Rapport de projet Infographie
Ace Attorney 3D

Auteurs :
Vincent Metton,
Grégoire Pommier

Responsable :
M. Yannick Guesnet

2 mai 2016

Table des matières

Table des figures

Remerciements

Nous remercions :
M.GUESNET de nous avoir soutenu et d'avoir répondu à nos différentes questions sur l'OpenGL lors de ce projet.

Introduction

En lisant le titre de ce rapport, vous avez pu vous demander : "Mais qu'est-ce que Ace Attorney ?" .
Ace Attorney est une série de jeux vidéos développés par Capcom et édités par Nintendo depuis 2001 sur GameBoy Advance puis portée sur DS et 3DS.
Ce type de jeu est le visual novel, c'est-à-dire que l'aventure du jeu se défile sous nos yeux, en superposant du texte, qui sera alors le script du jeu, sur une image de fond. Le joueur doit, à certains moments, agir afin de faire avancer l'action du jeu.
Dans ces jeux, nous incarnons un avocat à la défense, qui doit prouver l'innocence de ses clients en découvrant la vérité sur l'affaire en cours (généralement un meurtre).
Ces jeux se déroulent en deux phases : une d'enquête, où l'on cherche des preuves afin de disculper notre client, et une phase de procès où nous confrontons les preuves accumulées aux différents témoignages pour démêler le vrai du faux et ainsi résoudre l'affaire.

Dans ce projet, nous n'avons implémenté que la première affaire du premier jeu (The First Turnabout) qui ne contient qu'une phase de procès.

Nous décrirons dans un premier temps les raisons qui nous ont poussées à faire ce projet, puis nous détaillerons les structures et algorithmes utilisés pour la partie graphique (OpenGL), ensuite nous expliciterons les structures et algorithmes de la partie script. Nous finirons par exprimer les différents problèmes rencontrés, les idées d'améliorations et nous conclurons.

Figure 1 – Un aperçu du jeu original

1 Raisons du projet

Dans cette partie nous allons détailler les raisons qui nous ont amenées à choisir ce sujet.

1.1 Une passion commune

Tout d'abord, nous sommes tous les deux des passionés de jeux vidéos et tout particulièrement de la série Ace Attorney. De ce fait, cela nous a paru comme une évidence de choisir cet univers et de vouloir le représenter en 3D. En effet, les jeux originaux sont en 2D et offre une variété graphique assez faible au niveau des détails sur la profondeur et des mouvements de caméra. Pouvoir recréer entièrement cet univers en utilisant la biliothèque OpenGL nous a donc offert une totale liberté de mouvement créant des angles de vue totalements inédits.

2 Les Fonctionalités Graphiques

2.1 Les Fonctions Utisant OpenGL

2.1.1 Le sol

Le sol et constitué de 6 pavés mis côte à côte, afin d'avoir une plus belle réflexion de la lumière. Ces pavés sont eux même composés de sous pavés, qui ne sont au final que de simples GL_QUADS, sur lesquels on plaque une texture de pavage à neuf cases.

2.1.2 Les meubles

Les meubles sont créer grâce à la fonction creer_pave, qui prends en paramêtre les coordonées du centre du pavé, ainsi que sa largeur, sa profondeur et sa hauteur. Cette fonction crée correctement toutes les faces vers l'exterieur, et fixe leurs normales, afin que la lumière et les textures ajoutées s'affichent correctement.

2.1.3 Les murs

Les murs sont crées par la fonction creer_mur, qui ressemble beaucoup à creer_pave, mais ne crée pas les faces en z et -z, et qui de plus, crée les faces vers l'interieur du solide.

2.1.4 Le blason, et la balance

Les éléments situés au dessus du juge ont été crées à l'aide des fonctions de la glu, gluCylinder, et gluDisk. Elles sont tournées et translatées sur le fond de la pièce, et des texture réalistes sont plaquées dessus.

2.1.5 Le marteau et son support

Le marteau, tout comme les précédents éléments, est crée à l'aide d'un quadric, avec les fonctions de la glu. Il a aussi été ajouté un glRotatef dedans, afin de pouvoir faire varier la position du marteau. Cela simule un coup au rendu convainquant.

2.1.6 Le Stand des Témoins

Ce stand est constitué en bas d'un ensemble de pavés, créé par la fonction décrite si dessus, repartis de façon homogène sur un demi-cercle. Il est

surmonté par des gluPartialDisk empilés les uns sur les autres.

2.1.7 Les personnages

Pour créer les personnages, une nouvelle fonction a due être definie, c'est la fonction creer_pave_with_texture, cette fonction permet d'associer à chaques faces d'un pavé, une texture différente, fournie en argument de la fonction. De plus, certains personnages ont subit quelques modifications, permettant des animations, au grés du scénario. En effet, les bras du personnage principal sont affublés de deux glRotatef, ce qui leurs permet de bouger dans deux directions différentes, le bras du juge, à l'aide d'un glTranslatef, bouge au rythme de son marteau, et les têtes des spectateurs de la foule peuvent, à l aide d'un glRotatef, tourner de gauche à droite.

2.1.8 Les arcades

Des arcades ont été appliquées sur les murs afin d'améliorer leur esthétisme. Deux pavés créés normalement, surmontés de deux autres, qui sont tournés à 45 degrés et déplacés au dessus des premiers.

2.2 Le modèle de vu

Afin d'afficher la scène, un gluPerspective est fait à l'initialisation. Les arguments associés sont un fovy de 60, ce qui nous a semblé rendre une scène correcte, non déformée. Le rapport d'aspect reste à 1, near est à 10 et far est à 3000 ce qui permet, quelque soit notre emplacement dans la scène, de pouvoir l'afficher en entier.
Un gluLookAt est fait à chaque fois que la scène est affichée, il prend en paramètre des variables globales ce qui permet de pouvoir facilement déplacer la camera dans la scène. Il y deux façons de se déplacer dans la scène.

2.2.1 Le mode "Vue libre"

Grâce à la fonction PollEvent de le SDL, on peut récupérer les inputs fait au clavier quand la fenêtre à le focus, cela permet de pouvoir modifier les globales whereiamx,y,z et whereialookx,y,z offrant une totale liberté de mouvement dans la scène.

2.2.2 Le mode automatique

Certaines touches, ou certaines balises dans le script, lancent les fonctions de déplacement, qui elles-mêmes appellent movecamera, qui permet de

deplacer de manière fluide la camera dans la scène, en faisant un display de la scène tous les 1/100 ème de la distance entre où je suis et l'endroit où je dois être.

2.3 Les animations

Plusieurs fonctions permettent de créer de petites animations, et sont lancées par des évements clavier créés manuellement ou par le biais du script. Ces fonctions modifient la valeur des variables globales mentionnées plus haut, et affichent la scène plusieurs fois afin de pouvoir voir l'animation complète.

2.4 La musique

Grace à SDLmixer, de la musique à été ajoutée au projet, reprenant les sonorités classiques du jeu. Ces musiques sont jouées lors d'évènements de clavier, ou bien lors d'animations (comme par exemple celui du marteau qui est associé à son mouvement). Il y a deux types de musiques.

2.4.1 Les musiques standarts

Ce sont celles couramment jouées en fond sonore, pendant la lecture du script. Une seule peut être jouée à la fois, et certaines bouclent alors que d'autres ne sont jouées qu'une fois.

2.4.2 Les "morceaux" de musique

Ce sont les bruitages qui ne couperont pas la musique de fond quand ils seront joués. Ce sont principalement les objections des avocats, et autres interjections.

3 Les fonctionnalités du script

Le but de notre projet était de revisiter l'univers d'Ace Attorney et ce de deux manières différentes : d'une part grâce à la réalisation graphique, que nous verrons plus tard dans ce rapport, et d'autre part par le suivi du script.

En effet, comme dit en introduction, le script a une place majeure dans le déroulement du jeu car c'est lui qui fait avancer ou non l'intrigue.
Pour des soucis de temps, nous avons choisi de ne ré-implémenter que la première affaire du premier jeu.

Cette partie du rapport se découpe de la manière suivante : dans un premier temps, nous verrons comment les différents fichiers de script s'organisent entre eux.Dans un second temps, nous décrirons les structures utilisées pour la bonne gestion des fichiers de textes selon la situation. Enfin, nous verrons comment l'implémentation de cette partie en elle même.

3.1 Les fichiers

Tout d'abord, les fichiers de textes sont tous placés dans le dossier ./text_files/ afin de bien séparer les morceaux de l'application finale. Ces fichiers s'articulent donc en plusieurs catégories qui sont les suivantes :

- evidences.txt -> ce fichier sert à lister toutes les preuves ainsi qu'une petite description de celles-ci
- script.txt -> le script général. C'est le fichier parent à tous les autres.
- qcms/ -> les différents qcms du script en cours
- cross_exams/ -> les fichiers représentant la contre-argumentation en cours
- testimonies/ -> les différents témoignages du script en cours.

Dans le fichier script.txt, et ceux qui en découlent, nous trouveront un certains nombre de balises, délimitées par "[" et "]", qui indiquent une certaine action à effectuer en fonction du contenu de la balise (que ce soit le lancement d'une musique ou d'une autre partie du script), sauf quand un personnage parle : cette information est indiquée par une "*" avant le nom du personnage, et les lignes de dialogues qui suivent sont affichées sur la console.

3.2 Les structures

Dans cette partie, nous allons décrire les différentes structures associées à la gestion du script et à sa bonne exécution.

```
struct key_value {
char *key;
char *value;
};
```

Cette structure est la structure classique pour contenir un couple de clé-valeur.

```
struct qcm_struct {
char *talking;
char *question;
char *answer[5];
int nb_proposition;
struct key_value proposition_case[5];
char *case\_files[5];
};
```

La structure d'un QCM renseigne toutes les informations utiles à la progression de celui-ci, c'est à dire le personnage posant la question, la question en elle même, les réponses possibles, le nombre de propositions, un couple de clé/valeur associant comme clé "CASE_i" à la valeur de texte qui sera affiché dans le menu du QCM. Enfin, la structure enregistre les différents fichiers qui seront lus quand une réponse est proposée.

```
struct evidence {
char *evidence_name[30];
char *evidence_type[30];
char *evidence_other[30];
char *evidence_description[30];
int evidence_active[30];
int nb_evidence;
};
```

Cette structure permet l'enregistrement des informations sur les preuves contenues dans le fichier "evidences.txt", avec notamment son nom, son type,

sa description, si elle est active ou non et ses informations complémentaires. La structure renseigne également sur le nombre de preuves totales du fichier.

```
struct cross_exam_struct {
char *answer_line[15];
int evidence_to_show;
char *lines[20];
char *extras[20];
char *name;
int current_line;
int nb_lines;
int nb_extras;
};
```

Dernière structure de données permettant l'emcapsulation des informations pour le bon déroulement d'une contre-argumentation. Elle possède toutes les lignes du témoignage, les lignes correctes sur lesquelles on peut présenter une preuve et si le numéro de la preuve (evidence_to_show) est la bonne fini la contre-argumentation. Après avoir passé toutes les lignes du témoignage, on affiche les lignes "extras", c'est à dire, un petit dialogue qui se passe après mais qui en conclue pas cette phase de jeu.

3.3 L'implémentation de la lecture du script

En premier lieu, pour éviter que la lecture du script n'empiète sur l'affichage de la scène, et donc pouvoir utilisé les différentes touches claviers pour effectuer les actions que nous avons définies, nous avons dû l'encapsuler dans un thread POSIX afin de vraiment dissocier le thread principal gérant l'affichage de la scène et le thread utilisateur permettant la lecture des différents fichiers.

Dans un second temps, dès qu'une ligne indiquant le changement de fichier (par les balises [QCM] ou [CROSS_EXAM]), un nouveau fichier est lu, mettant en suspend la lecture en cours. Des fonctions sont prévues pour ces deux cas là, car ils sont différents d'une simple lecture puisqu'ils répondent à un certain gameplay.

Dans l'implémentation proposée, on vérifie également la ligne du script lue pour savoir le comportement a adopté. En effet, si la ligne commence par une étoile, on indique qu'il faut changer le point de vue de la caméra

pour pointer sur le personnage qui parle.
Pour des balises de type [START_MUSIC_TRIAL] ou [JUDGE_HAMMER] cela lance la musique ou bien la petite animation associée à ce type d'évènement.

Pour parvenir à indiquer à la partie graphique qu'un changement doit être opéré, et ce parce que les deux parties de l'application sont sur deux threads différents, nous avons dû nous servir de la fonction SDL_PushEvent(SDL_Event *event) permettant d'empiler sur la pile des évènements de la SDL pour que la partie graphique puisse se mettre à jour en fonction des données du script. Nous avons pu utiliser cette fonction puisque la documentation de la SDL indique que celle-ci est "thread-safe" et que les changements opérés sur un thread seront visibles sur un autre.

4 Autres

Dans cette partie, nous allons présenter les problèmes que nous avons rencontrés au cours de ce projet, les améliorations possibles pour pouvoir le continuer et enfin les compétences que nous avons acquises lors de la réalisation des différentes parties du projet.

4.1 Problèmes rencontrés

Dans ce projet nous avons rencontré différentes difficultés. Tout d'abord, il a fallu s'initier à l'utilisation d'OpenGL. Certains aspects furent compliqués à réaliser, notamenent réussir une projection au bon endroit avec volume de vue adéquat. Mais aussi créer des solides avec des faces dans le bon sens, des textures correctement appliquées. Il nous a fallu aussi gérer l'éclairement, la décompositon de certains éléments en pavage pour un rendu plus lisse. La lumière diffuse a aussi été dûre à gérer, longtemps les faces opposées à la lumière sont restées noires. Il a aussi fallu gérer un grand nombre de textures, leur application dans le bon sens sur chacunes des figures. De plus, il a fallu comprendre l'utilisation des quadriques et l'appliquation des textures sur eux. Le witness stand a requis l'usage de formules de trigonométrie afin d'obtenir une disposition homogène des piliers. Du coté de la lecture du script, les complications ont été d'encapsuler les accès mémoire dans un thread, pour ne pas ralentir la fenêtre graphique. Enfin, la modularisation du programme en fichiers disjoints a été longue à faire à cause des étroits liens entre les globales et leurs initialisations, ainsi que leur usage dans des fonctions.

4.2 Idées d'améliorations

Notre projet est loin d'être exhaustif, et il y a beaucoup d'amélioration qui peuvent être faites. En premier lieu, certaines des musiques du jeu n'ont pas été implantées, pour ne pas trop allourdir le projet, ce qui laisse quelques blancs dans le jeu. Ensuite, les textures et même les formes des personnages ne sont pas superbes, et mériteraient d'être refaites et améliorées. Un plafond, manquant pour l'instant, sera ajouté. La possibilité de sauvegarder sa progression, la fait que les réponses érronées soient pénalisées et mênent à l'echec du jeu si on en fait trop, ainsi que l'affichage du dossier des preuves, sont des fonctionalités présentes dans le jeu d'origine et qui seront aussi implémentées. Enfin, le netoyage du code, et des évenements personalisés rendrons le jeu plus propre.

4.3 Compétences acquises

Les compétences acquisent durant la réalisation de ce projet sont la manipulation des bibliotheques OpenGL, et d'une grande partie de leurs fonctionalités. L'usage de la SDL, version 2, notamment de la file d'êvenement qui a permis de gerer de manière efficace le deroulement du jeu. Le renforcement de notre connaissance des thread POSIX. La manipulation des chaines de caractères en C afin de pouvoir lire un script, depuis plusieurs fichiers sources, et effectuer des actions en conséquant.

Conclusion

Pour conclure, ce projet fut enrichissant en nous permettant de manipuler une bibliothèque capable de faire des scènes 3D avec un rendu très proche de ce que nous avions prévus de faire grâce aux gestions des textures et de la lumière.

Il fut également enrichissant dans le fait que nous avons réutilisés nos connaissances en système pour pouvoir utilisé les threads POSIX et ainsi bien séparer la partie graphique de la partie lecture de fichiers.

Dans l'avenir, ce projet pourra être repris et étoffé pour en faire un moteur de ce type de jeu ou bien le reprendre comme base pour recréer entièrement le jeu dont s'est inspiré notre réalisation.