Choplifter

TRAORE Oumar

GIRY LATERRIERE Charles

BOYER Clément

Contenu

I. Moteur de jeu

1. Moteur graphique

Nous avons utilisé comme moteur graphique la bibliothèque Simple Directmedia Layer, autrement connue sous le nom de SDL. Ce choix a été motivé par la présence de nombreux tutoriels sur le net, ainsi que la présence d'un grand nombre de bibliothèques complémentant celle-ci.

Ainsi, l'utilisation de SDL nous a permis d'utiliser SDL_Image, et SDL_TTF, deux bibliothèques que nous avons beaucoup utilisées.

SDL_Image nous a permis d'utiliser d'autres extensions d'images que « .bmp », car les fichier .bmp ne nous permettaient pas d'afficher la transparence comme nous le voulions. Cependant, les dernières versions de SDL_image ne fonctionnent pas avec Codeblocks/C, nous avons été obligé de prendre une versions plus ancienne. Nous avons donc pris la version 1.2.4.

SDL_TTF nous a permis d'afficher à l'écran des textes, ce qui nous a permis d'écrire a l'écran le nombre d'otage dans l'hélicoptère/mort/sauvé. Ainsi, nous avons pu implémenté virtuellement une notion de score.

2. Moteur Sonore

Nous avons utilisé la bibliothèque FMOD pour gérer les sons de notre jeu.

La raison pour laquelle nous avons choisi cette bibliothèque était, là encore, la présence de tutoriaux sur le net. SDL intègre nativement une gestion sonore, cependant, de nombreux commentaires nous ont indiqué que cette gestion sonore était très bas niveau et assez peu pratique, nous avons donc préféré choisir FMOD, une bibliothèque plus haut niveau, afin de simplifier la manipulation des sons. De plus, FMOD étant une bibliothèque spécialisée dans la gestion de sons, elle offre beaucoup plus de fonctionnalités que SDL.

3. Map

Les map du jeu sont écrites dans un document texte, afin d'éviter de coder en dur le contenu d'une map. Ce choix nous permet d'éviter la monotonie de notre jeu, et y ajoute un petit côté communautaire, ou des gens peuvent partager les maps qu'ils ont créé.

Ce document est constitué de chiffres, qui indiquent quel sprite afficher à quel endroit.

Nous avons créé un projet d'éditeur de map afin de créer cette map, cependant nous ne l'avons pas fourni dans cette version, car la création de map demandait certaines contraintes au niveau du positionnement des bâtiments que nous n'avons pas intégré dans le code de l'éditeur.

4. Scrolling

Faire du scrolling sur notre map a été pour nous la tâche la plus laborieuse, car nos premières sources de documentation n'en parlaient pas, ce qui nous a poussé à développer notre propre moteur de scrolling en effectuant une lecture sur notre fichier map. Cependant la vitessse n'était pas adaptée vu qu'ont chargeait une image à la fois à chaque deplacement de notre hélicopter, nous avons donc poussé un peu plus la documentation de la Librairie SDL et comprendre qu'on pouvait blitzer notre image à des sources différentes, nous avons donc procédé à un calcul simple pour diviser nos images de 50*50px en 5 et adapté la vitesse de scrolling selon les evènements transmis par notre hélicoptere pour retracher ou ajouter 10px, cependant nous avons remarqué que pour garantir une fluidité il fallait diviser la resolution de notre image par un diviseur de 50, 5 nous a été un bon choix, cela afin de garantir un effet visuel assez toléré.

Un bug est survenu à la fin du projet, dont nous n'arrivons pas à déterminer l'origine. Le scrolling vers la gauche gène le changement de sprite de l'hélicoptère, notamment lors d'un changement de positions (e , r , t), ou l'image suivante ne se blitze pas. Le même problème se voit lors des collisions avec un ennemi, ou le sprite de collision ne se charge pas bien alors que nous sommes effectivement immobilisés.

5. Déplacement des ennemis/Otages

Le déplacement des ennemis et des otages a été fait grâce à des délai et des positions.

En comptant le temps écoulé depuis le dernier mouvement, nous pouvons faire bouger le NPC quand nous le voulons, et nous avons créé une variable qui indiquera dans quelle direction le NPC ira.

Dans le cadre des tanks et des otages, nous avons créé des limites ou le NPC pouvait se déplacer sur la map, et si le NPC atteint cette limite il change de direction. Ainsi, le NPC balaie la zone de la map qui lui est allouée.

Dans le cadre des avions et du boss cependant, nous avons créé un système de phase pour gérer des mouvements plus complexes.

L'otage se dirigera aussi automatiquement vers l'hélicoptère s'il est posé suffisamment prêt de l'otage/de la tente dans laquelle il se trouvait.

6. Déplacement de l'hélicoptère

Le déplacement de l'hélicoptère a été implémenté en récupérant l'état des touches plutôt que l'évènement d'appui des touches. En effet, ce dernier ne permettait pas de déplacement en diagonale et était appelé trop souvent dans le cadre d'un « Poll Event »(le jeu tourne en continu même si aucun évènement est détecté).

Nous avons aussi implémenté un système qui fait descendre lentement l'hélicoptère si aucune touche n'est appuyée. L'objectif est de pousser le joueur à être plus réactif, et de stimuler le joueur en lui montrant que même s'il n'y a rien à l'écran et qu'il est en vol, il se passe quelque chose.

Nous avons implémenté un système de collision avec les ennemis, qui nous immobilisera pendant un cours temps de latence. Nous voulions représenter la perte de contrôle lors d'une collision, cependant cette collision n'empêche pas de tirer.

7. Gestion des tirs

La gestion des tirs est faite à l'aide de 2 fonctions par type de tir (tir de l'hélico/tank/avion/boss).

Une fonction qui initialise le tir, et une fonction qui l'affiche

La fonction qui initialise le tir va commencer par vérifier si une balle peut être tirée grâce a un timer et des conditions.

Si la balle peut être tirée, une valeur lui sera affectée pour savoir dans quelle direction elle va aller parmis 8 directions possibles. Cette fonction lancera alors un son aléatoire parmis 2, et mettra la variable booléenne « tirée » à 1.

La fonction affichage regardera si la variable booléenne « tirée » est à 1, et si c'est le cas, regardera la variable indiquant la direction de la balle, afin de calculer sa prochaine position et blitzera l'image de sa balle au bon endroit.

Nous avons ainsi limité les possibilités de tir des ennemis et de l'hélicoptère :

- L'hélicoptère ne peut tirer au-dessus de lui
- Le tank ne peut tirer en dessous de lui, et ne peut tirer que si l'hélicoptère est assez proche du sol ou au-dessus de lui
- L'avion et le boss peuvent tirer dans presque toute les directions.

II. Références

Ce programme a été créé en utilisant des images et des sons soumis à un copyright, ainsi que des bibliothèques soumises à des licences.

Vous n'avez pas le droit de distribuer/vendre ce programme.

Il est limité à une utilisation personnelle des personnes ayant créé ce programme, leur entourage et les professeurs dans le cadre du rendu de projet.

Provenance des ressources :

- **Musique du jeu** : Klendathu Drop, *Starship Troopers Soundtrack*

- **Musique du menu :** 「ガンバスターマーチ

- , *Gunbuster, Aim for the Top Soundtrack*

- **Sprites des Tanks/Hélicoptères :** série des Metal Slug, jeu

- **Sprite des Boss/Otage :** Dragon Ball Z : L'Appel du Destin, jeu

Les autres sprites (sonores ou graphiques) proviennent de sources libres.