Cahier des charges

PFA - De la 3D vers la 2D

Année scolaire 2014-2015

Client : BLANC Carole, DESBARATS Pascal

Encadrant : LOMBARDY Sylvain

Equipe : BOHER Anaïs - CABON Yohann - CHAUVAT Magali
LEVY Akané - MARCELIN Thomas
MAUPEU Xavier - PHILIPPI Alexandre

Table des matières

1 Domaine

Ce projet s'inscrit dans le domaine de la synthèse d'images et de la visualisation de modèles en trois dimensions. Depuis le quinzième siècle, grâce à la peinture, la perspective apparaît sur des supports en deux dimensions. Aux XIXème et XXème siècles, l'utilisation de stéréoscopes, tel que le stéréoscope de Holmes, permettait la visualisation de relief à partir de deux images planes et d'un dispositif optique. Dans la deuxième moitié du XXème siècle, l'utilisation du numérique permet de modifier les images et d'obtenir une meilleure visualisation de la profondeur sur des supports en deux dimensions.

On peut ainsi créer des anaglyphes, des autostéréogrammes ou des flipbooks, qui sur papier ou sur écran permettent d'apercevoir la profondeur d'une scène grâce à des techniques adaptées. Ces différents rendus seront présentés plus tard dans ce cahier des charges. De nos jours, il existe également des logiciels, tels que Meshlab et Blender qui sont gratuits, open source et permettent d'ores et déjà la visualisation en trois dimensions sur un écran. L'utilisateur peut tourner autour d'un objet et le voir sous tous ses angles grâce à un ensemble de projections successives autour de l'objet.

On appelle synthèse d'image l'ensemble des techniques qui permettent de visualiser des objets en trois dimensions en perspective sur un écran d'ordinateur, en tenant compte de lumières et de textures appliquées à l'objet. Il existe un grand nombre de techniques et les résultats obtenus peuvent eux aussi varier (perspective isométrique, perspective conique...). Nous nous préoccupons par la suite de la perspective conique, dite aussi vue naturelle.

Bien souvent, la synthèse d'image utilise le principe de scène. Il s'agit d'un espace à trois dimensions dans lequel des objets peuvent être placés. Ces derniers sont décrits par un ensemble de points disposés dans l'espace.

Pour pouvoir observer la scène et les objets, il est nécessaire de demander à l'ordinateur de les modéliser, c'est-à-dire d'afficher un rendu qui correspondrait à une vision de cette scène si elle était réelle. Pour cela, la machine simule le point de vue de l'utilisateur à l'aide d'une « caméra ». A partir de cette scène en trois dimensions, la caméra peut réaliser des projections ou photographies permettant de créer des anaglyphes, autostéréogrammes ou flipbooks. Plusieurs méthodes de projection existent, mais seule celle par matrice de projection sera utilisée.

Ces matrices sont décrites à l'aide de coordonnées homogènes. Celles-ci ont été introduites afin que l'ensemble des transformations de type rotation, translation et homothétie puissent être écrites sous forme de matrice. Ainsi, le produit des matrices de transformation peut être calculé en amont pour pouvoir appliquer la matrice de la transformation résultante à l'ensemble des points de l'objet sans avoir à recalculer le produit pour chaque point.

Pour pouvoir visualiser un modèle 3D il faut prendre en considération la lumière et sa réflexion sur l'objet. Si une sphère rouge était représentée dans un espace avec uniquement une lumière ambiante, il n'en ressortirait qu'un disque rouge, sans relief. En effet, la lumière ambiante atteint l'objet de la même façon en tout point. On ne peut donc pas savoir depuis un plan fixe s'il s'agit d'un objet en deux ou en trois dimensions. Si maintenant une lumière est ajoutée dans l'espace où est situé l'objet, celle-ci ne va pas atteindre tous les points de l'objet de la même façon. Elle sera plus faible sur un point plus éloignée, voire inexistante sur un point caché. En tenant compte de cette lumière, on peut obtenir une image comme présentée sur la figure 1.1.

Pour la création d'un anaglyphe, deux images espacées par une petite distance (qui correspond à la distance entre les deux yeux par exemple) sont générées. La composante rouge de l'une de ces images et la composante bleue de l'autre sont gardées et ensuite superposées dans une même image. Cette image est ensuite transformée en une image Rouge-Cyan, qui peut être visualisée à l'aide de lunettes Rouge-Bleue : l'image apparaît en trois dimensions.

1. http ://linut.free.fr/omgspl0kuberwebloglolz0r/ ?2010/02/01/93-raytracer-que-la-lumiere-soit

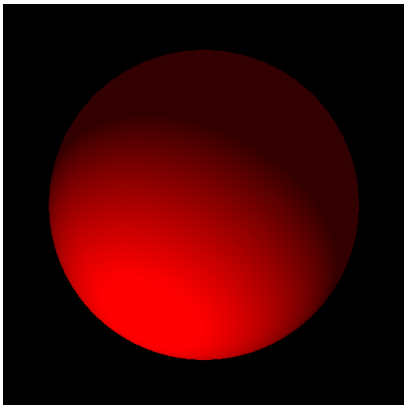

FIGURE 1.1 – Application d'une lumière diffuse à une sphère rouge [1]

Pour la création d'un autostéréogramme, une image permettant d'observer un objet en relief par vision parallèle est générée. Cette image est obtenue à partir d'une texture de base ou de points aléatoires pour l'image de fond.

Pour la création d'un flipbook, plusieurs images sont prises à intervalles réguliers par une caméra suivant un trajet prédéterminé dans ou autour de la scène. Le flipbook est visualisable en faisant rapidement défiler ces images tout en respectant l'ordre des prises de vue. Ce flipbook peut être transformé en GIF pour obtenir une visualisation animée des images.

2 État de l'art et existant

2.1 Logiciels existants

StereoPhoto Maker [2] et Anaglyph Maker 3D [3] sont des freewares permettant, entre autre, depuis deux photos d'obtenir l'anaglyphe rouge / cyan correspondant. La qualité de l'anaglyphe obtenu dépend du décalage entre les deux photos et de la qualité de la prise de vue.

Il existe également de nombreux logiciels de création d'autostéréogrammes, comme Stereographic Suite [4] (©IndaSoftware) et 3DMiracles [5] (©Urry Software Lab) qui permettent de créer des autostéréogrammes à partir d'une carte de profondeurs (image en deux dimensions en niveaux de gris).

2.2 Projection vers deux dimensions

Le passage d'un objet en trois dimensions à une image en deux dimensions se fait par projection. Une caméra présente dans la scène est déterminée par sa position et le vecteur direction qui indique vers quel endroit elle regarde dans la scène. La vue elle-même est obtenue par projection de la scène dans le repère de la caméra comme sur la figure 2.1.

2. StereoPhoto Maker : `http://stereo.jpn.org/eng/stphmkr/`
3. Anaglyph Maker 3D : `http//www.stereoeye.jp/software/index\_e.html`
4. Stereographic Suite : `http://indasoftware.com/stereo/`
5. 3DMiracles : `http://www.ixtlan.ru/`

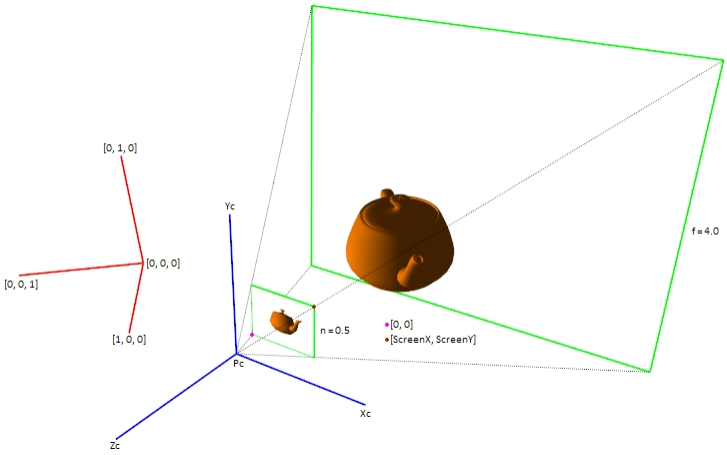

FIGURE 2.1 – Projection d'un objet en trois dimensions sur un support en deux dimensions [6]

2.3 Les flipbooks

Le principe d'un flipbook, ou folioscope en français, est de créer une suite d'images successives d'une scène par rapport à une trajectoire. Il suffira ensuite de mettre toutes ces images dans l'ordre les unes derrière les autres, et de les faire défiler rapidement pour avoir l'impression d'un rendu en relief et en mouvement. C'est également le principe des fichiers d'extension .GIF, qui font défiler une liste d'images.

La création d'un flipbook est possible en créant une animation à l'aide d'un logiciel de manipulation d'objets 3D et en ne capturant que certaines images, par exemple avec Blender. Ainsi l'impression de ces images successives permet de réaliser un flipbook (cf. figure 2.2). En combinant par exemple avec le logiciel Gimp (outil d'édition et de retouche d'image) l'animation peut être obtenue en GIF.

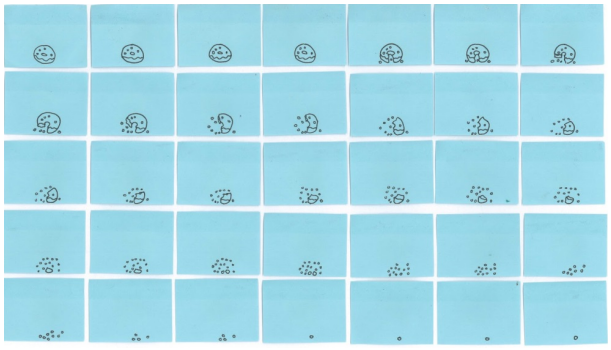

FIGURE 2.2 – Story-board d'un flipbook [7]

6. OpenGL perspective projection : `http://www.3dcpptutorials.sk/index.php?id=2`
7. `http://tracieliu.blogspot.fr/2010/08/flipbook-storyboard.html`

2.4 Les autostéréogrammes

Un autostéréogramme est une image qui cache une visualisation en trois dimensions d'un objet. Cette image est construite de sorte qu'à chaque point de l'objet en trois dimensions soient associés deux points de l'image. L'utilisateur doit observer chaque point d'un couple de points avec un seul œil, ce qui donne l'illusion au cerveau d'observer deux images différentes avec les deux yeux et donc l'incite à traiter cette information comme l'observation d'un objet en trois dimensions, comme illustré par la figure 2.3.

La visualisation en trois dimensions peut être difficile à obtenir, et demande une réelle gymnastique oculaire. Il faut pouvoir fixer le regard en avant ou en arrière de l'image, pour réussir à y voir l'objet caché. La plupart des autostéréogrammes sont observables en vision parallèle, c'est-à-dire qu'il faut faire le point au-delà de l'image pour pouvoir observer l'objet.

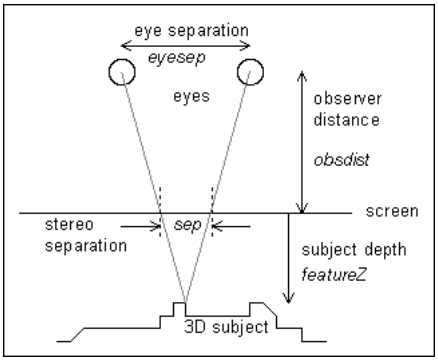

FIGURE 2.3 – Visualisation d'un autostéréogramme en vision parallèle [8]

Pour générer un autostéréogramme, il faut utiliser une carte des profondeurs, ou carte de disparité, de l'objet à dissimuler. Cette carte s'obtient grâce à deux visions d'une même scène prises à deux endroits différents, et permet de mettre en avant les informations sur la profondeur de l'objet. Par exemple, la carte des profondeurs de la figure 2.4 a permis d'obtenir l'autostéréogramme de la figure 2.5.

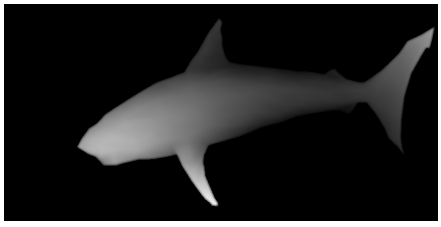

FIGURE 2.4 – Carte des profondeurs [9]

Il existe des algorithmes de génération d'autostéréogrammes très simples , comme celui proposé par Gary Beene [**?**]. Cependant un algorithme aussi peu optimisé produit des autostéréogrammes défecteux, comportant par exemple des échos (répétition d'une partie de l'objet 3D) ou des artefacts (défaut de l'objet 3D observé).

8. `http://www.techmind.org/stereo/geometry.gif`
9. `http://en.wikipedia.org/wiki/Autostereogram`

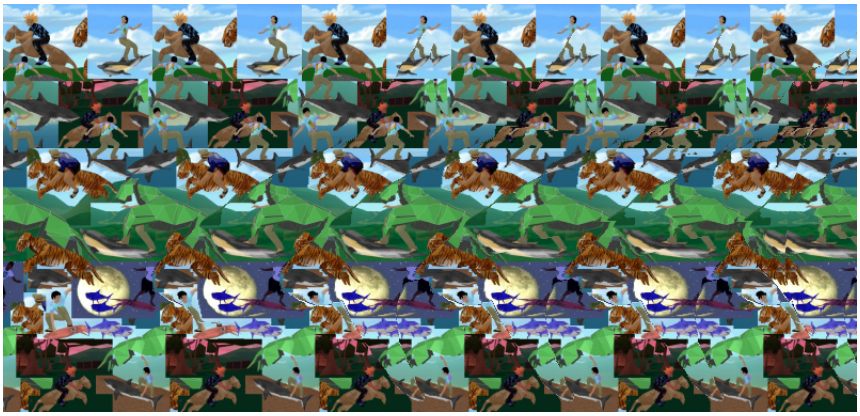

FIGURE 2.5 – Autostéréogramme obtenu [10]

Des algorithmes plus poussés ont été développés, comme celui présenté par Harold W. Thimbleby, Stuart Inglis et Ian H. Witten [**?**], et celui proposé par W. A. Steer [**?**], associés à une réalisation en C. Ils permettent la génération d'autostéréogrammes SIRDS (Single Image Random Dots Stereogram) à partir d'une image 2D. Ces articles présentent également des méthodes de correction de problèmes tels que l'écho ou la gestion des faces cachées (faces de l'objet tridimensionnel visibles par un seul œil). W. A. Steer propose de plus une méthode de génération d'autostéréogrammes utilisant un motif bitmap comme image de base plutôt qu'un nuage de points aléatoires, ce qui pallie à certaines limites des SIRDS comme le manque de détails et la grossièreté des surfaces courbes.

2.5 Les anaglyphes

Un anaglyphe est une image sur laquelle on superpose deux vues, si possible différentes, d'une scène. La meilleure distance entre ces deux visions est la même que celle entre les deux yeux, afin que le cerveau puisse recréer la même vision en trois dimensions que dans la réalité.

Les anaglyphes les plus fréquents sont les anaglyphes dits rouge-cyan. Ils se nomment ainsi car ils sont constitués d'une image sur laquelle on passe un filtre magenta, et une autre avec un filtre cyan (cf. figure 2.6). Pour pouvoir visualiser le relief sur une telle image, on utilise une paire de lunettes rouge-cyan, dont chaque verre est un filtre pour l'une des deux couleurs de l'image. Le plus souvent, le filtre magenta est placé sur l'œil gauche, le cyan sur l'œil droit. En regardant l'image, l'œil gauche ne verra alors que la composante cyan, et inversement pour l'œil droit. Les deux images ayant un léger décalage, le cerveau va percevoir l'image comme si elle était en trois dimensions. Les anaglyphes rouge-cyan sont principalement intéressants sur des images en noir et blanc. En effet, quand il s'agit d'images en couleur, celles-ci sont souvent détériorées par l'usage des filtres, car les couleurs possèdent généralement de plusieurs composantes. A l'inverse, quand l'image est en nuances de gris, l'image n'est pas modifiée, juste mise en relief. Plusieurs types d'algorithmes existent pour créer des anaglyphes depuis des paires d'images. Ils se différencient par la qualité de l'anaglyphe en sortie en diminuant le nombre d'artefacts [**?**] ou en améliorant le rendu pour l'impression [**?**] par exemple.

Le paragraphe précédent concernent la création d'un anaglyphe depuis deux prises de vue d'une même scène avec un léger décalage. Mais il est également possible de le faire à partir d'une image en deux dimensions grâce à l'algorithme de Dubois [**?**], qui définit une vision matricielle de l'anaglyphe (cf. figure 2.7). Dès lors, une matrice de transformation

10. `http://en.wikipedia.org/wiki/Autostereogram`

12. `http://www.david-romeuf.fr/3D/Anaglyphes/TCAnaglypheLSDubois/TransformationCouleursPourAnaglyphe.html`

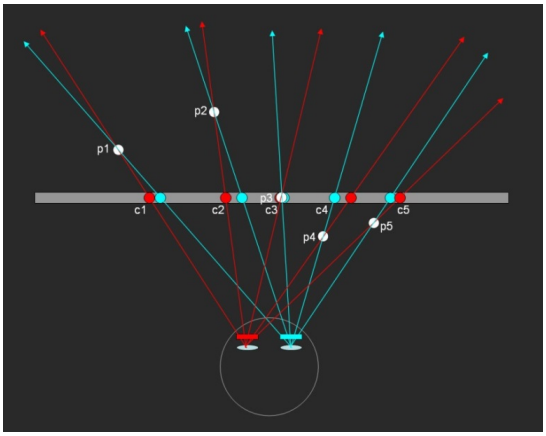

FIGURE 2.6 – Le décalage de la partie rouge et cyan permet à l'œil de percevoir l'image non plus dans un plan XY mais dans l'espace [12]

est introduite, qui permet de passer du taux RGB d'un pixel de l'image originale, aux deux taux RGB des anaglyphes gauche/droite. Les coefficients de cette matrice sont calculés pour satisfaire une bonne restitution de l'image originale, tout en supprimant les rivalités colorées induites par des lunettes bicolores. Ainsi, pour visualiser une image rouge pure, il faudra changer le taux RGB de ce pixel pour que les deux yeux puissent le voir après les filtres. Sinon l'information ne circulera que dans un œil, et l'on perdra la vision stéréoscopique, et donc la notion de profondeur.

FIGURE 2.7 – Rendu final d'une image par l'algorithme de Dubois [13]

13. http://zour.deviantart.com/art/Wernigerode-Boulevard-Dubois-Anaglyph-HDR-3D-276542278

3 Sujet

Le projet concerne la réalisation d'un logiciel permettant d'obtenir des projections en deux dimensions, des anaglyphes, des autostéréogrammes ou encore des flipbooks à partir de scènes virtuelles en trois dimensions.

L'objectif premier est de permettre la visualisation, sur un support en deux dimensions tel qu'un écran d'ordinateur ou une feuille de papier, d'un espace en trois dimensions. A partir de la visualisation d'objets 3D dans une scène il faudra donc réaliser des photographies qui une fois traitées donneront lieu à des anaglyphes, autostéréogrammes ou des animations type flipbook.

Afin d'atteindre cet objectif, un logiciel s'appuyant sur le moteur 3D OpenGL devra être réalisé. Il permettra la création d'une scène où s'insèreront des objets dont la position, la taille et l'orientation seront paramétrables. Une caméra permettra de se déplacer dans la scène, de s'en rapprocher ou s'en éloigner.

Une fois la scène mise en place, il faudra pouvoir prendre des photographies de celle-ci sous différents angles afin d'obtenir, après application d'algorithmes de traitement d'images :

- des anaglyphes rouge-cyan, qui permettront une visualisation en trois dimensions grâce à des lunettes adaptées ;
- des stéréogrammes, qui sont des images dissimulant un contenu qui apparaît quand on fixe le dessin de façon spécifique ;
- des flipbooks ou images animées, correspondant à une succession d'images suivant une trajectoire qui permettent en les faisant défiler de donner une impression de mouvement.

4 Cahier des besoins

4.1 Besoins fonctionnels

4.1.1 La scène

Cette partie montre l'ensemble des cas d'utilisation relatifs à la scène.

Création de la scène :

Au démarrage du logiciel, il sera possible de créer une nouvelle scène ou d'en charger une préexistante. Celle-ci permettra d'afficher et disposer plusieurs objets 3D avant d'être sauvegardée pour une utilisation ultérieure.

Sauvegarde de la scène :

On distinguera deux types de sauvegardes indépendantes entre elles : la sauvegarde automatique et la sauvegarde manuelle. La première s'effectuera après chaque modification apportée avec pour but d'éviter la perte de données en cas de crash du logiciel, elle se fera de manière discrète (c'est à dire en arrière-plan sans figer le logiciel le temps de la sauvegarde). La seconde se fera au format XML (dont le prototype est détaillé ultérieurement) sur demande de l'utilisateur. Il sera possible de choisir entre une importation des objets utilisés dans le répertoire courant de la scène ou d'une simple sauvegarde des chemins d'accès aux modèles 3D. Par ailleurs, les deux types de sauvegarde étant indépendants, une sauvegarde manuelle n'écrasera pas une sauvegarde automatique.

Chargement de la scène :

Comme vu précédemment, lors de la sauvegarde, les données relatives aux objets de la scène seront stockées dans un fichier au format XML. A l'ouverture du logiciel, l'utilisateur pourra choisir d'ouvrir une scène qu'il aura sauvegardée précédemment. Dans ce cas, les données enregistrées seront récupérées pour recréer la scène telle qu'elle était avant la fermeture du logiciel.

Suppression de la scène :

Le logiciel ne proposera pas directement de méthode pour supprimer une scène. Pour cela, il faudra directement aller supprimer le dossier associé à la sauvegarde à l'intérieur du dossier contenant le logiciel.

4.1.2 Les objets

Une fois la scène créée ou chargée, l'utilisateur aura la possibilité d'y placer des objets.

Chargement des fichiers objets :

Les objets en trois dimensions utilisés seront sous la forme de fichiers objets. Deux types seront acceptés : les fichiers d'extension .OBJ (version 3.0, ASCII) et .PLY (versions 1.0, ASCII et binaire). Ils devront permettre de définir des objets à facettes triangulaires. Les objets pourront être ceux de l'utilisateur ou de la bibliothèque implémentée dans le logiciel.

Ces objets ne devront contenir aucun trou, par exemple une face absente sur un cube ou un triangle non généré par le fichier de chargement. Le fonctionnement du logiciel n'est garanti que sur des objets topologiquement valides.

Si un fichier à charger possède trop de polygones, un algorithme de décimation de faces permet d'en réduire le nombre jusqu'à un certain seuil (cf. prototype). Ce traitement n'est pas automatique et l'utilisateur pourra ou non l'appliquer.

Placement d'un objet :

Après le chargement de l'objet il sera possible de le placer dans la scène via les trois translations du repère 3D. Ce placement n'est pas définitif et pourra être modifié ultérieurement lors de la constitution de la scène.

Modification d'un objet :

Trois caractéristiques d'un objet pourront être modifiées : son emplacement, son orientation et sa taille.

Modification de l'orientation d'un objet :

L'orientation d'un objet dans la scène pourra être modifiée en fonction des trois axes de rotation usuels.

Modification de la taille d'un objet :

L'objet pourra être agrandi ou réduit selon le besoin.

Mode de modification d'un objet :

Le placement et le paramétrage de l'objet (orientation, taille ...) n'est pas définitif et peut-être revu par la suite via un mode de modification après avoir sélectionné l'objet à modifier. Toute sortie de ce mode de modification entraînera une sauvegarde automatique de la scène.

Sélection d'un objet :

Il sera possible de sélectionner un objet pour réaliser des modifications.

Suppression d'un objet :

Il sera également possible, après sélection, de supprimer un ou plusieurs objets de la scène. Le ou les objets ainsi supprimés ne seront plus référencés dans la sauvegarde.

4.1.3 La caméra

La caméra possède un repère qui lui est propre et une distance correspondant à celle entre l'origine du repère de la scène (centre de rotation de la caméra) et le sien.

Observation de la scène :

La caméra permettra de se déplacer dans la scène en suivant les trois axes de rotation et les trois axes de translation usuels, ainsi qu'un zoom. Si l'on choisit de se déplacer relativement aux axes de translation, le centre de rotation de la caméra sera modifié.

Type de projection :

Deux types de projections seront disponibles : la projection orthographique (qui sera l'option par défaut) et la projection en perspective.

Zoom de la scène :

Il sera possible de se rapprocher ou s'éloigner de la scène via une fonctionnalité de zoom. Il ne sera pas possible d'atteindre le centre de rotation de la caméra.

Remise à zéro du centre de la caméra :

La position de la caméra pourra, à tout moment, être réinitialisée pour se retrouver dans les conditions initiales de visionnage de la scène.

4.1.4 Le passage en deux dimensions

Mode de passage en deux dimensions :

Différents rendus seront accessibles :

- Photographie résultant d'une projection sans traitement de la scène en trois dimensions.
- Anaglyphe rouge-cyan à partir de paires d'images en noir et blanc avec traitement possible pour l'impression.
- Autostéréogramme SIRDS selon l'algorithme de Witten, Inglis et Thimbleby.
- Flipbook d'une rotation complète ou non autour d'un objet ou d'une scène.

Pour les images générées par les modes présentés ci-dessous, il faudra choisir entre une qualité de 75 ou 300 pixels par pouce.

Mode photographie :

Le mode photographie permettra d'obtenir une image proche de la scène virtuelle créée. Pour ce mode, des informations concernant la qualité de l'image et sa taille devront être fournies.

Mode analgyphe :

Le mode anaglyphe permettra d'obtenir une image rouge, une image cyan, ainsi qu'une image superposant les deux. Pour ce mode, des informations concernant la qualité de l'image et sa taille devront être fournies.

Mode autostéréogramme :

Le mode autostéréogramme permettra d'obtenir un autostéréogramme fixe à une image. Pour ce mode, des informations concernant la qualité de l'image et sa taille devront être fournies.

Mode flipbook :

Le mode flipbook permettra d'obtenir un ensemble d'image sur une trajectoire donnée. Pour ce mode, des informations concernant la qualité de l'image et sa taille, mais également des informations nécessaires au lancement et à la construction du flipbook devront être fournies par l'utilisateur. Ainsi, il devra tout d'abord choisir une trajectoire parmi les trois axes de rotation, les trois axes de translation ou le zoom, puis un point de départ. Au choix, il pourra alors sélectionner un point d'arrivée et un nombre d'images, ou un nombre d'image et un angle pour connaître la mesure entre deux prises. Une barre de progression indiquera le nombre prises de vues faites par rapport au total. Si le temps d'attente est trop long, il sera toujours possible d'annuler l'action et redéfinir les paramètres.

Affichage du rendu :

Les algorithmes des modes présentés ci-dessus permettront d'obtenir un rendu correspondant aux attentes de l'utilisateur. Ce dernier sera ensuite affiché dans une nouvelle fenêtre contenant le ou les produits demandé qu'il sera possible de sauvegarder avant de quitter.

Enregistrement du rendu :

Si l'utilisateur choisit d'enregistrer les images générées par les algorithmes, les rendus qu'il souhaite sauvegarder devront être choisis parmi les rendus obtenus.

Enregistrement de la photographie :

Pour le cas simple de la photographie, il faudra nommer le fichier et donner son emplacement avant qu'il ne soit enregistré au format PNG.

Enregistrement de l'anaglyphe :

Les images rouge et cyan pourront être enregistrées séparément ou non. Deux options de rendus seront disponibles : pour l'impression et pour l'ordinateur. Elles permettront d'optimiser l'affichage et l'utilisation des lunettes dans les deux cas.

Enregistrement de l'autostéréogramme :

Pour l' autostéréogramme, il faudra nommer le fichier et indiquer son emplacement avant de le sauvegarder au format PNG.

Enregistrement du flipbook :

Dans le cas du flipbook, les images pourront être enregistrées une à une au format PNG, par tranche au format PNG ou sous forme d'animation GIF. Dans le cas d'une sauvegarde séparée de chaque image il faudra les nommer et donner leur chemin d'accès. Par tranche il faudra indiquer la première et la dernière image ainsi qu'un nom commun et un répertoire. Pour l'animation il faudra la nommer et donner son chemin d'accès.

4.2 Besoins non fonctionnels

4.2.1 La fluidité d'affichage

Pour que l'affichage soit fluide la machine devra avoir au minimum 2Go de mémoire vive, un processeur récent avec au moins 2 cœurs cadencés au minimum à 2GHz et les conditions suivantes devront être validées :

- Pour une machine avec processeur graphique type Intel HD Graphics 4000 ou plus récent, au maximum 100 000 points pour un affichage à 25 fps, ou 150 000 points pour un affichage à 20 fps ;
- Pour une machine avec une carte graphique type NVIDIA GTX 720 ou plus récente, au maximum 350 000 points pour un affichage à 25 fps, ou 500 000 points pour un affichage à 20 fps.

Test de validation : ouverture de plusieurs modèles 3D, présentés en annexe, dans Meshlab sur différentes machines. Les résultats sont également disponibles en annexe.

4.2.2 Fluidité d'obtention des rendus

A l'heure actuelle on ne peut garantir un nombre de points assurant l'obtention d'un rendu en un temps raisonnable. C'est pourquoi une barre de progression s'affichera pour que l'utilisateur ait une idée du temps d'attente avant l'affichage du résultat.

4.2.3 Portabilité du logiciel

La portabilité rend le logiciel accessible à un plus grand nombre de personnes. Les systèmes d'exploitation visés seront Windows XP, Windows Seven, Windows 8.1, Ubuntu 14.04, Fedora 20 et Debian Jessie, en plateforme 32 bits et 64 bits.

Test de validation : un prototype, qui pourra être utilisé comme base du projet, a été exécuté sur l'ensemble de ces systèmes d'exploitation (cf. la partie 6 Prototype test).

4.2.4 Extensibilité du logiciel

Le client souhaite que l'extensibilité du logiciel soit facilitée par son architecture. En effet, il envisage par la suite d'ajouter de nouvelles fonctionnalités ou de nouveaux algorithmes au logiciel, et demande à ce que de telles modifications soient aisément envisageables.

La facilitation de cette extensibilité sera permise par l'utilisation d'une architecture spécifique. Celle-ci est détaillée dans la partie 7 Architecture du projet de ce cahier des charges.

4.3 Contraintes

4.3.1 Superposition des objets dans la scène

Pour faciliter le traitement d'une scène, on considèrera que deux objets placés dans une scène ne pourront pas être superposés. On parle ici d'intersection des boîtes englobant des objets et non de la gestion des parties non visibles des objets lors de l'affichage.

4.3.2 Utilisation des bibliothèques

Afin de minimiser les dépendances et valider le besoin en portabilité les bibliothèques Qt et OpenGL seront utilisées. CMake permettra de générer les scripts de compilation quel que soit la plateforme.

4.3.3 Langage de programmation

Le langage C++ sera utilisé.

4.3.4 Open Source

Pour permettre l'utilisation du logiciel et son extensibilité à quiconque souhaiterait le modifier, on choisira de laisser le code du projet en Open Source. Pour garantir cela, on s'assura bien que l'on dispose des droits nécessaires pour n'importe quel algorithme ou fragment de code utilisé au cours de ce projet.

5 Maquette

Cette partie montrera l'ensemble des maquettes de la future interface du logiciel.

5.1 Fenêtre principale

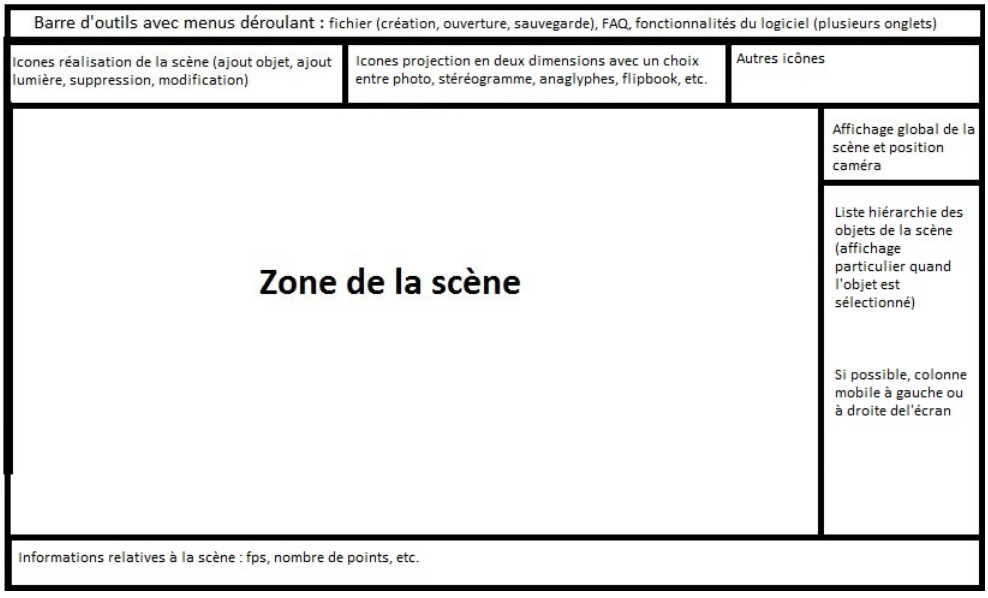

FIGURE 5.1 – Fenêtre principale

La fenêtre principale 5.1 se détaille comme suit :

- Barre d'outils avec menus déroulants

  Ce bandeau supérieur correspond à un bandeau supérieur de logiciel classique, comme on le trouve par exemple dans le logiciel Meshlab.

  L'intérêt de ces menus déroulants est de proposer l'accès à l'ensemble des fonctionnalités du logiciel. Parmi ces menus se trouvent quelques redondances avec des icônes présentes dans d'autres parties de la fenêtre principale, afin que l'utilisateur puisse choisir l'utilisation du logiciel qui lui convient le mieux.
- Icônes de réalisation de la scène

  Les icônes permettant de réaliser la gestion des objets et des lumières de la scène : il sera possible d'ajouter un objet ou de le modifier à partir de ces icônes.
- Icônes de projection en deux dimensions

  L'ensemble des icônes permettant d'obtenir l'un des rendus possibles à l'aide du logiciel, c'est-à-dire quatre icônes pour : une projection en deux dimensions classique, la création d'un anaglyphe, la création d'un stéréogramme et la création d'un flipbook.

  L'ensemble de ces icônes permettra d'ouvrir une nouvelle fenêtre et de choisir les paramètres de l'image à obtenir.
- Autres icônes

  Cette zone pourra contenir d'autres icônes, qui seront triées et rassemblées en fonction de leurs cas d'utilisation.
- Zone de la scène

  Vu de la scène du point de vue de la caméra. L'utilisateur pour s'y déplacer librement via la caméra pour pouvoir l'observer depuis différents angles. Il pourra également sélectionner des objets pour les déplacer, modifier leur orientation, les redimensionner ou les supprimer.
- Liste des objets de la scène

  L'ensemble des objets présents dans la scène listés : l'utilisateur pourra cliquer sur le nom d'un objet afin de le sélectionner et de pouvoir agir dessus.
- Informations relatives à la scène

  Des informations intéressantes pour l'utilisateur par rapport à la scène et à son affichage seront affichées : le nombre d'objets dans la scène, le nombre de points, ou encore la qualité d'affichage en frames par seconde. Cette zone pourra également servir à l'affichage de messages informatifs ou de messages d'erreurs à l'intention de l'utilisateur.

5.2 Fenêtre de chargement d'un objet

L'image 5.2 schématise la fenêtre de chargement d'un nouvel objet. L'utilisateur du logiciel doit d'ores et déjà avoir ouvert une scène. Il pourra ensuite choisir d'y insérer un nouvel objet grâce à la fenêtre ci-dessus, en choisissant un modèle à ouvrir ainsi que le nom de l'objet dans la scène.

Pour le choix du modèle 3D, l'utilisateur pourra utiliser soit la bibliothèque du logiciel soit ses propres fichiers. Toutefois, seules les extensions OBJ et PLY seront acceptées.

Une fois toutes les informations entrées, l'utilisateur pourra cliquer sur le bouton Valider pour passer en mode Modification de l'objet et placer son objet dans sa scène.

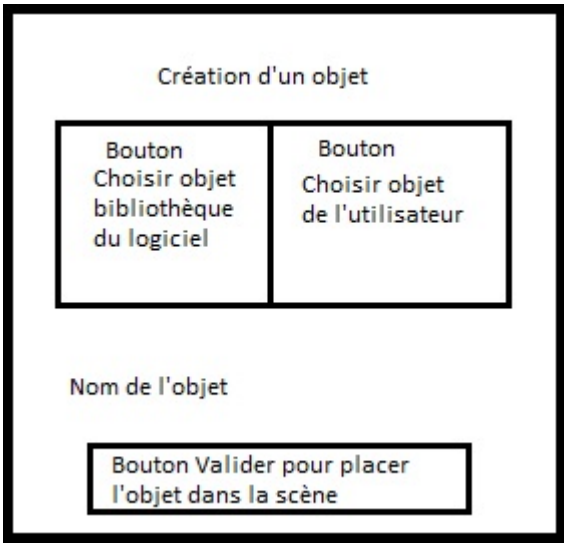

FIGURE 5.2 – Fenêtre de chargement d'un objet

5.3 Interface de modification d'un objet

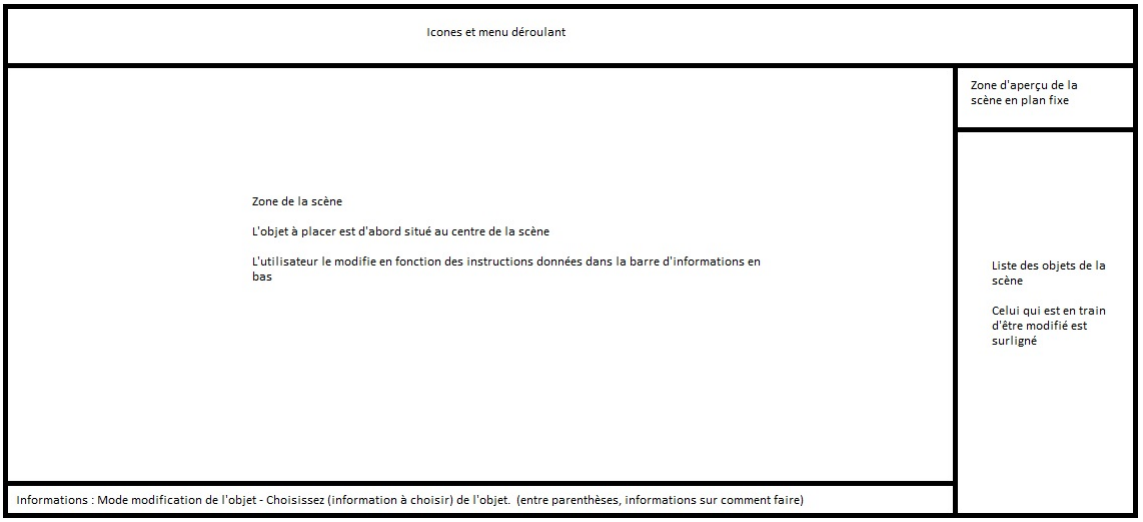

FIGURE 5.3 – Interface de modification d'un objet

Une fois qu'il aura choisi le modèle à placer, l'utilisateur verra de nouveau la fenêtre principale s'afficher, et devra suivre les informations indiquées dans la barre Informations en bas de la fenêtre pour modifier la position, l'orientation et la taille de l'objet.

S'il souhaite de nouveau effectuer des modifications sur son objet, il devra repasser en mode Modification de l'objet, et retrouvera la même interface 5.3. Il devra à chaque fois obligatoirement passer par les trois modes de modification :

position, orientation, taille.

5.4 Fenêtre de choix des options du rendu

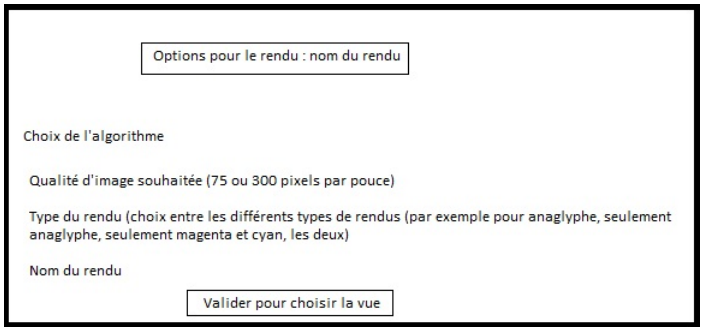

FIGURE 5.4 – Choix des options du rendu

La fenêtre 5.4 s'ouvre lorsque l'utilisateur clique sur le bouton de création d'une projection, d'un anaglyphe, d'un autostéréogramme ou d'un flipbook. Il devra alors choisir l'algorithme à utiliser pour le rendu souhaité (un ou plusieurs algorithmes pourront être proposés), la qualité d'image qu'il souhaite obtenir, ainsi que les types de retour qu'il souhaite obtenir si plusieurs choix existent. Par exemple, dans le cas d'un anaglyphe, le retour pourra contenir ou bien uniquement l'anaglyphe, ou bien les deux vues rouge et cyan, ou bien les deux.

L'utilisateur choisira ensuite le nom du rendu et appuiera sur le bouton "Valider" pour passer en mode de sélection de la vue. Ce mode reviendra sur la fenêtre principale du logiciel, et des informations seront données à l'utilisateur pour sélectionner la vue qu'il souhaite.

Toutes les informations seront obligatoirement complétées pour pouvoir accéder au mode de sélection de la vue.

5.5 Fenêtre d'affichage du rendu

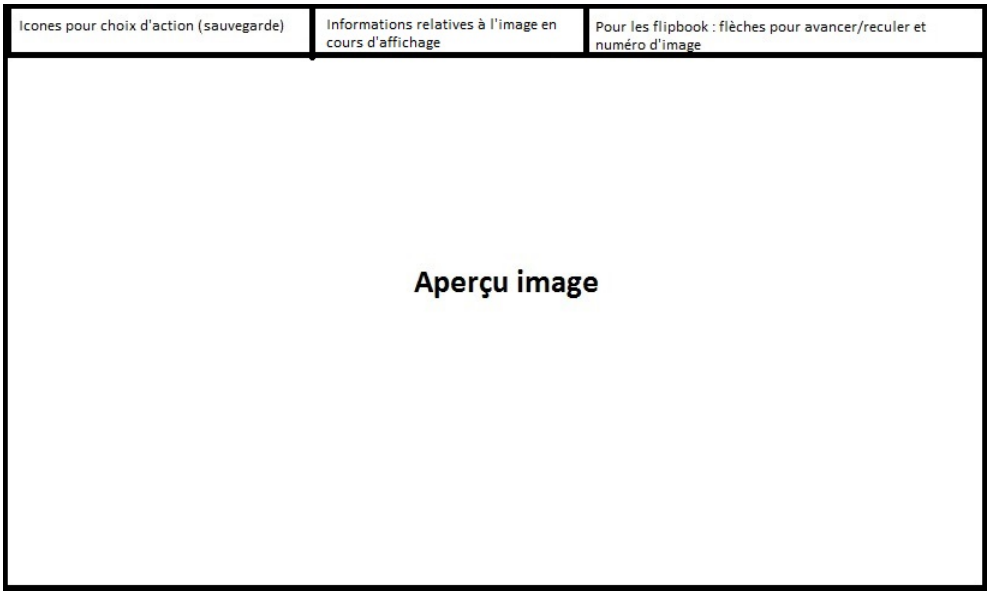

FIGURE 5.5 – Affichage du rendu

Une fois l'action paramétrée, le rendu s'affiche dans la fenêtre 5.5 dont les options sont détaillés ci-dessous.

- Zone de choix d'action
  Cette zone contiendra des icônes pour indiquer à l'utilisateur les choix qui s'offrent à lui. On trouvera par exemple une icône pour l'enregistrement du rendu, qui ouvrira une nouvelle fenêtre pour laisser le choix à l'utilisateur de ce qu'il souhaite enregistrer.
- Zone d'affichage des informations
  Cette zone concernera uniquement l'image en cours d'affichage dans la zone d'aperçu. On détaillera ici la taille de l'image et sa qualité en pixels par pouce.
- Zone de défilement
  Cette zone contiendra des flèches de déroulement vers la gauche ou vers la droite pour permettre à l'utilisateur de voir l'ensemble des images obtenues, ainsi qu'un ou plusieurs boutons pour pouvoir afficher un GIF ou le stopper.
- Zone d'aperçu de l'image
  S'affichera ici l'image obtenue après la requête d'un utilisateur.

6 Prototype test

6.1 Format de sauvegarde d'une scène

La sauvegarde d'une scène se fera au format XML. La racine scène contiendra une suite d'objets et une caméra. Un objet aura pour attribut son format et son répertoire d'accès et sera caractérisé par une translation, une rotation et un redimensionnement. La caméra sera caractérisée par une translation et une rotation. Le fichier sera validé par un schéma XML tout au long de la phase de développement et lu via des requêtes XPath.

Le schéma XML ainsi qu'un fichier XML de test ont été réalisés et sont disponibles en annexe.

6.2 Test de fluidité

Le logiciel Meshlab a été utilisé sur différentes machines avec différents modèles 3D pour déterminer un rapport FPS / nombre de points convenable. Les modèles 3D utilisés contiennent entre 40000 et 14 millions de points.

6.3 Test de portabilité

Pour s'assurer de la portabilité de QT et OpenGL sur Windows et Linux, un affichage simple d'un modèle 3D dans une fenêtre QT, sans colorations de face, a été fait et testé sur différentes distributions (fig. 6.1). Les versions 5 de QT et 3.3 d'OpenGL ont été utilisées pour ce prototype et seront utilisées dans la suite du projet. Il a été testé avec des modèles de Stanford comme le dragon, le lapin ou le bouddha.

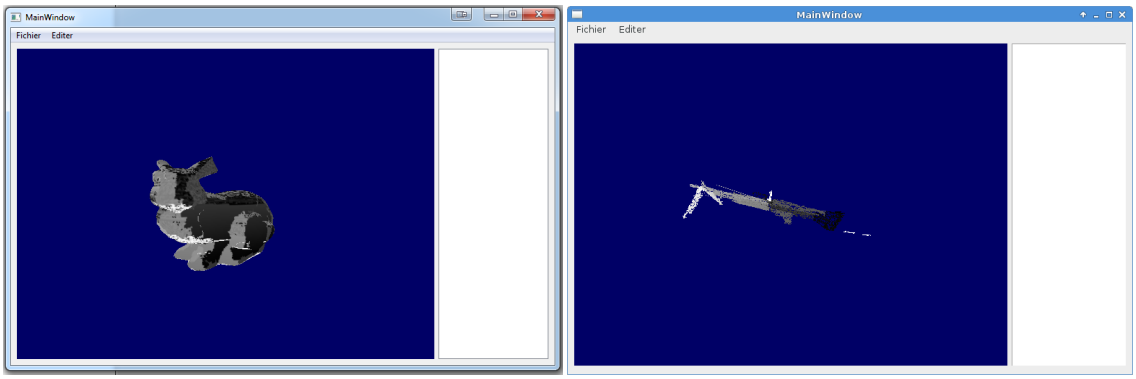

FIGURE 6.1 – Affichage du modèle 3D de lapin sur Windows 7 et M60 sous Linux - Fedora 20

6.4 Réduction du nombre de polygones d'un modèle 3D

Un algorithme permet de réduire le nombre de faces d'un modèle 3D qui en possèderait trop et serait donc impossible à afficher. Il supprime des triangles et fusionne les sommets autour, jusqu'à ce que le nombre de polygones tombe en dessous d'un certain seuil (fig. 6.2).

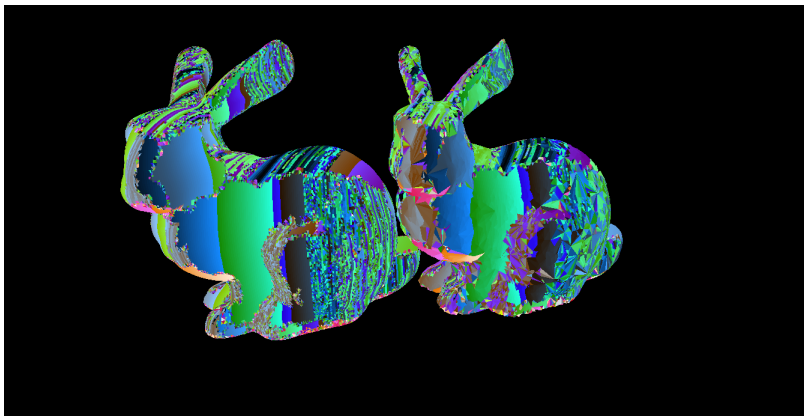

FIGURE 6.2 – À gauche modèle 3D d'origine, à droite modèle après suppression de faces

Ce traitement est effectué lors du chargement et est relativement rapide pour les objets (fig. 6.3).

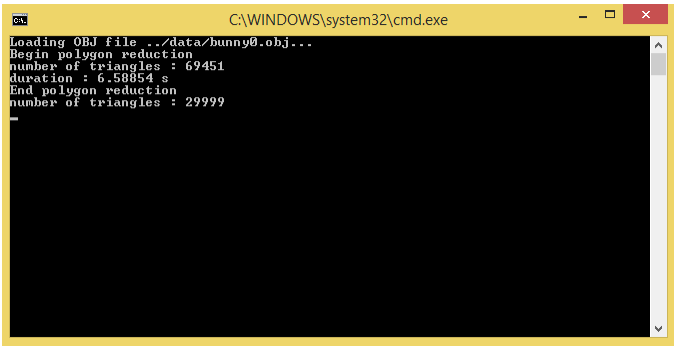

FIGURE 6.3 – Un test effectué avec l'algorithme sur le modèle de la figure 6.2 montre qu'il met moins de 7 secondes pour supprimer 40000 faces

7 Architecture du projet

Dans cette partie, nous présenterons un ensemble de diagrammes de séquences et de diagrammes de classes présentant le fonctionnement prévisionnel du logiciel et l'architecture qui en découle. Cette architecture tiendra compte du besoin d'extensibilité du logiciel.

Le diagramme de paquetage présenté 7.1 représente l'architecture globale du futur logiciel. Il montre que l'interface graphique sera le point d'entrée pour l'utilisateur. En effet, c'est celle-ci qui redirigera les actions demandées vers soit le créateur d'image, soit le chargeur. Ce dernier sera toujours utilisé pour accéder à son attribut Scène, car il sera responsable de la création de celle-ci et de la transformation des fichiers objets en modèles 3D.

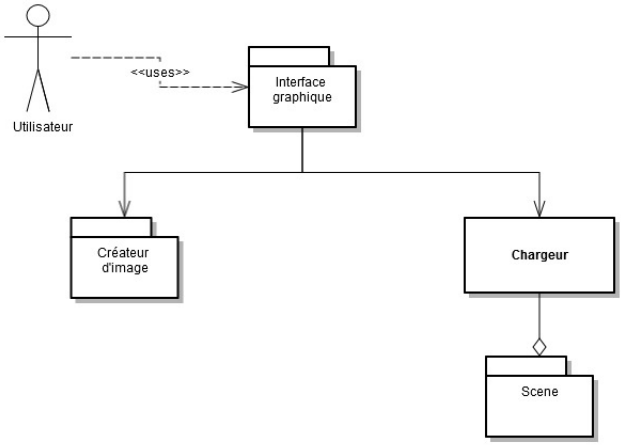

FIGURE 7.1 – Diagramme de paquetages

L'ensemble des diagrammes de séquence et des diagrammes de classe présentés par la suite auront été créés grâce à la version d'essai du logiciel en ligne Gliffy [14].

7.1 Diagramme de séquences

Les diagrammes de séquence permettent de simuler les différentes communications qui auront lieu entre les classes du logiciel.

On considèrera que lorsqu'il commence à utiliser le logiciel, la première action d'un utilisateur va être de créer une scène. Grâce à l'interface graphique correspondante, qui a été présentée dans la partie Interface de ce cahier des Charges, l'utilisateur demande à l'interface de créer une scène. Celle-ci passera par l'intermédiaire d'un chargeur, déjà initialisé, qui devra créer l'instance correspondante soit à partir de rien, soit à partir d'une scène déjà existante. Une fois sa tâche effectuée, le chargeur valide à l'interface qu'il a bien répondu à sa demande, et celle-ci peut modifier l'interface pour afficher la fenêtre principale, qui contient désormais la scène de travail.

7.1.1 Création d'un objet

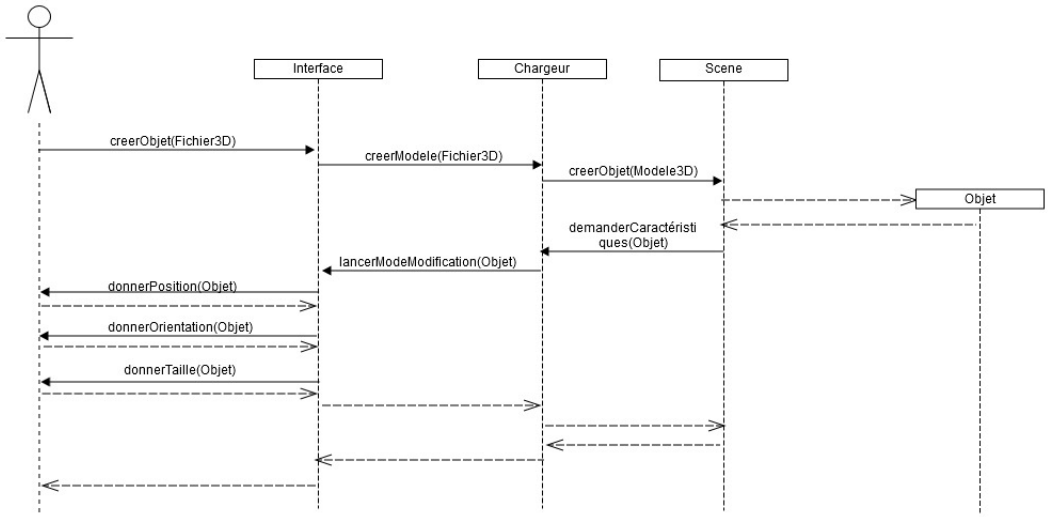

FIGURE 7.2 – Création d'un objet

Une fois la scène de travail créée, l'utilisateur pourra y ajouter des objets s'il le souhaite (cf. 7.2). Pour cela, il va une fois de plus devoir communiquer avec l'interface graphique correspondante.

Une fois qu'il aura choisi un fichier objet d'extension .OBJ ou .PLY, l'interface enverra celui-ci au chargeur qui vérifiera sa validité et le transformera en un modèle 3D. Ce dernier sera transmis à la scène, qui se chargera de créer l'objet en plaçant les points du modèle par rapport à son origine. Ce premier objet généré sera initialisé en position centrale de la scène, avec une taille et une orientation de base définie par ses points. Enfin, une confirmation sera envoyée au chargeur, puis à l'interface, qui affichera la fenêtre correspondant au mode modification de l'objet.

14. `http://www.gliffy.com/`

Durant ce mode, l'interface demandera à l'utilisateur de modifier la position, l'orientation et la taille de l'objet s'il le souhaite, et appellera des méthodes soient publiques de l'objet, soit en passant par le chargeur puis la scène, pour effectuer les changements demandés.

Ce diagramme de séquence montre l'importance du chargeur dans l'intéraction de l'interface avec la scène. C'est lui qui s'occupera de créer la scène et de lui transmettre les demandes de l'utilisateur.

7.1.2 Déplacement dans la scène

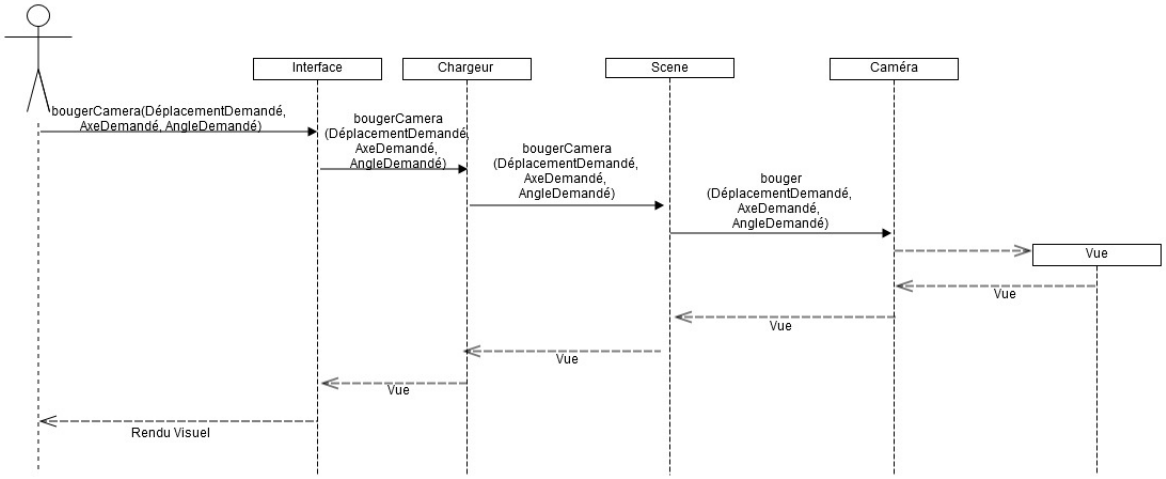

FIGURE 7.3 – Déplacement de la caméra

Quand il aura placé des objets dans sa scène, l'utilisateur pourra avoir envie de s'y déplacer pour observer les différents angles de vue. Pour cela, grâce à une action sur l'interface, il va demander à la caméra de se déplacer (cf. 7.3). La première demande sera donc initialisée par l'utilisateur, qui demandera le déplacement de la caméra à l'interface, selon un type (rotation, translation, homothétie), un axe (horizontal, vertical) et un angle ou une valeur donnée.

La caméra étant un attribut de la scène, l'interface se chargera de transmettre l'information au chargeur, qui lui-même l'enverra à la scène. Cette dernière se chargera d'appeler la fonction adéquate de son attribut qui génèrera une nouvelle vue correspondant à la projection de la scène depuis l'angle demandé par l'utilisateur. Cette vue sera renvoyée étape par étape jusqu'à l'interface, qui se chargera de la transformer en un rendu visualisable par l'utilisateur dans la zone de la scène.

7.1.3 Création d'un autostéréogramme

De la même façon que pour l'anaglyphe, c'est par l'intermédiaire de l'interface et du créateur d'image que l'utilisateur va créer son rendu (cf. 7.4). Pour un autostéréogramme toutefois, un autre type de rendu sera également utilisé : la carte des profondeurs. C'est l'autostéréogramme qui demandera au créateur d'image de la créer pour lui, afin qu'il puisse l'utiliser pour générer l'autostéréogramme qu'il renverra vers l'utilisateur.

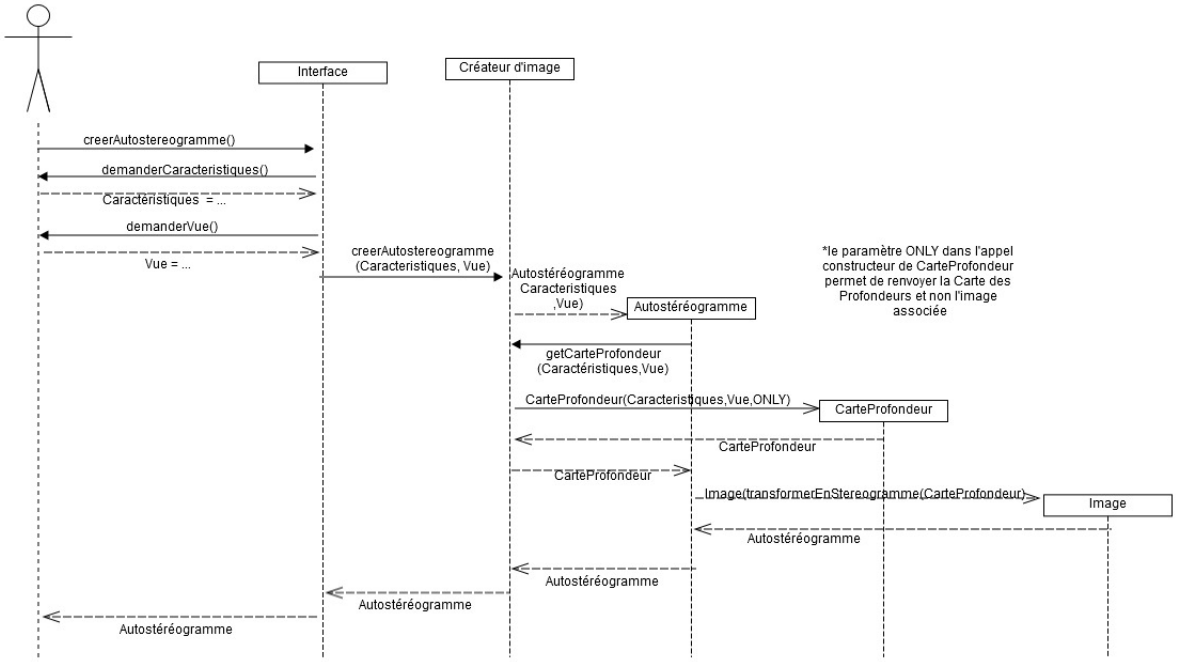

FIGURE 7.4 – Création d'un autostéréogramme

7.1.4 Création d'un flipbook

Une fois de plus, le passage entre l'utilisateur et le flipbook se fait par l'intermédiaire de l'interface et du créateur d'image (cf. 7.5). Cette fois-ci toutefois, la génération se fera à l'aide d'une boucle qui permettra de remplir un vecteur d'image, qui sera renvoyé jusqu'à l'interface. Celle-ci se chargera de pouvoir afficher toutes les vues les unes après les autres dans la fenêtre d'affichage.

Les arguments deplacementDemandé, axeDemandé et angleVoulu qui sont utilisés en paramètre de la fonction prendreVue sont issus des caractéristiques choisies par l'utilisateur. Cette fonction change temporairement la position de la caméra avant de la replacer à sa position initiale une fois la prise de vue faite.

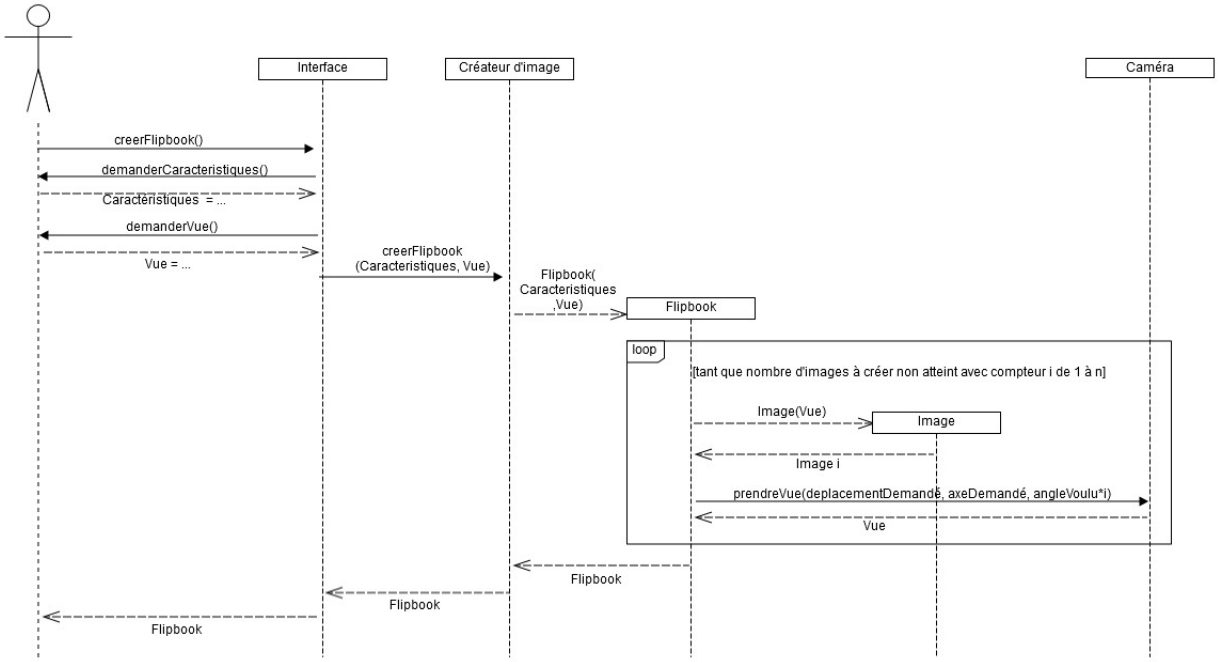

FIGURE 7.5 – Création d'un flipbook

7.2 Diagrammes de classes

A partir des diagrammes de séquences présentés précédemment, l'architecture du projet a pu être déterminée. La version finale de celle-ci dépendra grandement des bibliothèques OpenGL et Qt et des différentes utilisations de leurs modules.

7.2.1 Paquetage Création

L'architecture du paquetage de création a été pensée pour que de futures extensions puissent être insérées à ce niveau du logiciel (cf. 7.6). En effet, le créateur d'image génère un rendu, qui peut être l'un des types situés en dessous. Si l'on souhaite ajouter un nouveau rendu possible, il suffit donc de le placer au même niveau que les interfaces Flipbook, Autostéréogramme, etc. De la même façon, si l'on souhaite ajouter un algorithme différent pour obtenir l'un des rendus, il suffit de l'ajouter en-dessous de la classe virtuelle correspondante. Ainsi cette réalisation répond au besoin d'extensibilité.

Toutes les classes sont en relation avec la caméra par l'intermédiaire de la scène, de façon à pouvoir récupérer les différentes vues nécessaires à la création des rendus.

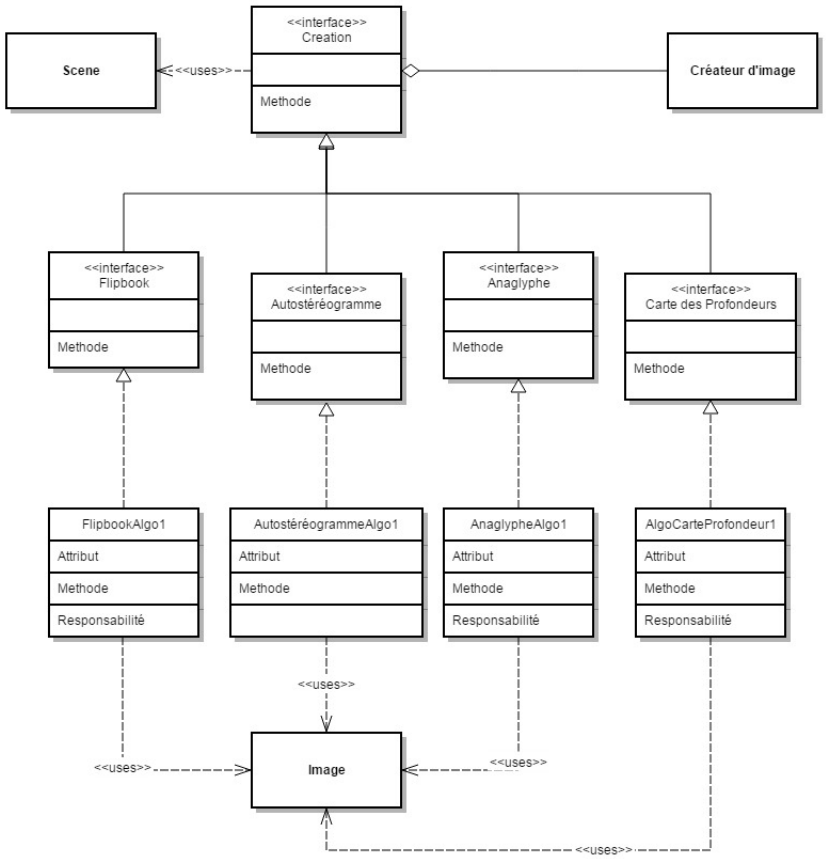

FIGURE 7.6 – Architecture du paquetage Création

7.2.2 Paquetage Scène

On considèrera pour le logiciel la scène comme un ensemble d'objets et une caméra. C'est la scène qui gèrera elle-même l'accès à ses différentes composantes (cf. 7.7). La caméra utilisée, qui sera liée à la caméra de la bibliothèque OpenGL, génèrera des vues qui pourront être utilisées pour l'affichage de la scène ou la création de rendus. Le contenu réel de ces vues et leur utilisation dépendront de la bibliothèque OpenGL.

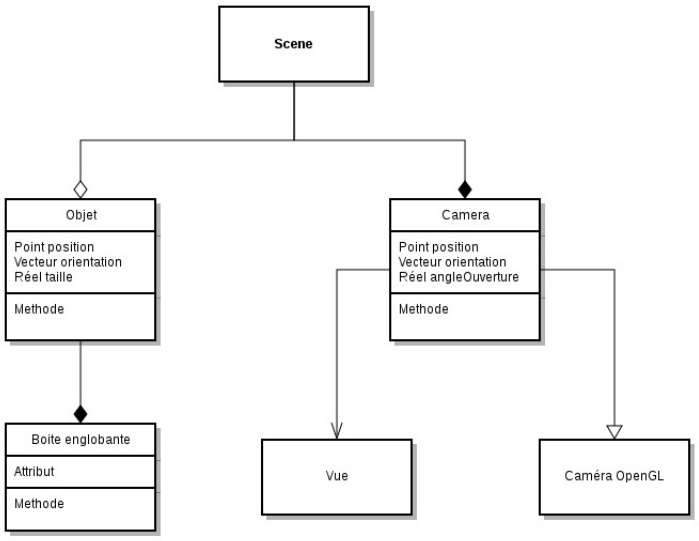

FIGURE 7.7 – Architecture du paquetage Scène

7.2.3 Paquetage Interface

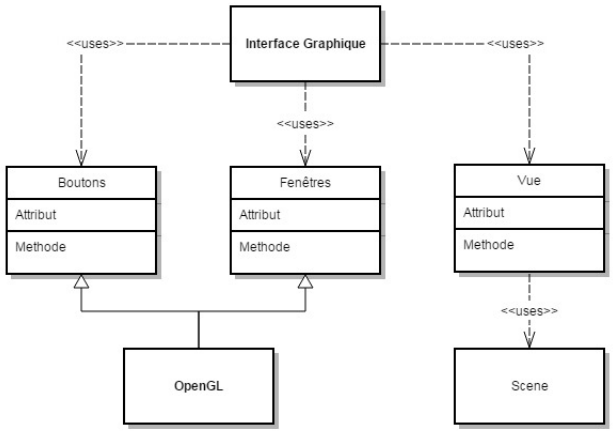

FIGURE 7.8 – Architecture du paquetage Interface

Le paquetage Interface est le paquetage utilisé par l'utilisateur pour interagir avec le logiciel. L'interface sera constituée de boutons et de fenêtres, qui seront des produits de la bibliothèque Qt et du module OpenGL pour Qt (cf. 7.8). L'affichage de la scène se fera à partir des vues générées par la Caméra de la Scène, comme expliqué dans la partie précédente.

La création de l'interface sera pensée pour que l'ajout de boutons et de fonctionnalités au logiciel se fasse de façon simple.

Annexes

A Modèles utilisés pour les prototypes

Le tableau A.1 présente les modèles utilisés en annexe B.

Nom du modèle	Faces	Sommets	Lien de téléchargement
bunny000.ply	79 312	40 256	`http://graphics.stanford.edu/pub/3Dscanrep/bunny.tar.gz`
cart_obj.obj	162 552	87 909	`http://www.oyonale.com/downloads/cart_obj.zip`
pan_obj.obj	205 697	103 032	`http://www.oyonale.com/downloads/casserole_obj.zip`
extincteur_obj.obj	300 632	151 425	`http://www.oyonale.com/downloads/extincteur_obj.zip`
M60_obj.obj	344 140	174 476	`http://www.oyonale.com/downloads/M60_obj.zip`
dragon_vrip.ply	871 414	437 645	`http://graphics.stanford.edu/pub/3Dscanrep/dragon/dragon_recon.tar.gz`
happy.ply	1 087 716	543 652	`http://www.cc.gatech.edu/data_files/large_models/happy.ply.gz`
blade.ply	1 765 388	882 954	`http://www.cc.gatech.edu/data_files/large_models/blade.ply.gz`
lucy.ply	28 055 742	14 027 872	`http://graphics.stanford.edu/data/3Dscanrep/lucy.tar.gz`

FIGURE A.1 – Les modèles utilisés

B Résultats des tests de fluidité

Les résultats de fluidité obtenus avec les modèles de la section A sont présentés dans les tableaux B.1 et B.2. Parmi les machines ayant servi pour les tests, deux sont relativement anciennes et les résultats s'en ressentent (voir tableau B.1, première et troisième machines).

Version de Meshlab Système Carte graphique	V1.3.4 Windows 7 Intel HD Graphics 4000	V1.3.2 Debian Jessie Intel HD Graphics 4000	V1.3.3 Debian Jessie/Sid Intel Mobile Series 4
bunny.ply	54 à 64	-	-
cart.obj	15 à 35	23 à 36	17 à 18
pan_obj.obj	20 à 30	23 à 32	15 à 18
extincteur_obj.obj	7 à 9	19 à 31	11 à 12
M60.obj	6 à 8	14 à 16	9,7 à 11,5
dragon_vrip.ply	6 à 10	12 à 13	-
happy.ply	4 à 8	8 à 10	-
blade.ply	3 à 5	0,5 à 0,6	-
lucy.ply	< 0,5	< 0,5	0,3

FIGURE B.1 – Nombre de FPS obtenus pour chaque modèle (tests réalisés sur des machines avec processeur graphique)

Version de Meshlab Système Carte graphique	V1.3.4 Windows 7 NVIDIA GTX 770	V1.3.3 Windows 7 NVIDIA GTX 760	V1.3.3 Windows XP ATI HD 4850	V1.3.2 Fedora 20 NVIDIA GTX 520	V1.3.3 Windows 8 NVIDIA GTX 330M
bunny.ply	500	-	-	-	-
cart.obj	200	-	-	-	-
pan_obj	150	-	-	-	-
extincteur_obj	100	-	-	-	-
M60.obj	95	-	-	-	-
dragon_vrip.ply	50 à 60	45 à 55	8 à 9	15 à 18	23
happy.ply	32 à 39	35 à 37	6 à 7	13 à 14	18
blade.ply	16 à 18	18	4 à 5	8 à 9	12,5
lucy.ply	0 à 2	0 à 3	Crash	0,5	Crash

FIGURE B.2 – Nombre de FPS obtenus pour chaque modèle (tests réalisés sur des machines avec carte graphique)

C Schéma XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="coordinates">
    <xsd:sequence>
      <xsd:element name="x" type="xsd:float" />
      <xsd:element name="y" type="xsd:float" />
      <xsd:element name="z" type="xsd:float" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="translation">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="vector" type="coordinates" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="scale">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="factor" type="coordinates" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="rotation">
    <xsd:complexType>
```

```
      <xsd:sequence>
        <xsd:element name="angle" type="coordinates" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="object" maxOccurs="unbounded">
    <xsd:complexType>
      <xsd:attribute name="type" type="xsd:string" />
      <xsd:attribute name="src" type="xsd:anyURI" use="required" />
      <xsd:element ref="scale" />
      <xsd:element ref="translation" />
      <xsd:element ref="rotation" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="camera" maxOccurs="1">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="translation" />
        <xsd:element ref="rotation" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="scene" use="required">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="object" />
        <xsd:element ref="camera" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```