Anggota kelompok :

* Angelo Ricardo Clement - c14210152
* Andreas George - c14210186

Pembagian tugas:

* Andreas George

Membuat model dalam bentuk dan memberikan pada warna pada model tersebut, kemudian saya memberikan lighting di awal awal, membuat video untuk project ini, membuat rencana story, menambahkan environment seperti karakter karakter yang ada seperti mario, princess peach, bowser, bat dan stadium. Menggerakan camera untuk dijadikan film.

* Angelo Ricardo Clement

Saya membuat penempatan dengan menambahkan lamp objek untuk environment, lalu mengatur lighting dan shading, lalu pada lighting juga dibuat animasi sesuai scenario story, membuat jalannya kamera cinematography, juga pada pembuatan kamera FPS dan TPS, membuat animasi homerun mario dan juga collision.

Rencana Story:

Pada suatu hari karena mario sudah terlalu lama menahan perasaan kepada princess peach. Mario akhirnya memutuskan untuk menyatakan perasaanya kepada princess peach. Namun ketika mario ingin menyatakan perasaanya tiba tiba datanglah bowser dan langsung menculik peach. Kemudian mario ingin menghentikannya, lalu bowser menantang mario untuk bermain baseball dan jika mario menang maka peach akan dikembalikan. Dan mario setuju dan mau melakukannya.

Pertandingan dimulai. Saat awal awal bowser mendominasi pertandingan, mario merasa putus asa dan tidak ada lagi harapan bagi mario untuk menang. Namun hati mario mengatakan bahwa dia bisa jika berjuang. Akhirnya, mario memutuskan tidak ingin kalah dan melakukan serangan homerun yang membuat bowser takjub dan memutuskan untuk menyerahkan kemenangan kepada mario. Mario pun pulang dengan bahagia bersama dengan princess peach. Makna dari cerita mario ini adalah apapun yang kita lakukan seringkali tidak sesuai yang kita harapkan namun kita harus terus berjuang dan tidak boleh menyerah karena jika ingin berjuang pasti kita dapat mengalahkan hal yang bagi kita tidak mungkin untuk berhasil.

Cara mengatur lighting sesuai story:

Dengan cara menambahkan lighting pada class Object yang dibuat di createUniformap

public Object(List<ShaderModuleData> shaderModuleDataList

, List<Vector3f> vertices

, Vector4f color) {

super(shaderModuleDataList);

this.vertices = vertices;

// this.size = size;

// setupVAOVBO();

uniformsMap = new UniformsMap(getProgramId());

uniformsMap.createUniform(

"uni\_color");

uniformsMap.createUniform(

"model");

uniformsMap.createUniform(

"projection");

uniformsMap.createUniform(

"view");

uniformsMap.createUniform("dirLight.direction");

uniformsMap.createUniform("dirLight.ambient");

uniformsMap.createUniform("dirLight.diffuse");

uniformsMap.createUniform("dirLight.specular");

//tambah loop untuk tambah light

for(int i = 0; i < 10; i++){

uniformsMap.createUniform("pointLight["+i+"].position");

uniformsMap.createUniform("pointLight["+i+"].ambient");

uniformsMap.createUniform("pointLight["+i+"].diffuse");

uniformsMap.createUniform("pointLight["+i+"].specular");

uniformsMap.createUniform("pointLight["+i+"].constant");

uniformsMap.createUniform("pointLight["+i+"].linear");

uniformsMap.createUniform("pointLight["+i+"].quadratic");

}

uniformsMap.createUniform("spotLight.position");

uniformsMap.createUniform("spotLight.direction");

uniformsMap.createUniform("spotLight.ambient");

uniformsMap.createUniform("spotLight.diffuse");

uniformsMap.createUniform("spotLight.specular");

uniformsMap.createUniform("spotLight.constant");

uniformsMap.createUniform("spotLight.linear");

uniformsMap.createUniform("spotLight.quadratic");

uniformsMap.createUniform("spotLight.cutOff");

uniformsMap.createUniform("spotLight.outerCutOff");

uniformsMap.createUniform("viewPos");

this.color = color;

model = new Matrix4f().identity();

childObject = new ArrayList<>();

centerPoint = new Vector3f(0f,0f,0f);

}

Lalu tambahkan pada scene.frag di NR\_POINT\_LIGHTS untuk menambahkan banyak pointlight yang dibutuhkan

struct PointLight {

vec3 position;

float constant;

float linear;

float quadratic;

vec3 ambient;

vec3 diffuse;

vec3 specular;

};

//tambah light

#define NR\_POINT\_LIGHTS 10

uniform PointLight pointLight[NR\_POINT\_LIGHTS];

Pada drawsetup juga tambahkan point light position dimana untuk mengatur penempatan cahaya yang digunakan dan dibawahnya saya menggunakan point light yang sudah ada untuk diatur mulai dari diffuse, ambient, specular, constant, linear, dan quadratic. Pointlight tersebut juga digunakan dapat mengubah menjadi lampu stadium dan juga dapat mengubah malam hari menjadi pagi hari.

//tambah posisi light

Vector3f[] \_pointLightPositions = {

new Vector3f(0f, 1000f, -20f),

new Vector3f(195f, 70f, -20f),

new Vector3f(-190f, 70f, -13f),

new Vector3f(0f, 65f, 225f),

new Vector3f(-23f, 50f, -180f),

new Vector3f(150f, 70f, -190f),

new Vector3f(-190f, 70f, -150f),

new Vector3f(140f,70f,140f),

new Vector3f(-140f,70f,140f),

new Vector3f(0f, 0f, 30f)

};

for(int i = 0; i < \_pointLightPositions.length; i++){

uniformsMap.setUniform("pointLight["+i+"].position", \_pointLightPositions[i]);

if(scene){

uniformsMap.setUniform("pointLight["+i+"].ambient", new Vector3f(0.01f, 0.01f, 0.01f));

uniformsMap.setUniform("pointLight["+i+"].diffuse", new Vector3f(0.8f, 0.8f, 0.8f));

uniformsMap.setUniform("pointLight["+i+"].specular", new Vector3f(0.05f, 0.05f, 0.05f));

uniformsMap.setUniform("pointLight["+i+"].constant", 1.0f);

uniformsMap.setUniform("pointLight["+i+"].linear", 0.09f);

uniformsMap.setUniform("pointLight["+i+"].quadratic", 0.032f);

} else {

uniformsMap.setUniform("pointLight["+i+"].ambient", new Vector3f(0.4f, 0.4f, 0.4f));

uniformsMap.setUniform("pointLight["+i+"].diffuse", new Vector3f(0.8f, 0.8f, 0.8f));

uniformsMap.setUniform("pointLight["+i+"].specular", new Vector3f(0.5f, 0.5f, 0.5f));

uniformsMap.setUniform("pointLight["+i+"].constant", 1.0f);

uniformsMap.setUniform("pointLight["+i+"].linear", 0.09f);

uniformsMap.setUniform("pointLight["+i+"].quadratic", 0.032f);

}

if (i >= 0 && i < 5){

uniformsMap.setUniform("pointLight[" + i + "].ambient", new Vector3f(0.005f, 0.005f, 0.005f));

uniformsMap.setUniform("pointLight[" + i + "].diffuse", new Vector3f(0.8f, 0.8f, 255 / 255f));

uniformsMap.setUniform("pointLight[" + i + "].specular", new Vector3f(0.5f, 0.5f, 0.5f));

if (mode\_light[0] == 1) {

uniformsMap.setUniform("pointLight[" + i + "].constant", 1.0f);

uniformsMap.setUniform("pointLight[" + i + "].linear", 0.0014f);

uniformsMap.setUniform("pointLight[" + i + "].quadratic", 0.000007f);

}

else {

uniformsMap.setUniform("pointLight[" + i + "].constant", 1.0f);

uniformsMap.setUniform("pointLight[" + i + "].linear", 0.7f);

uniformsMap.setUniform("pointLight[" + i + "].quadratic", 100f);

}

}

if (i >= 5 && i < 9){

uniformsMap.setUniform("pointLight["+i+"].ambient", new Vector3f(0.005f, 0.005f, 0.005f));

uniformsMap.setUniform("pointLight["+i+"].diffuse", new Vector3f(1f, 1f, 102/255f));

uniformsMap.setUniform("pointLight["+i+"].specular", new Vector3f(0.5f, 0.5f, 0.5f));

if (mode\_light[1] == 1) {

uniformsMap.setUniform("pointLight["+i+"].constant", 1.0f);

uniformsMap.setUniform("pointLight["+i+"].linear", 0.007f);

uniformsMap.setUniform("pointLight["+i+"].quadratic", 0.0002f);

}

else {

uniformsMap.setUniform("pointLight[" + i + "].constant", 1.0f);

uniformsMap.setUniform("pointLight[" + i + "].linear", 0.7f);

uniformsMap.setUniform("pointLight[" + i + "].quadratic", 100f);

}

}

if (i == 9){

uniformsMap.setUniform("pointLight["+i+"].ambient", new Vector3f(0.005f, 0.005f, 0.005f));

uniformsMap.setUniform("pointLight["+i+"].diffuse", new Vector3f(1f, 51/255f, 51/255f));

uniformsMap.setUniform("pointLight["+i+"].specular", new Vector3f(0.5f, 0.5f, 0.5f));

if (mode\_light[2] == 1) {

uniformsMap.setUniform("pointLight["+i+"].constant", 1.0f);

uniformsMap.setUniform("pointLight["+i+"].linear", 0.014f);

uniformsMap.setUniform("pointLight["+i+"].quadratic", 0.0007f);

}

else {

uniformsMap.setUniform("pointLight[" + i + "].constant", 1.0f);

uniformsMap.setUniform("pointLight[" + i + "].linear", 0.7f);

uniformsMap.setUniform("pointLight[" + i + "].quadratic", 100f);

}

}

}

Pada main :

//light mode

if (window.isKeyPressed(*GLFW\_KEY\_1*) && !delay){

if(mode\_light[0] == 0){

mode\_light[0] = 1;

} else{

mode\_light[0] = 0;

}

delay = true;

}

if (window.isKeyPressed(*GLFW\_KEY\_2*) && !delay){

if(mode\_light[1] == 0){

mode\_light[1] = 1;

} else{

mode\_light[1] = 0;

}

delay = true;

}

if (window.isKeyPressed(*GLFW\_KEY\_3*) && !delay){

if(mode\_light[2] == 0){

mode\_light[2] = 1;

} else{

mode\_light[2] = 0;

}

delay = true;

}

Keyword yang digunakan untuk lighting :

* Ubah waktu pagi dan malam (L)
* Ubah lampu stadium (1,2,3)

Cara Mengatur Kamera untuk mendukung story :

Set TPS dan FPS digunakan untuk mencari objek yang akan digerakkan dan membuat lingkaran di sekitar objek tersebut untuk membuat kamera tersebut akan mengikuti objek setiap kali objek bergerak.

public void setTPS(){

Vector3f pos = objects.get(7).model.transformPosition(new Vector3f());

ArrayList<Vector3f> vertices = new ArrayList<>(List.*of*());

for(double i=0;i<360;i+=360/360){

float x = (float)(pos.x + 110f\*Math.*sin*(Math.*toRadians*(i)));

float z = (float)(pos.z + 110f\*Math.*cos*(Math.*toRadians*(i)));

vertices.add(new Vector3f(x, pos.y+13f, z));

}

camera.setPosition(vertices.get((int)rotation).x,vertices.get((int)rotation).y, vertices.get((int)rotation).z);

}

public void setFPS(){

Vector3f pos = objects.get(7).model.transformPosition(new Vector3f());

ArrayList<Vector3f> track = new ArrayList<>(List.*of*());

for(double i=0;i<360;i+=360/360){

float x = (float)(pos.x + 75f\*Math.*sin*(Math.*toRadians*(i)));

float z = (float)(pos.z + 75f\*Math.*cos*(Math.*toRadians*(i)));

track.add(new Vector3f(x, pos.y + 9f, z));

}

camera.setPosition(track.get((int)rotation).x, track.get((int)rotation).y, track.get((int)rotation).z);

}

Menggunakan mouse untuk melakukan pergeseran dan membesarkan padangan.

if (window.getMouseInput().isLeftButtonPressed()){

Vector2f displVec = window.getMouseInput().getDisplVec();

camera.addRotation((float) Math.*toRadians*(displVec.x \* 0.1f), (float) Math.*toRadians*(displVec.y \* 0.1f));

}

if (window.getMouseInput().getScroll().y != 0){

projection.setFOV(projection.getFOV() - (window.getMouseInput().getScroll().y \* 0.1f));

window.getMouseInput().setScroll(new Vector2f());

}

Mengatur kamera saja yang bergerak

//camera movement

if (window.isKeyPressed(*GLFW\_KEY\_W*)){

camera.moveForward(move);

}

else if (window.isKeyPressed(*GLFW\_KEY\_S*)){

camera.moveBackwards(move);

}

if (window.isKeyPressed(*GLFW\_KEY\_A*)){

camera.moveLeft(move);

}

else if (window.isKeyPressed(*GLFW\_KEY\_D*)){

camera.moveRight(move);

}

if (window.isKeyPressed(*GLFW\_KEY\_Q*)){

camera.moveUp(move);

}

else if (window.isKeyPressed(*GLFW\_KEY\_E*)){

camera.moveDown(move);

}

Keyword yang digunakan untuk menjalankan kamera :

* Maju (W)
* Mundur (S)
* Kanan (D)
* Kiri (A)
* Atas (Q)
* Bawah (E)

Digunakan untuk mengatur cinematography

//camera muter sendiri

float movee = 1f;

if (window.isKeyPressed(*GLFW\_KEY\_M*)){

pressed = true;

}

if (pressed){

float posx = camera.getPosition().x;

float posy = camera.getPosition().y;

float posz = camera.getPosition().z;

camera.setPosition(-posx, -posy, -posz);

camera.addRotation(0.0f, (float) Math.*toRadians*(movee));

camera.setPosition(posx, posy, posz);

rotation += movee;

if (rotation >= 360.0f){

rotation = 0.0f;

pressed = false;

}

}

//camera muterin object

if (window.isKeyPressed(*GLFW\_KEY\_B*)){

move = 1f;

Vector3f pos = objects.get(10).model.transformPosition(new Vector3f());

Vector3f posCam = camera.getPosition();

ArrayList<Vector3f> vertices = new ArrayList<>(List.*of*());

for(double i=0;i<360;i+=360/360){

float x = (float)(pos.x + 160f\*Math.*sin*(Math.*toRadians*(i)));

float z = (float)(pos.z + 160f\*Math.*cos*(Math.*toRadians*(i)));

vertices.add(new Vector3f(x, pos.y+70f, z));

}

camera.setPosition(vertices.get(0).x, vertices.get(0).y, vertices.get(0).z);

camera.setPosition(-posCam.x, 0, -posCam.z);

camera.addRotation(0.0f, (float) Math.*toRadians*(-move));

camera.setPosition(posCam.x, 0, posCam.z);

rotCamera += move;

if (rotCamera > 359.0f){

rotCamera = 0.0f;

}

camera.setPosition(vertices.get((int)rotCamera).x,vertices.get((int)rotCamera).y, vertices.get((int)rotCamera).z );

}

Keyword yang digunakan :

* Perputaran estetik (B)
* Perputaran kamera sendiri estetik (M)

Pada FPS dan TPS :

// FPS triggering

if (window.isKeyPressed(*GLFW\_KEY\_F1*) && !first\_p){

Vector3f pos = camera.getPosition();

if (!first\_p) {

camera.setPosition(-pos.x, -pos.y, -pos.z);

camera.setRotation(0f, (float) Math.*toRadians*(0));

camera.setPosition(pos.x, pos.y, pos.z);

}

first\_p = true;

third\_p = false;

setFPS();

}

// FPS Camera

if (window.isKeyPressed(*GLFW\_KEY\_LEFT*) && first\_p){

objects.get(7).translateObject(-1f, 0.0f, 0.0f);

setFPS();

}

if (window.isKeyPressed(*GLFW\_KEY\_RIGHT*) && first\_p){

objects.get(7).translateObject(1f, 0.0f, 0.0f);

setFPS();

}

if (window.isKeyPressed(*GLFW\_KEY\_UP*) && first\_p){

objects.get(7).translateObject(0.0f, 0.0f, -1f);

setFPS();

}

if (window.isKeyPressed(*GLFW\_KEY\_DOWN*) && first\_p){

objects.get(7).translateObject(0.0f, 0.0f, 1f);

setFPS();

}

// TPS triggering

if (window.isKeyPressed(*GLFW\_KEY\_F2*) && !third\_p){

Vector3f pos = camera.getPosition();

if (!third\_p) {

camera.setPosition(-pos.x, -pos.y, -pos.z);

camera.setRotation(0f, (float) Math.*toRadians*(0));

camera.setPosition(pos.x, pos.y, pos.z);

}

first\_p = false;

third\_p = true;

setTPS();

}

// TPS Camera

if (window.isKeyPressed(*GLFW\_KEY\_LEFT*) && third\_p){

objects.get(7).translateObject(-1f, 0.0f, 0.0f);

setTPS();

}

if (window.isKeyPressed(*GLFW\_KEY\_RIGHT*) && third\_p){

objects.get(7).translateObject(1f, 0.0f, 0.0f);

setTPS();

}

if (window.isKeyPressed(*GLFW\_KEY\_UP*) && third\_p){

objects.get(7).translateObject(0.0f, 0.0f, -1f);

setTPS();

}

if (window.isKeyPressed(*GLFW\_KEY\_DOWN*) && third\_p){

objects.get(7).translateObject(0.0f, 0.0f, 1f);

setTPS();

}

Keyword yang digunakan :

* Menggunakan FPS (F1)
* Menggunakan TPS (F2)
* Jalan ke depan (Arrorw UP)
* Jalan ke belakang (Arrow Down)
* Jalan ke kanan (Arrow Right)
* Jalan ke kiri (Arrow Left)

Animasi homerun ini digunakan untuk menunjukkan adanya story yang berjalan

//homerun animation

if (window.isKeyPressed(*GLFW\_KEY\_H*)){

press\_homerun = true;

}

if (press\_homerun){

setHomerun();

objects.get(11).translateObject(0f,0f,1f);

move\_homerun += 1f;

if (move\_homerun >=45){

move\_homerun = 0.0f;

press\_mario\_run = true;

}

}

if (press\_mario\_run){

setHomerun();

Vector3f posObj = objects.get(7).model.transformPosition(new Vector3f());

objects.get(7).translateObject(-posObj.x, -posObj.y, -posObj.z);

objects.get(7).rotateObject((float) Math.*toRadians*(1), 0f, 1f, 0f);

objects.get(7).translateObject(posObj.x, posObj.y, posObj.z);

setHomerun();

move\_mario\_run += movee;

if (move\_mario\_run >= 360.0f){

move\_mario\_run = 0.0f;

press\_mario\_run = false;

}

press\_homerun = false;

objects.get(11).translateObject(pos\_ball.x,pos\_ball.y,-pos\_ball.z);

}

Keyword yang digunakan :

* Animasi homerun dijalankan (H)

Cara mengatur Collision dan meningkatkan performa :

Collision digunakan pada animasi homerun dimana ketika bola sampai pada bat pemukul, maka bola akan melayang melewati stadium dan homerun terjadi, lalu objek mario akan berlari memutari lapangan.

Meningkatkan performa tidak ada yang spesial, hanya penggunaan objek dan pergerakan yang minimal pada draw objek.