Rapport final

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 Introduction

Le projet PFA a pour objectif de permettre aux élèves la gestion d'un projet entièrement, depuis la création du cahier des charges en réponse à une demande de clients jusqu'à l'implémentation du projet visé. Il aura ainsi permis de travailler sur le cahier des charges d'un projet, ce qui est un exercice constructif auquel nous n'avions jamais encore eu à faire, mais également de travailler dans une équipe conséquente de 7 personnes quand la plupart des projets réalisés dans le cadre de notre cursus se font par équipe de 3 à 4 personnes.

Le projet PFA effectué est un projet d'imagerie numérique. Son but est de travailler à l'aide d'un logiciel dans une scène virtuelle en trois dimensions, et d'obtenir grâce à cette dernière des images en deux dimensions permettant de recréer une illusion de trois dimensions. Les algorithmes qui auront notamment été étudiés au cours de ce projet permettent l'obtention d'anaglyphes, d'autostéréogrammes, ou encore de folioscopes. L'imagerie numérique n'est pas étudiée au cours des deux premières années d'Informatique à l'Enseirb-Matmeca. Ainsi, ce projet constituait une réelle ouverture d'esprit sur un nouveau domaine de l'informatique.

Dans ce rapport, nous présenterons dans un premier temps le domaine de la synthèse d'image et les rendus à implémenter dans notre logiciel, puis nous résumerons les besoins visés lors de la phase de cahier des charges avant de présenter l'implémentation du projet. Présentation intérêt du projet, pourquoi on voulait ce projet, ce qu'on cherchait à améliorer grâce à ce projet (nous + en général)

2 Présentation du projet

Cette partie permettra de présenter le domaine de la synthèse d'image ainsi que le sujet du projet sur lequel nous avons eu à travailler. Nous présenterons également les trois rendus qui ont été implémentés pour le logiciel : l'anaglyphe, l'autostéréogramme et le folioscope.

2.1 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 **??**.

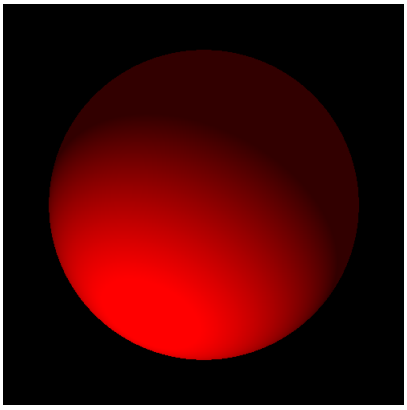

FIGURE 2.1 – Application d'une lumière diffuse à une sphère rouge [1]

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.

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. définition caméra, objet, scène domaine et petit historique

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

2.2 Sujet du projet

présentation du sujet reprendre présentation du cahier des charges améliorer avec une vision plus poussée du projet

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.

reprise sujet version cahier des charges en amélioré

2.3 Présentation des rendus souhaités

Nous présenterons dans cette partie les trois rendus que nos clients nous ont demandé d'implémenter dans notre logiciel : les autostéréogrammes, les folioscopes ou flipbooks, et les anaglyphes.

2.3.1 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 **??**). En combinant par exemple avec le logiciel Gimp (outil d'édition et de retouche d'image) l'animation peut être obtenue en GIF.

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

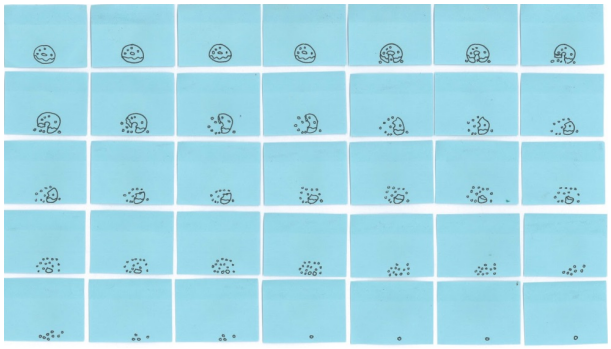

FIGURE 2.2 – Story-board d'un flipbook [2]

2.3.2 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 **??**.

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.

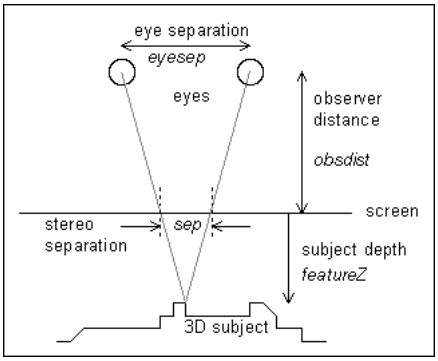

FIGURE 2.3 – Visualisation d'un autostéréogramme en vision parallèle [3]

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 **??** a permis d'obtenir l'autostéréogramme de la figure **??**.

3. http://www.techmind.org/stereo/geometry.gif
4. http://en.wikipedia.org/wiki/Autostereogram

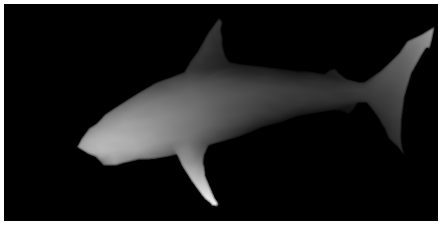

FIGURE 2.4 – Carte des profondeurs [4]

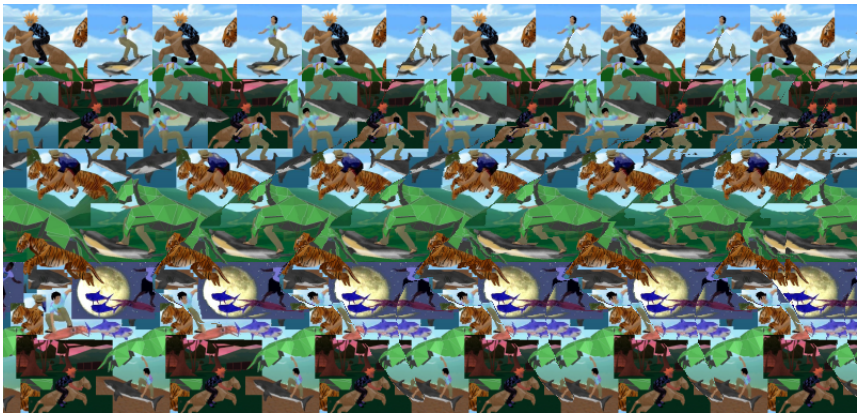

FIGURE 2.5 – Autostéréogramme obtenu [5]

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).

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.

2.3.3 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 **??**). 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.

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

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.

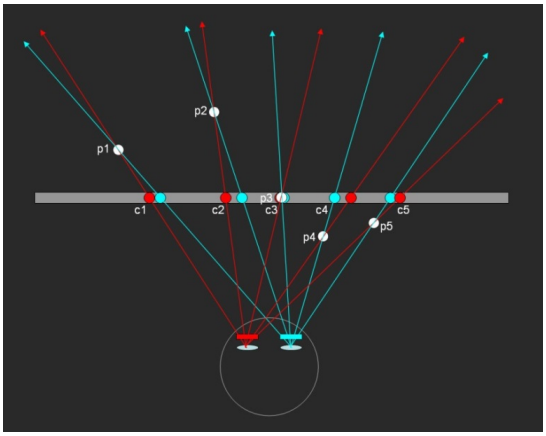

FIGURE 2.6 – Le décalage de la partie rouge et cyan permet à l'œil de percevoir l'image non plus dans un plan XY mais dans l'espace [7]

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 **??**). Dès lors, une matrice de transformation 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.

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

FIGURE 2.7 – Rendu final d'une image par l'algorithme de Dubois [8]

présentation anaglyphes,autostéréogrammes, folioscopes

3 Cahier des charges

Nous reviendrons dans cette partie sur la création du cahier des charges et les recherches qui ont été effectuées pour les algorithmes des rendus présentés dans la partie ci-dessus.

3.1 Résumé du cahier des charges

retour sur la rédaction du cahier des charges aide du client (cours du début pour ce qu'il attendait comme contenu) suivi hebdomadaire pour contrôler l'avancée et nous orienter pour un cahier des charges comme il l'attendait méthode de travail, recherches, prototypes

Dans le cadre du projet PFA, la première partie consistait à rédiger un cahier des charges fonctionnel pour définir l'ensemble des besoins du futur logiciel.

Lors de notre premier rendez-vous, notre client a d'ores et déjà défini le format qu'il souhaitait pour le cahier des charges, et le contenu qui était essentiel pour lui. Il fallait ainsi présenter le domaine d'études et les connaissances actuelles sur des algorithmes de création d'anaglyphes, d'autostéréogrammes ou de folioscopes. Ensuite, le sujet était à redéfinir précisément, puis les besoins fonctionnels et non fonctionnels demandés par les clients pour ce logiciel, ainsi que les contraintes engendrées par ceux-ci. Enfin, il fallait présenter des prototypes permettant de répondre aux différents besoins ennoncés, quelques interfaces graphiques pour simuler l'utilisation du logiciel, et surtout réfléchir à l'architecture future du projet.

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

La rédaction des parties Domaine et Etat de l'existant aura demandé de nombreuses recherches, notamment pour trouver des articles présentant des algorithmes de création d'anaglyphes et d'autostéréogrammes. L'étude du domaine aura permis de se familiariser avec le vocabulaire de la synthèse d'image et avec les différents rendus souhaités par nos clients. Des notions telles que la scène, la caméra, ainsi que la compréhension du fonctionnement des anaglyphes et des autostéréogrammes, auront ainsi permis une meilleure immersion dans notre projet. Les algorithmes relatifs aux anaglyphes concernent principalement le traitement des couleurs pour qu'aucun artefact n'apparaisse au moment de la visualisation finale. Pour les autostéréogrammes, le traitement d'une carte des profondeurs permet d'obtenir une image qui, lorsque l'on sait l'observer, fait apparaître un relief. Enfin, il n'existe pas d'algorithme particulier pour la génération de folioscopes. Seule une série de prises de vue d'une scène avec des angles d'observation proches peuvent permettre, si elles sont visualisées les unes à la suite des autres et suffisamment rapidement, de pouvoir imaginer un mouvement, et ainsi un relief.

Les besoins fonctionnels et non fonctionnels d'un projet doivent impérativement être ciblés durant la phase de cahier des charges, car c'est grâce à eux que le contrat passé entre les clients et l'équipe de programmeurs pourra être exhaustif et protéger les deux parties contre d'éventuelles envies de modification au cours de la phase de réalisation. Pour un tel logiciel, les besoins fonctionnels s'apparentent bien souvent à de futures fonctionnalités du logiciel, tel que la création d'une scène, sa manipulation, et les prises de vue pour obtenir les rendus demandés dans le sujet. Toutefois, les besoins non fonctionnels, même s'ils peuvent être clairement énoncés par le client, sont bien souvent implicites et doivent être déterminés en fonction du discours tenu. Pour un logiciel de synthèse d'images, la fluidité d'affichage est bien souvent un besoin essentiel, car l'utilisateur s'attend à une certaine rapidité du logiciel lors de la manipulation de la scène et de la génération de rendus. Mais d'autres besoins, à savoir la portabilité du logiciel et sa maintenabilité, ont également été exprimés par les clients et ont été considérés comme essentiels pour ce projet. En effet, l'utilisaton de ce logiciel, au moins sous les systèmes d'exploitation Linux et Windows, était importante pour pouvoir travailler sur différentes machines dans leur travail. De plus, un tel logiciel pouvait être amené à être amélioré ou réutilisé en partie dans de futurs projets, et c'est pourquoi il devait être facilement maintenable. Ces besoins ont apportés des contraintes quand à la réalisation du logiciel, par exemples sur les langages et les bibliothèques utilisées. Les choix ainsi effectués pour permettre de tenir ces contraintes seront détaillées par la suite.

//passer vite fait sur les proto et l'archi besoins fonctionnels, non fonctionnels, contraintes archi prototypes

3.2 Anaglyphes

algo cherchés, trouvés, implémentés raisons des choix, difficultés rencontrées (dire pourquoi on a utilisé Dubois mais pas celui du cahier des charges

3.3 Autostéréogrammes

algo cherchés, trouvés, implémentés raisons des choix, difficultés rencontrées

3.4 Choix de programmation

Pour la réalisation du projet, nos clients nous proposaient des langages comme C, C++ ou encore Python. C++ nous est apparu comme le choix le plus judicieux pour notre logiciel, afin de faciliter l'utilisation de bibliothèques telles que OpenGL et Qt et de pouvoir bénéficier de l'aide d'une grande communauté internet en cas de difficultés. //pourquoi c++ 11 : voir avec xavier

Comme nous l'avons exprimé plus haut, l'un des besoins non fonctionnels primordiaux de notre projet est la maintenabilité du code. Pour celà, il nous fallait choisir des outils et des bibliothèques qui étaient destinées à perdurer le plus longtemps possible. L'utilisation de Qt et OpenGL pour Qt est assez répandue pour des logiciels tel que le nôtre de visualisation et de manipulation en trois dimensions, et également très complètes car Qt propose un vaste choix de modules qui permettent de traiter un ensemble très variable de tâches (notamment le parsage de fichier XML que nous

avons utilisé). Comme ces bibliothèques sont de plus portables, elles convenaient parfaitement à l'implémentation que nous souhaitions réaliser. La version 5 de Qt est la plus récente actuellement, mais malgré sa jeunesse elle est devenu au fil des modifications suffisament stable pour pouvoir être utilisée sans problème, et elle dispose d'une communauté Internet active prête à aider en cas de difficultés de code. De plus, cette bibliothèque utilise la version ES 2.0 de OpenGL, qui est une version non seulement qui utilise des shaders, mais également qui peut être utilisée pour de la programmation d'applications mobile par exemple. C'est donc pour cette variété, cette communauté, cette maintenabilité et cette extensibilité que nous avons choisi d'utiliser ces bibliothèques et ces versions. C++11 Qt5 OpenGL ES 2

4 Déroulement de la réalisation

4.1 Diagramme de Gantt prévisionnel

Une fois le cahier des charges mis en place et l'accord des clients donnés, la première étape du projet consistait à réfléchir au diagramme de Gantt prévisionnel pour les trois mois destinés à la réalisation du projet. La priorité était donnée aux algorithmes de réalisation des différents rendus espérés grâce au logiciel, et plus particulièrement aux anaglyphes et aux autostéréogrammes. Il fallait revenir sur les algorithmes choisis et présentés dans le cahier des charges, approfondir les recherches pour être sûr de ne passer à côté d'aucun autre algorithme plus performant, et s'approprier le fonctionnement des algorithmes pour les comprendre puis les retranscrire. En parallèle, d'autres personnes pouvaient travailler sur une partie plus proche du logiciel finale, à savoir la manipulation de la scène ou encore les parseurs de fichiers pour le chargement des objets. Une fois les manipulations premières de la scène effectuées, les travaux prioritaires seraient le chargement et la sauvegarde de la scène grâce à des fichiers d'extension XML, la création de prises de vue simples ou de folioscopes à partir de la scène et leur sauvegarde, et enfin l'assemblage complet du logiciel avec une interface d'utilisation permettant d'utiliser les différents modules.

Un récapitulatif du diagramme de Gantt prévisionnel du projet est donné en annexe.

Pour déterminer les durées nécessaires à chaque partie de ce projet, nous avons dû réfléchir aux éventuelles recherches à réaliser au préalable, ainsi qu'à la complexité des tâches en les découpant en sous-parties. Nous avons également réfléchi aux éventuels examens qui auraient pu nous ralentir dans la réalisation en fonction des périodes.

Ainsi, pour une tâche telle que l'implémentation d'algorithme d'anaglyphes, il fallait dans un premier temps revenir et compléter les recherches faites sur l'état de l'existant. En effet, comme nous l'avons vu dans la partie présentation du projet, de nombreuses études ont été effectuées pour mettre en place des algorithmes plus ou moins performants capables de générer des anaglyphes. Une fois le choix des algorithmes effectués, l'implémentation de celui-ci en C++ pourrait être effectuée. Au final, la période déterminée pour la recherche d'algorithmes et l'implémentation de ces algorithmes était d'environ un mois pour une équipe de deux personnes.

Un autre exemple de période déterminée est celle destinée à la sauvegarde et au chargement des scènes. Nous avions d'ores et déjà un fichier XML modèle sur lequel s'appuyer, ce qui a permis de passer rapidement à l'implémentation. La sauvegarde consistant simplement en une écriture dans un fichier, seul le chargement demandait des recherches pour déterminer la meilleure bibliothèque pour effectuer le découpage d'un fichier XML. La période déterminée pour cette tâche était de deux semaines pour une équipe de deux personnes. étapes du gantt indication choix des durées

4.2 Réalisation du projet

La partie Réalisation de ce projet se sera étalée sur les mois de Décembre, Janvier, Février et Mars. Grâce à la réflexion effectuée durant la phase de rédaction du cahier des charges, nous avons pu aborder notre projet avec une idée claire des priorités.

La partie réalisation des algorithmes était très importante pour nos clients, c'est pourquoi nous avons dès le début de la réalisation mis en place deux équipes de deux personnes pour les deux algorithmes principaux : les anaglyphes et les autostéréogrammes. Les folioscopes quand à eux ne demandaient pas d'algorithme particulier de réalisation, si ce n'est la manipulation de la caméra vis à vis de la scène qui allait être mise en place grâce à la troisième équipe.

En effet, notre troisième équipe, constituée des trois derniers membres, a pu continuer de travailler sur le prototype généré lors de la phase du cahier des charges. Il a ainsi été amélioré pour mettre en place les parseurs de fichiers qui allaient être nécessaires pour la génération des objets de la scène, et améliorer la manipulation de la scène.

Malheureusement, l'implémenation des algorithmes s'est révélée plus longue que nous l'avions initialement prévue. En effet, cette partie étant l'une des plus importantes pour nos clients, nous avons eu besoin d'aller plus loin que ce que nous avions envisagé dans un premier temps. Il a également fallu que nous fassions davantage de recherches pour compléter celles qui avaient été effectuées pour la rédaction du cahier des charges. De plus amples explications seront données dans la partie Difficultés rencontrées de ce rapport. Au final, l'implémentation des algorithmes Anaglyphe et Autostéréogramme aura duré environ deux mois et demi au lieu du mois initialement prévu, avant de pouvoir les intégrer au reste du projet.

En parallèle, la troisième équipe, appuyée parfois par l'un ou l'autre des autres membres en cas de besoin, a continué d'avancer sur la partie Scène du logiciel. L'implémentation de certaines manipulations de scène (déplacements de la caméra ou des objets notamment) auront parfois étaient plus rapides que prévu, mêmes si quelques retours en arrière ont été nécessaires au milieu du projet comme nous l'expliqueront dans la partie Difficultés rencontrées. L'ordre chronologique initialement prévu dans le diagramme de Gantt n'aura finalement pas toujours été respecté. Par exemple, la différence de travail entre l'ajout d'un objet dans la scène et l'ajout de plusieurs objets n'étant pas énorme, ces deux parties auront été effectuées simultanément. Il en est de même pour la sauvegarde automatique qui aura été ajoutée en même temps que la sauvegarde classique de la scène, ou encore pour les manipulations de la scène et de l'objet dont l'implémentation était grandement facilitée par l'utilisation de la bibliothèque OpenGL pour Qt.

Au final, la totalité des tâcjes prévues jusqu'au début du mois de Mars auront pu être effectuées dans les temps, nous laissant le mois de Mars pour l'intégration des algorithmes au logiciel, l'amélioration de certaines fonctionnalités, le développement de l'interface et la rédaction de ce rapport. progression chrono (un peu) sur la réalisation

4.3 Difficultés rencontrées

lien avec les contraintes (utilisation d'outils pas optimums pour qtcreator pour rester dans la contrainte de portabilité plus de temps que prévu sur les algos difficultés à démarrer besoin de créer une boite noire pour séparer ça du projet temporairement visualisation de quoi utiliser quand et pour quoi gant remodelé

Au cours de ce projet, certaines difficultés rencontrées auront parfois ralenti l'avancement initialement prévu.

4.3.1 Implémentation des algorithmes

L'implémentation des algorithmes d'anaglyphes et d'autostéréogrammes était le coeur de notre projet et du logiciel que nous avions à mettre en place. Il était donc important que nous lui accordions un temps conséquent durant notre projet, et c'est pour cela que nous avions initialement prévu de faire travailler deux personnes sur ces algorithmes durant un mois, en partant des recherches qui avaient été effectuées pour le cahier des charges. Toutefois, l'implémentation de deux algorithmes étant nécessaire pour chacun de ces rendus, ce mois a finalement été trop court pour nos équipes de programmeurs.

Pour les autostéréogrammes, le rendu donné par le premier algorithme choisi était découpé en tranches et ne donnait pas suffisament l'impression de profondeur comme il aurait dû [figure]. Cet effet avait en effet été souligné dans l'article de [NOMS DES AUTEURS], mais l'utilisation de points aléatoires pour le fond de l'image accentuait cet effet de tranches. Il a alors fallu mettre en place le second algorithme que nous avons présenté dans la partie Cahier des charges de ce rapport, ce qui a allongé la période de réalisation. De plus, l'optimisation proposé pour cet algorithme par [NOMS DES AUTEURS] posait dans un premier temps problème pour la génération de l'autostéréogramme. Il aura alors fallu procéder étape par étape pour obtenir un premier rendu sans l'optimisation de l'algorithme, qui aura été ajoutée par la suite.

Pour les anaglyphes, la première difficulté était pour la récupération de paires d'images sur lesquelles travailler. Notre équipe a tout d'abord essayé de travailler sur la scène avec OpenGL pour récupérer des images qu'ils pourraient utiliser. Toutefois, une autre difficulté, liée aux shaders comme nous le présenterons dans la partie qui suit, les a empêché de poursuivre dans cette voie. Ils ont donc choisi de travailler sur des images trouvées sur Internet afin d'avancer dans l'implémentation. Une autre difficulté de l'algorithme des anaglyphes est le travail sur la [saturation gamma ?]. Pour permettre un rendu agréable à visualiser, notre équipe a cherché à travailler sur cette saturation, mais l'effet obtenu n'était pas toujours celui qu'ils espéraient. [FINALEMENT ?] Enfin, nos clients souhaitant pouvoir imprimer les anaglyphes, il était très important que les couleurs lors de l'impression révèlent aussi bien l'anaglyphe. En effet, si un écran d'ordinateur possède les composantes rouge, verte et bleue pour son affichage, la plupart des imprimantes utilisent le pattern cyan, magenta, jaune et noir. Pour éviter d'éventuels artefacts qui auraient pu être causés par la transformation automatique des couleurs, il a fallu traiter celles-ci pour que l'impression soit le plus parfaite possible.

4.3.2 L'utilisation de shaders

On appelle shaders des programmes informatiques qui vont travailler directement sur la carte graphique ou le processeur graphique d'un ordinateur pour permettre un rendu meilleur et plus rapide. Ces shaders sont souvent utilisés en imagerie numérique, notamment par le logiciel Blender qui aura été un des logiciels-modèle pour la création du nôtre. L'utilisation de ces shaders nous paraissait importante pour permettre un rendu agréable dans notre logiciel, et pour atteindre nos objectifs de fluidité énoncés dans notre Cahier des charges.

Dans certains ordinateurs, et surtout dans la plupart des ordinateurs portables, il n'existe pas de carte graphique à proprement parlé. Un processeur est entièrement utilisé pour permettre un certain niveau de compétences graphiques, bien que celles-ci soient largement inférieures à celles de cartes graphiques de même récence. Toutefois, les processeurs graphiques empêchent parfois l'utilisation de shaders, car des processeurs trop anciens ne reconnaîtront pas les versions de shaders les plus récentes.

Dans une première version de notre logiciel, nous avions souhaité utiliser une version assez récente de shaders. Toutefois, celle-ci posait problèmes car elles n'étaient pas reconnue sous des machines Linux. La portabilité du logiciel étant primordiale, du moins sous les environnements Windows et Linux, nous avons dû effectuer une modification de l'ensemble du code déjà construit afin de passer à une version plus ancienne des shaders et permettre la portabilité du logiciel.

4.3.3 Les outils utilisés pour la réalisation du projet

Pour permettre la portabilité et la maintenabilité du projet, nous avons choisi de l'implémenter en C++11, en utilisant Qt5 et la bibliothèque OpenGL pour Qt, et en compilant avec l'outil CMake.

Malgré sa récence, la communauté Internet sur la bibliothèque Qt et son module OpenGL est très importante et de nombreux problèmes rencontrés ont pu être résolus facilement grâce à des recherches sur certains forums. A l'inverse, l'utilisation de CMake pour la bibliothèque Qt est très rare. En effet, l'outil le plus souvent dans la réalisation de projet Qt est QMake. [VOIR POURQUOI ON L'A PAS UTILISE, PARCE QUE VISIBLEMENT ON AURAIT PU] CMake étant très

peu utilisé pour la bibliothèque Qt, il est parfois difficile de trouver sur Internet de l'aide pour inclure des modules, par exemple le module QtXml. La plupart de la documentation relative à ce problème est très simplement géré dans QMake, mais il aura fallu beaucoup de recherches pour parvenir à trouver les bons noms de bibliothèques et de modules pour l'inclure avec CMake.

4.4 Bilan final

Comme nous l'avons expliqué dans la partie Réalisation, le diagramme de Gantt a dû être remanié au fil de l'avancement du projet car certaines tâches, plus faciles que prévues, ont été anticipées alors que d'autres, posant plus de problèmes, ont été allongées. Le diagramme de Gantt a été donné en Annexe.

La principale modification consiste ainsi en l'allongement de la période destinée à la recherche d'algorithmes pour les rendus et à leur implémentation. Les raisons de cette différence ont été expliquées dans la partie Difficultés rencontrées ci-dessus. Les manipulations des objets et de la scène ont été effectuées plus rapidement que prévu, mais une période de retour en arrière à cause des shaders a été ajoutée, allongeant la période de travail sur le chargement et la sauvegarde de la scène qui a été momentanément interrompue le temps de régler ce problème.

Malgré ces débordements dans le temps, la réalisation du projet se sera achevée dans les temps, le dernier mois ayant permis d'intégrer les algorithmes au logiciel, de compléter et de parfaire des fonctionnalités. Au final, l'ensemble des manipulations de la scène et de ses objets qui avaient été promises dans le cahier des charges ont été implémentées. De même, l'ensemble des rendus prévu est générable à partir de la scène, et deux algorithmes de génération sont proposés pour les autostéréogrammes et les anaglyphes alors qu'un seul avait été cité pour l'autostéréogramme dans le cahier des charges, et que seul le traitement de couleurs pour l'impression avait été initialement prévu pour m'anaglyphe (et pas la saturation).

Au niveau des besoins non fonctionnels, la portabilité aura été respectée malgré l'utilisation des shaders qui auraient pu poser problème sur certaines machines. Grâce aux nombreux modules proposés par la bibliothèque Qt, portable et très complète, des solutions auront été trouvées pour l'ensemble des modules et des cas d'utilisation sans avoir à sacrifier de fonctionnalité pour permettre l'utilisation du logiciel aussi bien sous Windows que sous Linux.

[FLUIDITE]

Enfin, la maintenabilité du logiciel sera également possible grâce au choix des outils et à l'architecture du projet. Tout d'abord, l'utilisation de Qt5, qui est actuellement la plus récente de Qt, laisse envisager que le logiciel pourra être utilisé longtemps sans avoir à migrer vers une nouvelle version de la bibliothèque. Ensuite, l'utilisation de OpenGL ES 2.0, qui est la version d'OpenGL de base avec Qt5, pourrait éventuellement permettre de générer une application mobile du logiciel. L'architecture a également été pensée pour permettre cette maintenabilité. En effet, comme le prouvent les deux algorithmes

Ce bilan positif montre que l'ensemble des besoins fonctionnels et non fonctionnels ciblés dans le cahier des charges ont été implémentés avec succès. Nous espérons que ce projet sera réutilisé et maintenu, puisqu'il a été conçu dans cet optique. Il sera éventuellement possible de générer une application mobile à partir du code existant, ou d'ajouter et de tester d'autres algorithmes ou d'autre rendus possible à partir d'une scène en trois dimensions. Nous espérons également que nos clients auront été satisfaits du travail réalisé et du logiciel final. gantt final expliquer ce qui a changé pourquoi réalisation par rapport au cahier des charges

5 Conclusion

apports du projet connaissances acquises compétences acquises pour le cahier des charges compétences acquises pour la gestion de projet avantage d'un client qui nous a suivi tout du long retour sur l'utilisation d'une forme de méthode agile (plus ou moins)

Apports du projet Bilan Idée d'évolution du projet