République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Rapport de projet synthèse d'image
« 3D Pong »

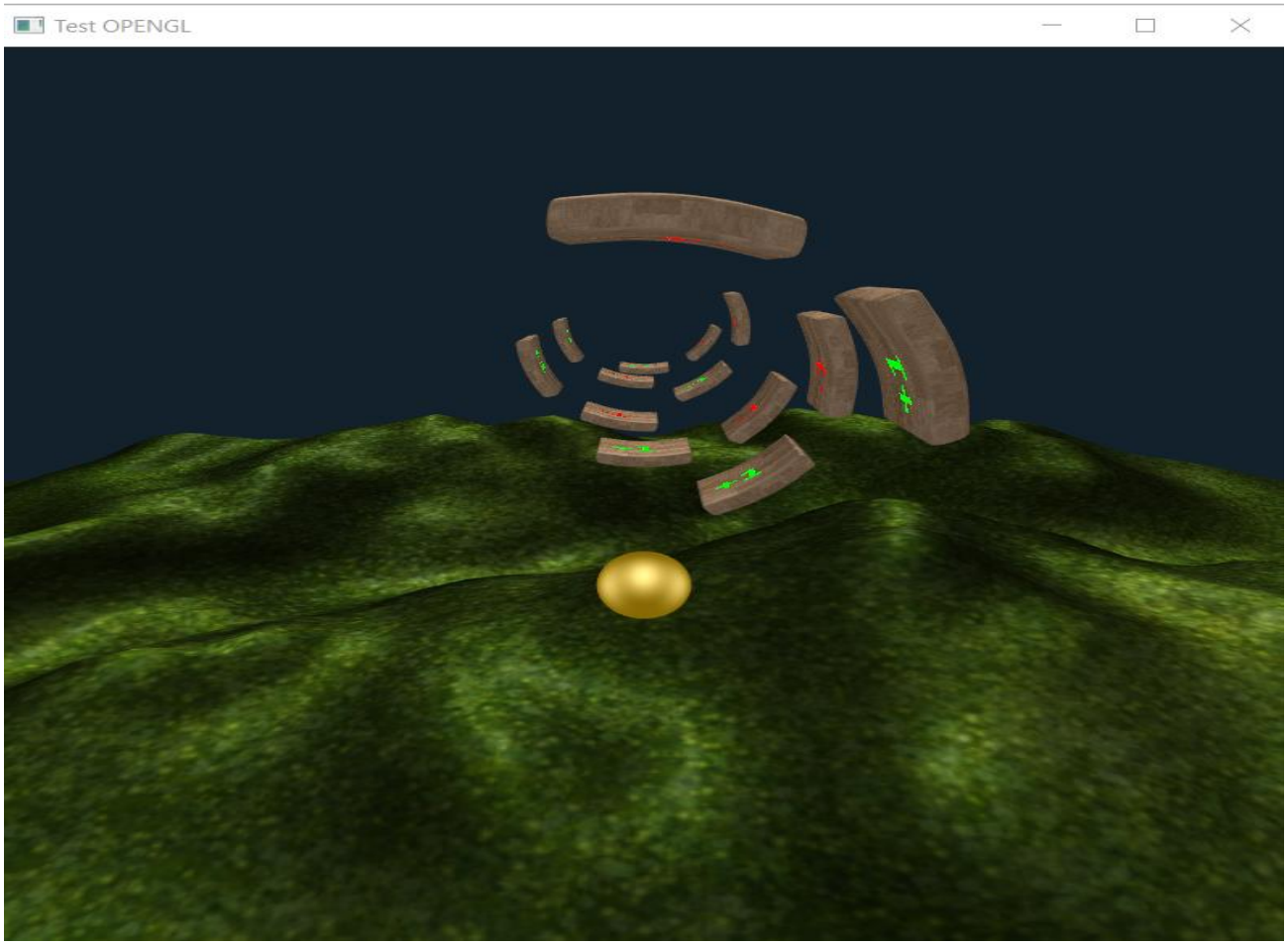
Réalisé par :

DJEKOUNE Imad Eddine 181831089663
SOUIDI Med Amine 181831044438

2022/2023

Introduction

Notre application conçue en OpenGL est en forme d'un jeu, où la scène est un monde ouvert qui contient une balle et des palettes.
Notre joueur représente la balle, il peut se déplacer en sautant sur les palettes et le but c'est d'éviter de tomber, par contre il existe deux types de palettes, celle qui rapporte un score positive et celle qui diminue le score comme illustre la figure suivante :



1. Mécanisme

Le caractère du joueur qui est la balle avance en avant avec le temps, mais il peut manipuler les mouvements à gauche et à droite respectivement avec les touches : 'A', 'D' qui change l'orientation des palettes.

2. Gestion de jeu

Cette structure est divisée en deux parties:
L'une appelée "partie moteur" qui gère tous les graphismes, ce que nous avons étudié, et l'autre appelée "partie jeu" qui utilise les fonctionnalités du moteur pour créer une expérience de jeu jouable.
La partie moteur (ou engine de jeu) contient toutes les fonctionnalités nécessaires pour gérer l'affichage, telles que les shaders, les modèles, la lumière, la caméra, le moteur de rendu et l'affichage.
En ce qui concerne les shaders, il y en a trois :

- BasicShader : il n'utilise pas de lumière et affiche simplement la texture.
- MainShader : il prend en charge 4 types de lumière dans le modèle de Phong (ambiante, diffuse, réfléchie).
- AdvanceShader : similaire au MainShader, mais il prend en charge les textures doubles.

Et pour les models, il contient un mesh, un ou deux textures, et des propriétés de lumière qui sont réflexivité et shineDumpness.

Exemple :

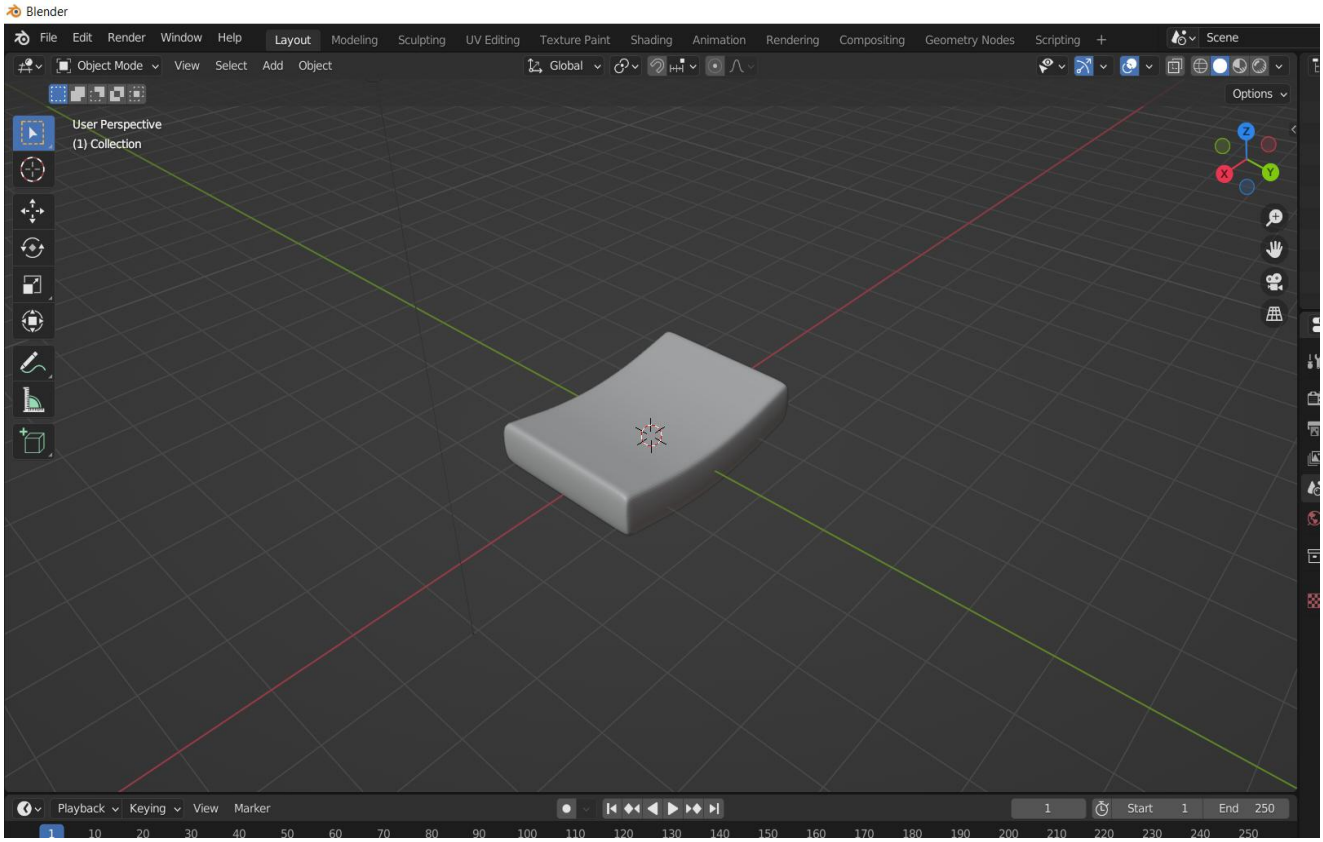
```
GLFWwindow* window = DisplayManager::createDisplay();

ShaderProgram sh = ShaderProgram("shader/vertexShader.txt", "shader/fragmentShader.txt");
ShaderProgram sh2 = ShaderProgram("shader/vertexShaderText.txt", "shader/fragmentShaderText.txt");
ShaderProgram sh3 = ShaderProgram("shader/vertexShaderBasic.txt", "shader/fragmentShaderBasic.txt");


int textures[] = {
    Loader::loadTexture("textures/number.jpg", GL_RGB) ,
    Loader::loadTexture("textures/numberm.jpg", GL_RGB) };



Model shpere = Model(Loader::loadToVao_PTN("models/sphere.obj"), Loader::loadTexture("textures/Metal034_1K_Color.jpg", GL_RGB), Material(3, 1.5));
Model2 palette = Model2(Loader::loadToVao_PTN("models/palette.obj"),
    Loader::loadTexture("textures/WoodFloor051_1K_Color.jpg", GL_RGB),
    textures[0],
    Material(5, 0.1));
Model model = Model(Loader::loadToVao_PTN("models/plan.obj"), Loader::loadTexture("textures/Moss002_1K_Color.jpg", GL_RGB), Material(10, 0.1));
```

3. Modélisation des Objets

En ce qui concerne la modélisation des objets, nous les avons créés à l'aide de Blender, puis enregistrés dans un format .obj. Ensuite, nous avons chargé les modèles dans notre scène en utilisant la méthode de chargement de modèle (vue en TP) avec leur texture, et les avons disposés dans la scène.

4. Structure du jeu

En réalité, la balle ne se déplace pas vers l'avant, mais c'est le monde (qui contient des palettes) qui se déplace vers l'arrière, donnant l'illusion que la balle se déplace vers l'avant.
À chaque instant, nous avons les angles de rotation de chaque palette, sur la base desquels nous calculons les collisions de la balle avec les palettes. Les palettes sont générées à l'infini car elles sont stockées dans un vecteur qui les traque. À chaque collision de la balle avec une palette, une autre palette est générée à la fin. Le score est calculé lors de la collision.
Il y a deux scénarios:
Soit la palette contient un (+1), auquel cas le score est augmenté, ou bien si elle contient un (-1), le score est diminué de 1 et Le score sera affiché à la fin du jeu.


Blender
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Scene
Object Mode View Select Add Object
Global
Options
User Perspective
(1) Collection
Playback Keying View Marker
1
Start 1
End 250