Université de Franche-Comté

Rapport de projet tuteuré
Licence informatique 3ème Année

Création d'un Third-Person Shooter avec le Shine Engine

Virgil Manrique Quentin Guillien

Encadrant :
Sylvain Grosdemouge

Année 2015-2016

Remerciements

Ce projet tuteuré n'aurait pu être réalisé sans l'aide de plusieurs personnes que nous tenons à remercier.

Tout d'abord, nous remercions notre encadrant M. Sylvain Grosdemouge pour nous avoir guidé et prodigué ses conseils durant la réalisation de ce projet.

Ensuite, nous voulons remercier M. Bastien Schatt, pour nous avoir aidés lors de notre apprentissage de l'utilisation du Shine Engine ainsi que pour sa réactivité lorsque nous avons rencontré des problèmes.

Nous voulons également remercier M. Nicolas Diot, pour son assistance apportée par rapport au fonctionnement de l'éditeur de niveaux du Shine Engine et des ressources graphiques.

Nous tenons enfin à remercier toutes les personnes qui nous ont aidés de près ou de loin.

Table des matières

1 Cadre du projet

Dans le cadre de notre 3ème année de licence informatique à l'université de Franche-Comté, il nous a été demandé de réaliser un projet dans le cadre du module projet tuteuré. La durée du projet s'étendait d'Octobre à Mars.

Parmis les sujets proposés, l'un d'entre eux a particulièrement retenu notre attention. Il s'agissait du développement d'un Third-Person Shooter en utilisant un moteur jeu : le Shine Engine développé par M. Sylvain Grosdemouge.

Nous avons pu nous voir attribuer ce sujet l'ayant placé en première position dans notre liste de choix.

Pour comprendre l'intérêt que nous portions à ce sujet, il faut nous pencher sur les différents éléments qui le constituent, à savoir les Third-Person Shooter, les moteurs de jeu, et le Shine Engine.

1.1 Les Third-Person Shooter

Un TPS (Third-Person Shooter ou jeu de tir à la troisième personne en français) est un sous-genre des jeux de tir et donc des jeux d'action. Les jeux de tir mettent souvent la rapidité et la réactivité du joueur à l'épreuve. L'objectif de ce genre de jeu est de vaincre ses ennemis en utilisant une arme de tir. La particularité des `TPS`[1] est que le joueur voit son personnage de manière externe contrairement aux `FPS`[2] où le joueur voit à travers les yeux de son personnage.

Figure 1.1 – Un exemple de `TPS` : Resident Evil 4

1. **T**hird-**P**erson **S**hooter
2. **F**irst-**P**erson **S**hooter : Jeu de tir à la 1ère personne

1.2 Les moteurs de jeu

"Un moteur de jeu est un ensemble de composants logiciels qui effectuent des calculs de géométrie et de physique utilisés dans les jeux vidéo. L'ensemble forme un simulateur en temps réel souple qui reproduit les caractéristiques des mondes imaginaires dans lesquels se déroulent les jeux. Le but visé par un moteur de jeu est de permettre à une équipe de développement de se concentrer sur le contenu et le déroulement du jeu plutôt que la résolution de problèmes informatiques."

– *Définition Wikipédia*

1.3 Le Shine Engine

Le Shine Engine est un moteur de jeu créé par notre encadrant, Monsieur Sylvain Grosdemouge. Monsieur Grosdemouge a commencé le développement du Shine Engine en 2005 et en 2012 est sorti R.A.W.(Realm Of Ancient War), premier jeu développé en utilisant le Shine Engine. Ce moteur de jeu est développé en C++ et permet de faire du développement multi-plateforme. Il permet aussi de gérer facilement la 3D ce qui permet de s'affranchir de beaucoup de limites pour le développement d'un jeu.

1.4 Le Projet

Les trois éléments cités précédemment rendaient pour nous le projet attrayant : les TPS sont un genre de jeu au concept simple mais distrayant ; utiliser un moteur de jeu permet de se concentrer sur les mécaniques de jeu, qui sont pour nous la partie la plus intéressante du développement d'un jeu ; et enfin, le Shine Engine, un moteur qui a fait ses preuves, qui est efficace et qui nous permet de développer en C++, le langage que nous préférons.

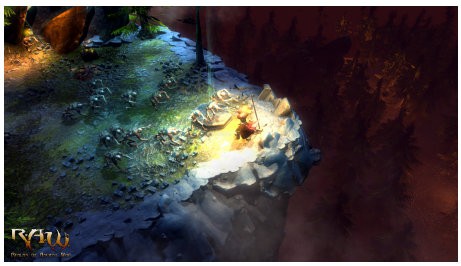

Figure 1.2 – Realm Of Ancient War

2 Préparation

2.1 Installation des outils

Avant de commencer à coder, il a d'abord fallu installer plusieurs outils logiciels.

Tous les outils mentionnés ont été utilisés, dans notre cas, sur Windows uniquement.

2.1.1 Les indispensables

Les trois logiciels indispensables sont le `Shine SDK`[1], le `Shine Editor`[2] et `Microsoft Visual Studio 2010`.

Le `Shine Editor` et le `SDK` ne requérant pas d'installation, n'ont pas posé de problèmes dans un premier temps. En revanche, il n'a pas été facile de retrouver la version 2010 de `Visual Studio`, car aujourd'hui, Microsoft ne propose que la version *Community* de leur logiciel. `Visual Studio` est un `IDE`[3]. Il en existe d'autres, mais le `Shine Engine` a été développé et prévu pour être intégré à `Visual Studio`.

Même après avoir installé les outils `Shine`, il ne peuvent pas encore être exécutés indépendamment. En effet, ils requièrent le `DirectX SDK` mais aussi `Microsoft .NET Framework` (version 4.0 ou supérieur).

2.1.2 Pour plus de confort

Bien qu'ils ne soient pas réellement indispensables, les outils suivants nous été extrêmement utiles.

Nous avons utilisé `Git`, plus précisément `Github`, l'application graphique de Git pour Windows. `GitHub` est un service web d'hébergement et de gestion de développement de logiciels utilisant Git, le logiciel de gestion de versions décentralisé.

Nous avons également utilisé `Trello`, un outil de gestion de projet en ligne permettant d'assigner facilement des tâches à des utilisateurs.

1. SDK = Software Development Kit (trousse de développement logiciel en Français)
2. Editeur graphique de Shine permettant d'ajouter facilement des éléments visuels dans un jeu
3. IDE = Integrated Development Environment (Environnement de Développement en Français)

GIMP [4] est un outil d'édition et de retouche d'image. GIMP a été utile pour créer des `sprites` [5] simples.

En complément de GIMP nous avons utilisé `XnView`, qui nous a permis de vérifier le formats de certains fichiers, et de convertir au bon format si besoin.

Enfin, nous avons utilisé LaTeX pour la rédaction de ce rapport.

2.2 Formation au Shine Engine

Pour apprendre à utiliser le moteur, nous avons assisté aux formations au Shine Engine dispensées par Shine Research au Cub' à Essais, à l'incubateur d'entreprises innovantes de Franche-Comté, tous les jeudis à 17h pendant 10 semaines. Ces formations nous ont été utiles pour apprendre les bases du fonctionnement du Shine Engine et ainsi pouvoir maitriser par nous-même des fonctionnalités plus avancées.

4. GNU Image Manipulation Program

5. Elément graphique qui peut se déplacer sur l'écran. Dans notre cas, cet élément était une simple image en deux dimensions

3 Réalisation

Dans ce chapitre nous allons expliquer chaque élément constituant le jeu, que ce soit le joueur ou les ennemis, les armes et leurs munitions ou bien la gestion des collisions.

3.1 Jeu ou Plugin ?

Pour réaliser ce projet, notre encadrant nous a demandé de développer le jeu non pas de manière classique mais en utilisant la fonctionnalité de plugin du Shine Engine. Nous avons donc développé les mécaniques de jeu en gardant en tête que la fonction de plugin permettait d'utiliser le plugin simplement en l'activant lors de la création d'un niveau, par exemple en utilisant l'éditeur de Shine, ce qui nous a poussé à développer de manière générique et non pas en fonction du niveau. Monsieur Grosdemouge nous a donné des sprites simples pour faire un niveau de test mais nous avons par la suite réalisé d'autres sprites pour faire d'autres niveaux afin de vérifier le bon fonctionnement du plugin.

3.2 Mécaniques de jeu

3.2.1 Les personnages

Le plugin doit pouvoir gérer deux types de personnages : le personnage du joueur, contrôlé par ce dernier et le ou les personnages ennemis, qui disposent de leur propre IA. Pour cela nous avons crée une classe générique Character de laquelle hérite les classes Player et Enemy. Cette classe a pour grande utilité de permettre un traitement générique ainsi que la possibilité d'une modification générique. Ainsi lorsque nous avons voulu implémenter la gestion de la 3D, il nous a suffit d'ajouter un attribut à cette classe pour stocker le modèle 3D et le joueur ainsi que les ennemis ont été dotés de cet attribut. Un personnage peut posséder une arme et tirer avec celle-ci, nous détaillerons cela plus en détail dans la partie consacrée à la classe associée aux armes, à savoir la classe Gun. Un personnage peut aussi mourir s'il entre en contact avec un projectile, exception faite des projectiles qu'il a tiré lui-même.

3.2.2 Le Joueur

Pour bien différencier le joueur des ennemis, et puisque les deux n'ont pas le même fonctionnement, nous avons crée une classe Player qui hérite de la classe Character, la rendant similaire donc à la classe Enemy mais dénué d'IA et contrôlé à l'aide des touches du clavier. Le joueur peut ainsi faire avancer son avatar avec la flèche haut ou la touche Z. Les touches flèche gauche (ou la touche Q) et flèche droite (ou la touche D) permettent d'effectuer une rotation respectivement vers la gauche ou la droite. Le joueur peut aussi reculer en utilisant la touche flèche bas (ou la touche S) mais se déplace dans ce cas à la moitié de sa vitesse. Initialement notre encadrant ne nous avait pas demandé de permettre au joueur de reculer mais nous avons rajouté cette fonctionnalité pour un meilleur confort de jeu après avoir fait quelques test lors de l'implémentation des projectiles et de la possibilité pour l'avatar de mourir.

3.2.3 Les Ennemis

La classe Enemy permet de représenter un ennemi auquel le joueur peut-être confronté. Le plugin gère une liste d'ennemis pour représenter tous les ennemis d'un niveau. La classe Enemy héritant de la classe Character, un Enemy est sensiblement identique au joueur mais à un fonctionnement différent : en effet il est contrôlé par une IA.

L'IA ne fut pas simple à mettre au point bien qu'elle ait elle-même un fonctionnement simple : si l'ennemi ne voit pas le joueur, il reste sur place, si il le voit, il se déplace vers lui tout en tirant. L'IA est basée sur un automate à deux états, l'état attaque et l'état repos. L'état repos est son état par défaut et il passe en état d'attaque lorsqu'il voit le joueur.

L'ennemi possède un attribut Target qui représente la dernière position connue du joueur. A chaque update du jeu, le gestionaire de collisions va vérifier si l'ennemi a une ligne de vue directe sur le joueur. Si c'est le cas, il met à jour l'attribut Target de l'ennemi. Ce faisant, l'ennemi va passer en état d'attaque s'il ne l'est pas déjà et va se rendre jusqu'à la position cible tout en tirant en direction de celle-ci. Si en cours de chemin il perd sa ligne de vue directe avec le joueur, il se rendra tout de même à sa dernière position connue, ce qui peut lui permettre de voir à nous le joueur. Nous avons doté les ennemis d'une vision à 360° pour pouvoir repérer le joueur.

3.3 Problèmes rencontrés

3.3.1 Problèmes logiciels

Nous l'avons mentionné dans la partie Installation (page 5), trouver la version complète de `Microsoft Visual Studio 2010` n'a pas été tâche facile.

Cependant, cela n'a pas été le plus important des problèmes logiciel que nous avons eu. En effet, il nous est arrivé par exemple de ne plus pouvoir lancer l'éditeur de `Shine` alors qu'il marchait auparavant, sans savoir pourquoi. Réinstaller l'éditeur, ou installer une version plus récente résolvait en général les problèmes rencontrés. Il nous est également arrivé de simplement ne pas du tout pouvoir faire fonctionner le `Shine Editor` sur un PC (la seule option était ici d'utiliser un autre PC).

3.3.2 Problèmes de code

Le problème majeur qui nous a couté le plus de temps est celui-ci :

```
1   /// @todo comment
```

Cette ligne, combinée à l'inexistence de documentation, nous a souvent laissé dans le flou quand à la manière de procéder pour effectuer des tâches parfois assez simples. La seule chose dont on pouvait s'aider était les fichiers d'entête dans les dossier du `Shine SDK` : en lisant les définitions des méthodes et les noms des paramètres, on pouvait au mieux *deviner* la façon d'utiliser ces méthodes.

Bien sûr, quand on ne pouvait plus avancer, nous avons pris contact avec l'équipe de `Shine`. Cependant, la réponse n'étant évidemment pas immédiate, il nous est arrivé de rester bloquer des après-midis entiers à essayer de faire marcher un bout de code.

4 Bilan

4.0.1 Optimisations Possibles

Ennemis avec cône de vision