Compte rendu projet Portal 0.0

BIDAULT Matthieu, BADSTÜBER Elian, FOCHEUX Vital

February 28, 2024

Remerciements

Nous tenons à remercier notre enseignant de projet, M. Bernard qui a su nous guider et nous conseiller tout au long de ce projet.

Table des matières

I Introduction **3**
II Besoins et objectifs du projet **3**
II.i Contexte 3
II.ii Motivations 6
II.iii Objectif et contraintes 6
III Gestion du projet **7**
III.i L'équipe 7
III.ii Planification et outils de gestion 7
III.iii Répartition des tâches 7
IV Développement **7**
IV.i Programmer en C++ 7
IV.ii Apprendre à utiliser la bibliothèque GF . . . 7
IV.iii Différentes stratégie 7
IV.iv Détails du développement 8
V Conclusion technique & personnelle **11**
V.i Bilan du projet 11
V.ii Perspectives 12
VI Bibliographie **12**

Figure 1: Knights of the Sky

Figure 2: Hovertank 3D

I Introduction

Dans le cadre du projet semestriel de troisième année de licence informatique à l'université de Franche-Comté, nous proposons de coder une version très simplifiée du jeu Portal qui sera afficher grâce à un moteur de type Raycaster à la façon du jeu Wolfenstein 3D. Il a pour but de nous faire découvrir le monde du développement de jeux vidéo en nous faisant réaliser un jeu vidéo en C++ avec la bibliothèque Gamedev Framework (GF)[1].

II Besoins et objectifs du projet

II.i Contexte

Les graphismes : Au début des années 90, la société Id Software a entrepris des recherches pionnières dans le domaine des graphismes 3D, alors principalement réservés aux simulateurs de vols tels que Wing Commander ou Knights of the Sky (Knights of the Sky), deux titres parus en 1990. Face aux contraintes de performance des ordinateurs de l'époque, le développement de jeux d'action en 3D rapide représentait un défi de taille. C'est dans ce contexte que John Carmack a proposé l'utilisation de la technique du raycasting, permettant de calculer uniquement les surfaces visibles par le joueur. En six semaines, Carmack développe un moteur 3D innovant utilisant des sprites 2D

[1]Tous au long de ce rapport, nous utiliserons abbrévation GF. Cela signifie Gamedev Framework.

Figure 3: Ultima Underworld

Figure 4: Wolfenstein 3D

Figure 5: Doom (1993)

Figure 6: Quake (1996)

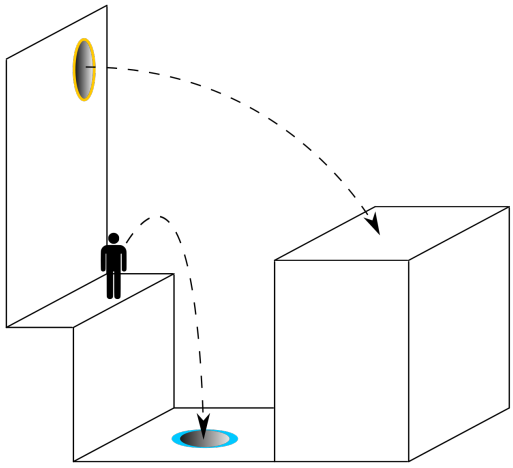

Figure 7: Schéma de fonctionnement du système de portail

pour représenter les entités du jeu. Ce moteur a été utilisé dans le jeu Hovertank 3D (Hovertank 3D), publié en avril 1991. À l'automne 1991, alors que John Carmack et John Romero finalisaient le moteur de Commander Keen in Goodbye, Galaxy, Carmack découvre Ultima Underworld (Ultima Underworld), un jeu développé par Blue Sky Productions (qui deviendra plus tard Looking Glass Studios), doté d'un moteur capable de rendre des graphismes 3D texturés sans subir les limitations de Hovertank 3D. Inspiré, Carmack décide d'améliorer son propre moteur pour intégrer le mapping de textures tout en conservant de hautes performances. Après un intense travail de six semaines, le nouveau moteur 3D est achevé et utilisé pour le jeu Catacomb 3D, publié en novembre 1991. La révélation de Catacomb 3D a poussé Scott Miller d'Apogee à convaincre l'équipe de développer un jeu d'action en 3D sous forme de shareware. Cela a conduit au lancement du projet Wolfenstein 3D (Wolfenstein 3D), un remake en 3D de Castle Wolfenstein. Sorti le 5 mai 1992 sur PC, ce jeu a non seulement été un succès commercial mais a également posé les bases du genre du jeu de tir à la première personne, préfigurant ainsi des titres légendaires tels que Doom (Doom (1993)) et Quake (Quake (1996)).

Le système de jeu : En 2007, Valve Corporation a révolutionné le genre des jeux de réflexion à la première personne avec la sortie de Portal. Ce jeu introduit un mécanisme unique permettant au joueur de générer deux

portails, l'un orange et l'autre bleu, sur des surfaces planes et interconnectées. Ces portails offrent la possibilité de traverser instantanément l'espace d'un point à un autre, tout en conservant l'inertie. L'objectif est de résoudre divers puzzles en se servant de cette capacité à manipuler l'espace pour atteindre la sortie des différents niveaux proposés.

II.ii Motivations

L'une des motivations principales de ce projet est de réaliser un jeu vidéo avec graphique comme à l'époque de Wolfenstein 3D mais avec des comcepts de jeux plus récentes et qui plus est pourrai être un préquel de Portal.

II.iii Objectif et contraintes

Les principaux objectifs de ce projet sont :

- Réaliser un jeu vidéo en C++ avec la bibliothèque GF
- Implémenter un moteur de type Raycaster
- Implémenter un système de portail
- Implémenter un système de collision
- Offir une expérience de jeu simple et agréable à jouer.

Mais avec des contraintes :

- L'apprentissage d'un langage de programmation nouveau pour nous
- L'apprentissage d'une bibliothèque de programmation nouvelle pour nous
- L'apprentissage de la programmation d'un moteur de type Raycaster
- L'apprentissage de la programmation d'un système de portail
- L'apprentissage de la programmation d'un système de collision
- L'apprentissage de la programmation d'un système de jeu

III Gestion du projet

III.i L'équipe

III.ii Planification et outils de gestion

Pour la gestion du projet nous avons utilisé le site github qui est un outil de gestion de projet en ligne.

III.iii Répartition des tâches

IV Développement

IV.i Programmer en C++

Programmer en C++ a pu s'avérer assez techniques car cela était un tout nouveau langage de programmation à apprendre pour nous car il n'avait jamais été abordé auparavant lors de notre parcours universitaire.

IV.ii Apprendre à utiliser la bibliothèque GF

Une des plus grandes difficultés de ce projet était de développé avec la bibliothèque GF. En effet, quand bien même c'est un bibliothèque possèdant un bon nombre de fonctions pouvant permettre le développement graphique 2D. Elle reste néanmoins parfois limité pour les besoins techniques du projet, mais grâce à cela, cela permet de donner des idées de méthodes à rajouter dans GF voire dans GF2.[2]

IV.iii Différentes stratégie

Première approche

Une première approche du projet a d'abord été envisagée. Elle consistait basiquement à envoyer plus de rayons possibles à partir de la vision du joueur afin de permettre le rendu des entités ainsi que des textures.

[2]GF2 est un projet qui est la suite de GF en C++ ainsi qu'en python3.

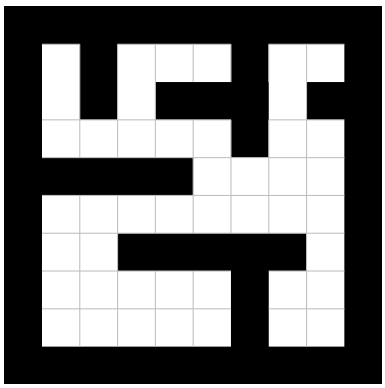

Figure 8: PNG d'une map

Cette approche étant facile et simpliste à implémenter une nouvelle approche du projet a été prise en compte.

Seconde approche plus technique

Cette nouvelle approche de nôtre projet repose sur quelques même principe que la première mais tout en étant plus technique sur le plan mathématiques et informatique.

IV.iv Détails du développement

Dans cette partie nous allons détailler les différentes étapes de développement de notre projet avec la seconde approche. En commencençant par la partie la moins technique pour finir par la partie la plus technique. Il faut noté que détails du développement dans les parties qui vont suivre sont pour un rendu 3D, mais tout au long de ce projet les différentes étapes ont été tester sur un rendu 2D pour faciliter le développement.

Les maps:

Afin de pouvoir afficher des maps[3], nous avons dû créer un fichier PNG[4] où nous dessinons les murs ainsi que la case de départ et d'arrivée (PNG d'une map). Lors du lancement du jeu, le programme lit le fichier PNG et

[3]En français les cartes
[4]Portable Network Graphics

enregistre les différentes cases dans un tableau à deux dimensions. En même temps, le programme va regarder si les coordonnées correspondent à une cellule de mur, la case de départ, la case d'arrivée ou aucune des deux. Si c'est une cellule de mur, alors on va effectuer un parcours en profondeur des coordonnées adjacentes pour savoir si ceux sont des cellules de murs ou non. Grâce à cela nous pouvons récuperer toutes les cellules d'un mur directement et ainsi compter le nombre de mur assez facilement.

Les murs:

Dans cette partie nous allons détaillés les différentes étapes pour la création des murs.

Récupération des sommets utiles: Pour la création des murs, on effectue un boucle sur la liste de coordonnées des cellules occupées pour un mur précédemment récupérée. Sur chacune des coordonnées, on va regarder quatre de ses coordonnées adjacentes qui sont celles en haut, en bas, à gauche et à droite. Si l'une d'entre elles fait partie de la liste des coordonnées des cellules alors on va incrémenter un compteur. Si ce compteur est égal à 1 ou 3 alors on va l'ajouter dans une liste de coordonnées dites utiles, c'est-à-dire que ceux sont les coordonnées des sommets des murs.

Triage des sommets: Afin de faciliter le rendu des murs, nous avons besoin de trier les sommets. Pour cela, nous effectuons un boucle tant que la taille de la liste des sommets utiles est supérieur à la taille de la liste des sommets triés. Dans cette boucle, nous allons effectuer une boucle sur la liste des sommets utiles si le sommet est dans la liste des sommets triés ou non. Si il ne l'est pas alors on va récupérer les coordonnées de ce sommet et faire appel à une fonction pour trier à partir de ce sommet, la liste des sommets utiles et la liste des sommets triés. Dans cette fonction, on va effectuer une boucle sur la liste des sommets utiles à partir du sommets donné en paramètre. On va regarder pour chaque sommet si il existe un sommet dans la direction choisie ainsi que dans sa direction opposée. (Explication de la fonction fctCanGo?)

- Si il existe un sommet dans la direction choisie et aucun sommet dans la direction opposée alors on va ajouter ce sommet dans la liste des sommets triés.

- Sinon si il existe un sommet dans la direction opposée et aucun sommet dans la direction choisie alors on va ajouter le sommet de la direction opposée dans la liste des sommets triés et changer la direction courante par la direction opposée.
- Sinon
  - Si il existe un sommet dans la direction choisie et un sommet dans la direction opposée alors on regarder si le sommet de la direction opposée n'est pas dans la liste des sommets triés et qu'un changement de direction doit être effectué ou bien que le sommet de la direction choisie est dans la liste des sommets triés et qu'un chanegement de direction ne doit pas être effectué alors on va ajouter le sommet de la direction opposée dans la liste des sommets triés et changer la direction courante par la direction opposée.
  - Sinon on va ajouter le sommet de la direction choisie dans la liste des sommets triés.

On va changer de direction en fonction de la direction courante, si le changement de direction reste le même que l'ancien alors on va changer de direction. Si n'existe aucune sommet dans les directions choisies et opposées alors on va incrémenter un compteur. Si ce compteur est supérieur ou égal à 4 alors on arrête la fonction de triage.

Contruction des murs: Une fois les sommets utiles triés nous allons pouvoir construire les murs et les ajouter dans une liste de murs. Chaque murs de cette liste a pour attributs les coordonnées de ses sommets triés, les coordonnées de ses cellules occupées et la taille de son périmètre.

La boucle de jeu:

Dans cette partie nous allons détailler les différentes étapes de la boucle de jeu.

Events: Nous allons tous d'abord gérer la gestion des évènements. Pour cela nous allons effectuer une boucle sur les évènements et pour chaque bouton sur lesquels nous avons ajouté une action, nous allons effectuer cette action. Nous allons aussi gérer les fonctions de déplacement de la vision

grâce à la souris avec la position relative de la souris. Ensuite, nous allons gérer la fonction d'envoie des portails en fonction du clic de la souris effectuer.

Update: Nous allons effectuer une update du jeu par rapport au temps écoulé depuis le dernier update. Cette update va permettre de mettre à jour (les attributs du joueur). De calculer la nouvelle position du joueur si il est en colision avec un mur. Et enfin calculer les nouveaux attributs du joueur si il est proche d'un portail lorsque les deux portails sont activés.

Render: Avant de faire le nouveau rendu des entités, nous allons d'abord effacer tous les rendus précédents. Enuiste nous allons effectuer le rendu de toutes les entités que nous allons détaillés dans la parties suivante.

Les renders:

Ici explication de chaque rendu des différentes entités

Le rendu des murs: Pour le rendu des murs, nous allons effectuer une boucle sur la liste des sommets triés. Pour chaque sommet, nous allons dessiner une ligne blanche entre le sommet suivant dans la liste des sommets triés et le sommet courant. Si nous sommes à la fin de la liste des sommets triés alors nous allons dessiner une ligne blanche entre le sommet courant et le premier sommet de la liste des sommets triés.

Les collisions:

Explication des collisions avec les murs et les portails avec les schémas.

Les portails:

Explication des portails avec les schémas.

V Conclusion technique & personnelle

V.i Bilan du projet

Résultats obtenus

Apports technique

Apports personnels

V.ii Perspectives

Améliorations possibles

La plus grande amélioration possible serait de pouvoir voir à travers les portails lorsque les deux sont activés. (Bref explication de comment cela serai possible)

Nouvelles fonctionnalités

Parmi de nouvelles fonctionnalités, que nous pourrions rajouter en voici quelques unes:

- Permettre à l'utilisateur de créer ses propres maps
- Avoir des murs où il est impossible de tirer un portail
- Avoir des endroits au sol que lorsqu'on marche dessus, on est téléporter à la case départ
- Pouvoir mettre plusieurs textures différentes sur les murs
- Avoir des murs qui possèdent des trous permettant de voir ainsi que tirer des portails à travers

Un plus dans nos CV

VI Bibliographie

Gamedev Framework (GF): https://gamedevframework.github.io/