UNIVERSITÉ DE LORRAINE | TELECOM nancy
Ingénieurs du numérique • Inspiring your digital future

Promotion 2016
1A G5

Projet de C

Réalisation d'un jeu vidéo

Valentin Giannini
Eric Perlinski

14/05/2014

Table des matières

Introduction

Le sujet de ce projet étant libre, il nous a été difficile de choisir un sujet qui soit à la hauteur, un sujet qui nous permette de progresser au mieux en programmation C. Cependant, l'utilisation de bibliothèques graphiques et la gestion de celle-ci, nous a paru être un facteur clé dans ce que l'on entendait être un bon sujet.

Pour ce projet de C nous avons donc décidé d'utiliser au mieux cette même bibliothèque et de se lancer dans la réalisation d'un jeu de type "plateformer horizontal", avec une vue en 2D non isométrique, c'est à dire que le joueur sera en vue de côté et la carte du jeu défilera suivant les différents déplacements du joueur.

Nous avons décidé de nous inspirer de différentes grosses licences du jeu vidéo afin de mener à bien ce projet, notamment Super Mario Bros et bien d'autres. Nous avons bien entendu mis en place quelques ajouts tout droits venus de notre imagination.

Le but de ce jeu est simplement d'arriver à la fin de chacun des niveaux présents. Pour ce faire, et suivant le type de carte que nous avons choisi, le joueur devra soit, arriver au bout du niveau, soit tuer tous les ennemis présents sur la carte.

Concernant la bibliothèque graphique utilisée, nous avons tout naturellement choisi la bibliothèque Allegro 5 recommandée par M. Quinson.

Développement

1) principes utilisés

Tout d'abord, nous avons décidé d'utiliser les mêmes principes que nous avons vu en programmation orienté objet, afin de produire un code le plus propre possible. De ce fait, et bien qu'en C, il n'y a pas de principe objet, nous avons utilisé une série de structures permettant, d'émuler quelques peu le comportement des objets présents dans des langages comme Java. De nombreuses fonctions associées à chacune des structures permettent donc de piloter celles-ci.

La boucle de jeu, c'est-à-dire ce qui permet au jeu de fonctionner et de faire des rotations de maps, est plutôt simple :

Premièrement, on récupère les évènements du joueur soit via le clavier soit via une manette XBox360. On met donc à jour le personnage suivant les évènements faits par le joueur, et on calcul ensuite toutes les différentes collisions et méthodes de gameplay associées au personnage ainsi qu'aux différents ennemis, la carte et les projectiles. Une fois mis à jour, on affiche tout cela et on recommence

2) détails de l'architecture

Nous allons maintenant voir plus en détails le fonctionnement de chacune de ces structures constituant les différents éléments du jeu :

Map :

Cette structure permet de gérer la carte. Notamment pour la dessiner et pour la déplacer dans l'espace. Une map n'est autres qu'un tableau à 2 dimensions de Tiles.

Ces Tiles sont des petites images qui représentent des bouts de la carte (voir l'image sprite_map.png), elles sont récupérées dans une image contenant toutes les Tiles et sont triées d'une certaine façon (chaque ligne a une dimension spécifique).

Personnage :

Cette partie permet de gérer le personnage principal ainsi qu'une partie des ennemis. Cette entité définit entièrement le personnage comme ses coordonnées sur la carte, les points de vie restants, l'image associée au mouvement actuel du personnage ainsi que les armes qu'il possède. Le personnage peut se déplacer de manière horizontale et peux également sauter et nager.

Ennemi :

Cette structure gère l'intelligence artificielle des ennemis du jeu. Un ennemi est composé d'un personnage et de fonctions permettant d'exécuter certaines actions par rapport à la position du joueur. Par exemple, un ennemi possède un certain champ de vision qui peut être plus ou moins grand ainsi qu'un angle de vision qui lui aussi peut être plus ou moins large.

Suivant le fait que le joueur se trouve dans le cône de vision d'un ennemi ou non, ce même ennemi agira différemment. Si l'ennemi ne voit pas le joueur, il est alors en état de recherche et se retourne au bout d'un certain temps prédéfini. Au contraire, si le joueur entre dans ce cône de vision, alors que l'ennemi est face au joueur, l'ennemi va alors rentrer dans le mode attaque.

Dans ce mode, l'ennemi fera tout pour être au contact du joueur, c'est à dire en sautant si cela est nécessaire, afin d'atteindre une distance minimal avec le joueur. De plus, celui-ci tirera vers le joueur repéré, suivant un intervalle de temps prédéfini en fonction de la difficulté.

Boss :

Cette structure n'est autre qu'une structure, très similaire aux structures ennemis et personnage, et qui permet de représenter un ennemi beaucoup plus puissant. Nous avons choisi de représenter ce boss par un avion se déplaçant uniquement de manière horizontale, c'est à dire non soumis à la gravité.

Event :

Cette partie gère les évènements de l'utilisateur, qu'ils passent soit par le clavier ou soit par une manette de XBOX360, cette structure permet donc de mettre à jour tous les inputs faits par le joueur.

Nous avons choisi de développer cette partie de façon modulaire pour permettre d'ajouter d'autres périphériques comme une manette de PS3, un joystick ou pourquoi pas l'Oculus Rift avec Virtuix Omni. En effet, cette modularité se traduit par une transparence dans les méthodes associées à la récupération d'inputs de la part du joueur.

Cependant, et du fait que nous ne possédons pas d'autres périphériques similaires, seulement la manette de Microsoft n'a été testé.

Projectile :

Cette structure permet de représenter les différents projectiles tirés par les ennemis, les boss ou encore le joueur. Ils ont une certaines puissance et vitesse qui peut également être défini.

Menu :

Cette structure permet de gérer le menu du jeu qui permet de mettre en pause, de modifier le périphérique utilisé, de quitter le jeu ou même de créer une carte aléatoire.

Jeu :

Cette partie permet de faire tourner le jeu suivant le cycle défini dans la partie « principes utilisés (p.4)». Il contient 3 fonctions principales permettant de gérer le jeu dans son intégralité. En premier lieu, nous pouvons retrouver la fonction essentielle jeu_parse qui permet, à partir d'un fichier map.txt, de charger tous les différents éléments du jeu (ennemis, map, joueur).

En second lieu, on peut retrouver la partie de traitement des évènements, où l'on récupère ce que l'utilisateur a fait afin d'appliquer les différentes fonctions adéquates sur le joueur, le menu ou encore les ennemis.

Et pour finir, on peut trouver la fonction de dessin, où l'on calcul les collisions et on affiche tout ce qui doit être affiché.

Main :

Cette fonction permet d'initialiser Allegro et de lancer la boucle de jeu. Notons également que l'on retrouve en quelques sortes la "super structure" permettant de charger les différentes cartes du jeu et de les enchaîner les unes à la suite des autres. Cette "super structure" notée "game", comprends un jeu, et la liste des différentes cartes, ainsi que différents paramètres permettant de gérer le jeu au mieux.

Customlinkedlist :

Afin de conserver la plupart des ennemis, projectiles, ou encore le nom des maps à charger, nous avons choisi d'utiliser une double liste chaînée. Nous avons donc crée une structure customlinkedlist permettant de gérer au mieux tous ces différents éléments.

3) Le gameplay

Les touches :

Comme nous l'avons dit précédemment nous avons implémenté la possibilité au joueur d'utiliser deux types de périphériques différents afin de jouer au jeu.

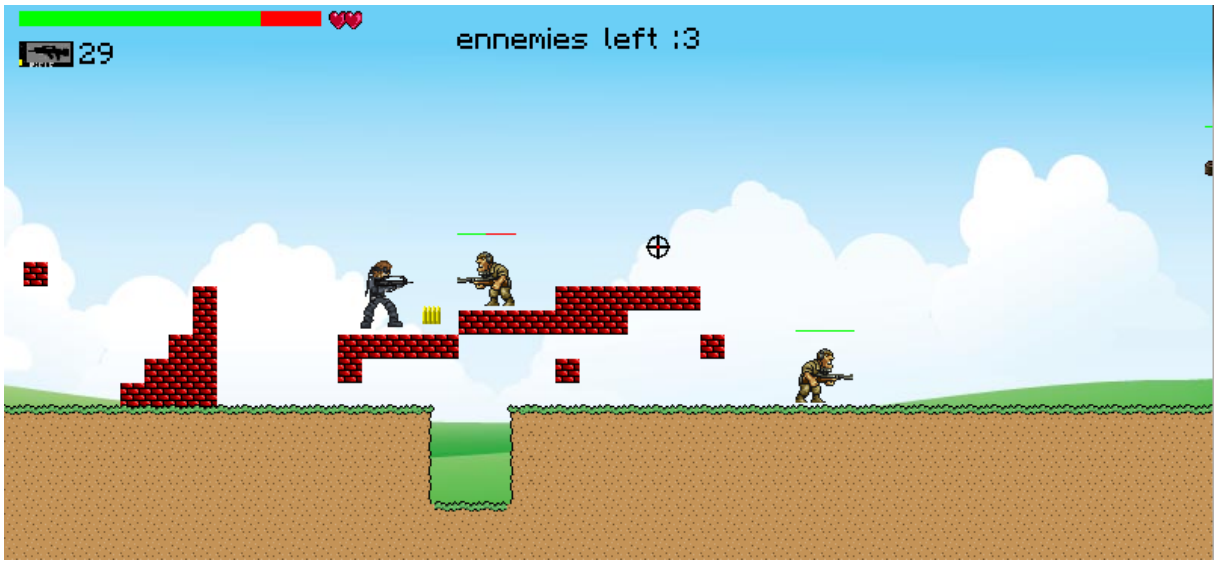
a) Clavier

- Q : Gauche
- D : Droite
- Espace : Sauter
- 1/& : changer d'arme
- Bouton gauche souris : Tirer
- Curseur de la souri : Viser
- Echap : menu

b) Manette XBox360

- Joystick gauche : déplacement joueur
- Joystick droit : déplacement viseur
- LB : sauter
- B : changer d'arme
- RB : tirer
- Y : activer/désactiver musique
- Start : menu

L'interface :



Ce que nous pouvons voir est l'actuelle interface du jeu, en haut à gauche se trouve la barre de points de vies du joueur, ainsi que le nombre de continus restants.

Nous avons également affiché l'arme courante tenue par le joueur ainsi que le nombre de balles restantes. Concernant l'objectif de fin de niveau, c'est-à-dire suivant le type de map défini, le joueur devra soit tuer tous les ennemis sur la carte, auquel cas les mots « Ennemies left : » suivi du nombre d'ennemis restants apparaîtront. S'il s'agit du second type de map, les mots « Go to the end ! » seront affichés et le joueur devra simplement atteindre la fin du niveau, sans pour autant tuer tous les ennemis présents.

Eléments de gameplay :

Outre le fait qu'il y ait différents types d'ennemis et d'arme, il nous a paru essentiel d'implémenter différents éléments de gameplay tels que des kits de soin ou de munitions. De ce fait, lorsque le joueur débute, il possède un certain nombre de points de vie matérialisés par sa barre de points de vie, lorsque celui-ci tue un ennemi ou touche un boss, des trousses de soins ou des kits de munitions ont une chance d'apparaître aux pieds de l'ennemi ou au cas échéant, aux coordonnées du boss lui-même.

De plus, nous avons également jugé nécessaire de pouvoir afficher les différents éléments concernant les points de vie restants des différents ennemis. De ce fait, une barre de points de vie relative à chaque ennemi, est affichée au-dessus de la tête de ceux-ci.

Afin d'offrir un gameplay davantage immersif, nous avons également pris en compte les différents éléments sonore nécessaires au gameplay même de tout jeu vidéo. Ainsi, un fond sonore provenant de

différents sites proposant de la musique ou des effets sonores libres de droits consacrés aux jeux indépendants ont été sollicités. De ce fait, à chaque tir du joueur et suivant le type d'arme utilisée par le joueur, ou encore à chaque tir ennemi, une sonorité d'arme à feu a été utilisée.

4) Le travail restant.

Bien que le jeu soit fonctionnel, nous n'avons pas pour autant réussi à terminer et à réussir tous les différents objectifs et contraintes que nous nous étions fixées. En effet, concernant les boss du jeu, nous avions prévu au départ, qu'il suive différents pattern d'attaque et de déplacement, permettant au joueur d'avoir une expérience d'autant plus enrichissante.

La configuration des touches suivant les paramètres personnels du joueur avait également été envisagée. En effet, nous avions pour projet de permettre au joueur de mapper ses propres touches, mais nous avons préféré nous concentrer sur les différents éléments de gameplay.

Il avait été également prévu à la base que le joueur ou encore les ennemis puisse se servir d'outils tels que des mitrailleuses fixes ou encore de véhicules comme des jeeps ou des avions. Cependant cela n'a pas pu être réalisé. Un système de scoring, ou encore un timer global et des statistiques avaient de même été envisagés.

Conclusion

Nous avons vraiment apprécié travailler sur ce projet notamment pour l'approfondissement de notre culture en matière de jeu vidéo (boucle de jeu, gestion de collisions, ...) mais surtout pour l'expérience acquise en C et l'utilisation d'une bibliothèque graphique telle qu'allegro5. De plus, avec un sujet libre comme celui-ci, nous avons véritablement pu développer sans contraintes, ou plutôt avec les contraintes que nous nous étions imposées, et ceci de la manière la plus efficace possible.

Cela a véritablement été une expérience plus qu'enrichissante du point de vue programmation, mais également en matière de gestion de projet.

Ce projet a été l'occasion de nous familiariser davantage avec les techniques et méthodes relatives à la réalisation d'un jeu vidéo en 2D ainsi qu'avec toutes les autres fonctions d'Allegro 5, telles que l'importation de musiques, d'images, ou encore l'affichage d'une interface graphique et la gestion des évènements du joueur. Toutes les techniques vues en cours nous ont permis d'aborder ce projet de la manière la plus réfléchie possible, et nous ont permis de mener à bien ce projet.