit d'un cinéma
[R1b] => Une personne qui attend sur le toit du cinéma
[R2a] => Une personne qui s'approche au cinéma
[R2b] => Une personne qui va au cinéma pour regarder un film
[R3a] => Une personne debout sur une chaise à coté d'une piscine.
[R3b] => Un sauveteur sur sa chaise qui s'apprête à plonger
[R4a] => Une personne qui saute d'un plongeoir dans une piscine.
[R4b] => Un plongeur qui saute dans une piscine.
[R5a] => Deux personnes qui se lancent un ballon

[R5b] => Un monsieur qui joue avec une fille au ballon
[R6a] => Une fille qui part au cinéma
[R6b] => Une fille qui part regarder un film dans un cinéma
[R7a] => Une pomme qui tombe d'un arbre
[R7b] => Une pomme qui tombe d'un arbre
[R8a] => Une personne qui marche dans une piscine
[R8b] => Un personne qui inspecte une piscine
[R9a] => Une personne debout sur un plongeoir.
[R9b] => Une personne qui se prépare a plonger d'un plongeoir dans une piscine.
[R10a] => Une personne qui entre dans un cinéma
[R10b] => Une personne qui entre dans un cinéma pour regarder un film
[sexe] => M
[age] => 25-35

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3		x	x	
4		x		x
5		x		x
6		x		x
7		x		x
8		x	x	
9		x	x	
10		x		x
Total	**0**	**10**	**3**	**7**

26.
[R1a] => C'est une personne sur le cinéma
[R1b] => Il y a quelqu'un sur le toit du cinéma.
[R2a] => C'est un cinéma
[R2b] => Quelqu'un marche vers le cinéma
[R3a] => Il y a une piscine
[R3b] => Une personne regarde une piscine
[R4a] => Il y a quelqu'un qui saute du tremplin
[R4b] => Quelqu'un saute d'un tremplin.

[R5a] => Des personnes se lancent un ballon.
[R5b] => Un adulte et une fillette se lance le ballon
[R6a] => quelqu'un marche sur la rue
[R6b] => une fillette se dirige vers le cinéma
[R7a] => Il y a une pomme qui tombe d'un arbre
[R7b] => une pomme tombe d'un arbre
[R8a] => Il y a une piscine et une personne qui rentre
[R8b] => quelqu'un rentre dans la piscine
[R9a] => Il y a quelqu'un sur le tremplin
[R9b] => Quelqu'un se prépare à plonger
[R10a] => Il y a un cinéma
[R10b] => Quelqu'un rentre dans le cinéma
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3	x			x
4		x		x
5		x		x
6	x			x
7		x		x
8		x		x
9		x		x
10	x			x
Total	3	7	0	10

27.
[R1a] => Il y a une personne debout sur le toit d'un cinéma.
[R1b] => Un homme se tient debout sur le toit de ce cinéma.
[R2a] => Une personne marche lentement vers l'entrée du cinéma.
[R2b] => Cet homme va au cinéma.
[R3a] => Il y a une personne debout sur une chaise dans la piscine.
[R3b] => Un homme se tient debout sur la chaise du sauveteur de la piscine.
[R4a] => Une personne a sauté du tremplin le plus haut de la piscine.

[R4b] => Un homme a sauté du tremplin le plus haut dans une piscine vide.
[R5a] => Un homme et une fillette frappent à coup de tête un ballon.
[R5b] => Un père et une fille qui jouent au ballon.
[R6a] => Une personne se dirige toute seule vers le cinéma
[R6b] => Une fille va au cinéma.
[R7a] => Quelque chose est tombée du haut de l'arbre.
[R7b] => Une pomme est tombé de l'arbre.
[R8a] => Une personne est allée dans une piscine.
[R8b] => Un homme marche dans la piscine.
[R9a] => Une personne est debout sur le haut tremplin de la piscine.
[R9b] => Un homme regarde du haut du tremplin la piscine qui est vide.
[R10a] => Une personne rentre dans le cinéma.
[R10b] => Un homme va au cinéma.
[sexe] => M
[age] => 35-45

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3		x		x
4		x		x
5		x		x
6		x		x
7		x		x
8		x		x
9		x		x
10		x		x
Total	0	10	0	10

28.
[R1a] => L'homme est sur le cinéma
[R1b] => L'homme est sur le cinéma
[R2a] => L'homme marche vers le cinéma
[R2b] => L'homme va au cinéma
[R3a] => L'homme est sur le plongeoir de la piscine
[R3b] => L'homme se prépare à plonger du plongeoir de la piscine

[R4a] => L'homme saute du plongeoir
[R4b] => L'homme plonge dans la piscine
[R5a] => un gars et une fille se passent le ballon avec la tête
[R5b] => un gars et une fille jouent au ballon sur un terrain
[R6a] => Une femme marche vers le cinéma
[R6b] => Une femme va au cinéma
[R7a] => Une pomme tombe de l'arbre du jardin
[R7b] => Une pomme tombe de l'arbre du jardin
[R8a] => un homme entre et marche dans la piscine
[R8b] => un homme entre dans la piscine
[R9a] => L'homme est sur le plongeon de la piscine extérieure
[R9b] => L'homme se prépare à plonger du plongeoir de la piscine extérieure
[R10a] => un homme entre au cinéma
[R10b] => un homme va au cinéma
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x		x
2		x		x
3	x		x	
4		x		x
5		x		x
6		x		x
7		x		x
8		x		x
9		x	x	
10		x		x
Total	**1**	**9**	**2**	**8**

29.
[R1a] => La personne est sur le toit d'un cinéma
[R1b] => La personne regarde la vue du haut du cinéma
[R2a] => Quelqu'un marche vers l'entrée du cinéma
[R2b] => Quelqu'un va voir un film au cinéma
[R3a] => Quelqu'un est debout sur la chaise de surveillance d'une piscine

[R3b] => Un sauveteur surveille les gens dans la piscine

[R4a] => Quelqu'un court sur un plongeon et saute

[R4b] => Un homme fait un plongeon.

[R5a] => Deux personnes se lancent un ballon

[R5b] => Deux personnes jouent au soccer

[R6a] => Une fille marche vers le cinéma

[R6b] => Une fille va voir un film

[R7a] => une pomme tombe de l'arbre

[R7b] => un coup de vent fait tomber la pomme de l'arbre

[R8a] => quelqu'un utilise l'échelle pour aller dans la piscine

[R8b] => un homme décide d'aller nager en se dirigeant vers la partie la plus profonde de la piscine

[R9a] => un homme est sur le plongeon

[R9b] => un homme se prépare à faire un plongeon

[R10a] => un homme marche vers le cinéma

[R10b] => un homme se rend dans un cinéma à la marche.

[sexe] => F

[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x		x
4		x		x
5		x		x
6		x		x
7		x		x
8		x		x
9		x	x	
10	x			x
Total	**1**	**9**	**1**	**9**

30.

[R1a] => La personne est sur le toit du cinéma.

[R1b] => La personne est sur le toit du cinéma.

[R2a] => La personne marche vers la porte du cinéma.

[R2b] => La personne va voir un film au cinéma.
[R3a] => La personne est debout sur la chaise de sauveteur.
[R3b] => La personne surveille la piscine.
[R4a] => La personne court sur le tremplin et saute.
[R4b] => La personne saute du tremplin.
[R5a] => Les personnes envoient le ballon l'un à l'autre.
[R5b] => Ils jouent à / avec le ballon.
[R6a] => La personne marche sur le trottoir et tourne en direction des portes du cinéma.
[R6b] => La personne va au cinéma regarder un film.
[R7a] => Une pomme tombe d'un arbre.
[R7b] => Une pomme tombe d'un arbre.
[R8a] => La personne marche sur la partie de la piscine peu profonde.
[R8b] => La personne va se baigner dans la partie creuse de la piscine.
[R9a] => La personne se tient sur le tremplin.
[R9b] => La personne va plonger dans la piscine.
[R10a] => La personne entre dans le cinéma.
[R10b] => La personne rentre voir un film au cinéma.
[sexe] => M
[age] => 15-25

No Phrase	Phrase A		Phrase B	
	Incorrecte	Correcte	Incorrecte	Correcte
1		x	x	
2		x		x
3		x		x
4		x		x
5		x		x
6		x		x
7		x		x
8		x		x
9		x	x	
10		x		x
Total		**10**	**2**	**8**

ANNEXE C

Compilation des réponses

No Sondage	**Phrase A**		**Phrase B**	
	Incorrecte	**Correcte**	**Incorrecte**	**Correcte**
1	2	8	2	8
2	2	8	3	7
3	2	8	2	8
4	5	5	1	9
5	0	10	2	8
6	2	8	5	5
7	0	10	2	8
8	2	8	1	9
9	1	9	5	5
10	1	9	1	9
11	5	5	5	5
12	1	9	3	7
13	2	8	3	7
14	5	5	7	3
15	7	3	10	0
16	0	10	4	6
17	2	8	3	7
18	1	9	6	4
19	1	9	2	8
20	2	8	4	6
21	6	4	7	3
22	0	10	4	6
23	6	4	5	5
24	1	9	2	8
25	0	10	3	7
26	3	7	0	10
27	0	10	0	10
28	1	9	2	8
29	1	9	1	9
30	0	10	2	8
Total	**61**	**239**	**97**	**203**