**Rapport de projet : To the moons**

Mesquita Quentin, Goulin Jolan

Sommaire

Présentation

Mode d’emploi

Architecture logicielle

Gestion de la map

Design pattern

Extensions du projet

Côté technique

Gestion du projet

Présentation :

To the moons est un jeu codé en c++ en utilisant l’OpenGL. C’est un jeu basé sur le principe du jeu temple run.

Dans ce jeu la Terre, n’ayant qu’une seule amie, la lune se sent seule. Elle décide donc de partir à l’aventure chercher de nouvelles amies. Ce périple la mène sur une route arc en ciel ou elle peut collecter des lunes. Mais cette route est dangereuse, elle contient des trous dans lesquels même la terre ne peut s’empêcher de tomber et des obstacles qui attireront le courroux de la route si la terre entre en collision avec eux.

Le joueur contrôle la Terre et doit collecter le plus de lunes et esquiver les pièges.

Le jeu est accessible sur le github suivant : <https://github.com/Jolan-Goulin/Projet-OpenGL-IMAC2>

Une image contenant stationnaire, très coloré

Description générée automatiquement

Mode d’emploi :

Commandes de déplacement :

Q : déplacement a gauche

S : déplacement a gauche

Z : saut

Commandes de caméra :

C : Changement de mode (tps/fps)

L : Blocage de la caméra

Attraper et bouger l’écran avec la souris : rotation de caméra (limité à l’axe vertical en vue fps)

Molette de la souris/flèches haut/bas : Zoom / Dézoom

Commandes de jeu :

E : Lancer/relancer le jeu

P : Mettre le jeu en pause / annuler la pause

A : Sauvegarder le jeu

W : Charger la partie sauvegardée

R : Ressusciter et être temporellement immortel (Pour les pauvres dev / testeurs fatigués)

Architecture logicielle :

Afin de créer notre architecture logicielle nous nous somme basé sur celle vue en TP.

Le dossier Asset contient les fichiers images et texte nécessaires au fonctionnement du projet.

Le dossier third-party contient les librairies glm et stb\_image.

Le dossier Projet contient les shaders utilisés dans le projet ainsi que projet.cpp le fichier principal.

Le dossier glimac contient les fichiers qui nous ont été mis a disposition pour réaliser ce projet ainsi que d’autres fichiers que nous avons créés afin de réaliser ce projet sous version .hpp et .cpp :

GameObjects : Fichier contenant la structure personnage qui contient majorité des informations nécessaires aux jeu

Map : Fichier contenant les fonction nécessaires a l’initialisation, la génération de la map et les colission de la Terre avec les objets

TrackballCamera : Fichier contenant la gestion de la caméra

PersoDraw : Fichier contenant la gestion de l’affichage des objets cubiques

BoxDraw : Fichier contenant la gestion de l’affichage des objets sphériques

Le fichier projet.cpp utilise ces différents éléments pour créer le jeu et gère lui-même des taches simples et la gestion des événements liés aux commandes du joueur.

Gestion de la map :

Nous utilisons afin de créer le parcours du jeu une approche différente de celle proposée dans le sujet. En effet notre parcours est représenté par une grille 1x10 sous la forme d’un tableau de 10 entiers. Ces entiers représentent des types de case (case neutre, case à obstacle, à pièce ou à trou). Quand le joueur avance sur le parcours, il se met a jour en supprimant les éléments dépassés par le joueur et en ajoutant de nouveaux élément générés aléatoirement.

Du fait de la simplicité de la carte initiale et de la génération aléatoire de la suite de celle-ci nous avons choisi de ne pas la stocker dans un fichier.

Une image contenant table

Description générée automatiquement

Design pattern :

Nous avons choisi dans ce projet d’utiliser le design pattern State. La caméra est définie par des états : un état bloqué/libre et un état fps/tps. Ces états influencent toutes les actions de la caméra.

Extensions du projet :

Le parcours est généré aléatoirement au fur et à mesure de la partie et la vitesse augmente lentement.

Il est possible de sauvegarder la partie en appuyant sur a et de la charger en appuyant sur w.

Côté technique :

|  |  |  |
| --- | --- | --- |
| Technique | Utilisé | Présent dans |
| Classes | Oui | BoxDraw.hpp, PersoDraw.hpp, TrackballCamera.hpp, GameObjects.hpp |
| Template | Non |  |
| Polymorphisme | Non |  |
| Encapsulation | Oui | Répartion du code dans les fichiers décrite ci-dessus |
| STL | Oui | Usage de vector dans : Map.cpp, Project.cpp, GameObject.cpp |
| Espace de nommage | Oui | Usage du namespace glimac dans tout les fichiers ajoutés a glimac |
| Exeptions | Oui | PersoDraw.cpp, GameObject.cpp |
| Messages d’erreur | Oui | PersoDraw.cpp, GameObject.cpp |
| Assert | Oui | GameObject.cpp |
| Fonctions lambdas | Non |  |
| héritage | Non |  |
| constexpr | Non |  |
| Doxygen | Non |  |

Gestion du projet :

Nous avons tous deux commencé le projets seul et avons eu un certain nombre de complications qui nous ont conduit à nous réunir afin de former ce groupe de projet

Jolan : J’ai essayé d’utiliser windows pour coder ce projet et ai passé plusieurs jours a essayer de paramétrer mon environnement de travail pour y travailler. Je n’ai pas réussi notamment cause de problèmes avec SDL. J’ai ensuite installé un dual boot linux et paramétré les librairies nécessaires sur ce dernier. Ceci m’a fait perdre un temps considerable.

Quentin:

Suite a cela nous avons décidé de continuer sur la base du projet de Jolan afin de le continuer et avons travaillé en utilisant une méthode agile. Nous avons donc travaillé par objectifs incrémentaux.

Nous avons rencontré des problèmes concernant l’utilisation de SDL\_TTF.