*HMIN317: Moteurs de jeux*

Rendu TP1 &TP2 : Prise en main

Tianning MA

M2 IMAGINA

01/10/2020

Table de matière

[1. Introduction 2](#_Toc52446769)

[2. Fonctionnement des touches 2](#_Toc52446770)

[3. Rendu et explication des exercices 2](#_Toc52446771)

[Question 1 2](#_Toc52446772)

[Question 2 3](#_Toc52446773)

[Question 3 4](#_Toc52446774)

[Question 4 5](#_Toc52446775)

[Question 5 6](#_Toc52446776)

[Question 6 6](#_Toc52446777)

[Question 7 6](#_Toc52446778)

# Introduction

Ce compte rendu est dédié au TP1 et TP2 prise en main de Qt Creator, Git et OpenGL ES 3.0.

Toutes les questions sont répondues.

L’adresse du git est :

# Fonctionnement des touches

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Touche | Fonctionnement |  |  |  |
| UP | Déplacement de caméra vers z+ | LEFT | Déplacement de caméra vers x- | |
| DOWN | Déplacement de caméra vers z- | RIGHT | Déplacement de caméra vers x+ | |
|  |  |  |  | |

# Rendu et explication des exercices

## Question 1

*Expliquer Le fonctionnement les méthodes de dessin et de transformation appliquées aux objets. Quelles sont les mécanismes et fonctions permettant de transmettre à l’application les mises à jour à partir des entrées utilisateur ?*

La méthode *paintGL()* dans la classe MainWidget permet de dessiner les contenus définis(de la classe geometryengine) dans la fenêtre principale. Les transformations se font par la transformation matricielle (dans la méthode paintGL) sur la matrice model en utilisant les fonctions comme *rotate*, *translate* etc.

Le mécanisme utilisé est principalement le mécanisme des signaux et des slots. C’est un principe propre à Qt pour gérer les évènements au sein d’une fenêtre. Un signal est un message envoyé par un widget lorsqu’un évènement se produit, et un slot est la fonction qui va être appelée (ex : la méthode d’une classe). Le signal et le slot se relie grâce à la méthode statique connect().

La macro Q\_OBJECT est nécessaire dans le header de la classe (dans le tp, on en a dans la classe « mainwidget »), qui consiste à demander le compilateur à accepter les slots. Grâce à ce mécanisme, on pourra réaliser les intéractions (clavier ou souris) avec la fenêtre via les slots (méthodes) comme : mousePressEvent(e), keyPressEvent(e) etc.

## Question 2

|  |  |
| --- | --- |
|  |  |
| Les nouvelles méthodes (Dans la classe GeometryEngine):  Void drawPlane(\*program);  Void initPlaneGeometry() ;  Ces deux méthodes permettent de générer la géométrie du plan et le dessiner à l’aide de shaders.  Les triangles sont dessinés dans le plan (z=0), puis j’ai modifié la caméra pour garder la surface visible. Le plan est de taille 16 \* 16, et enfin, j’ai appliqué la texture « grass » dessus. | |

## Question 4

|  |  |
| --- | --- |
|  |  |
| Pour cette question, tout d’abord, j’ai modifié l’altitude (coordonnée z) pour chaque vertex par un chiffre aléatoire. Donc, on a obtenu un relief comme l’image ci-dessus. La caméra est toujours fixée pour garder la surface visible. J’ai aussi réalisé le déplacement de la caméra, vous pouvez le tester. Le déplacement se fait par les touches de direction de clavier. | |

## Question 5

Modification d’altitude par heightmap

Le Heightmap que j’ai utilisé est comme l’image ci-contre :

c’est une image de taille 16 \* 16, rogné de fichier « heightmap – 1024 \* 1024 ». (pour convenir la taille de notre terrain 16 \* 16).

|  |  |
| --- | --- |
|  |  |
|  | |

## Question 6

Proposer un autre mode d’affichage pour regarder le trrain sous un angle de 45 degrés

Et le faire tourner autour de son origine avec une vitesse constante à l’aide d’un timer.

## Question 7