Nicolas Luciani
7/03/2023
Master 1 IMAGINE Université de Montpellier

HAI819I : Moteurs de jeux
TP 4
Compte-rendu

---

1 Introduction

L'objectif de ces TPs est d'implémenter des premiers mouvements d'objets et des niveau de détails. On commencera aussi à réaliser nos premiers calculs de collisions.

2 Déplacement

La fonction qui déplace un objet en fonction du clavier est donnée comme suit (la variable move étant une variable globale du programme principal dans lequel se trouce cette fonction :

```
if(glfwGetKey(window, GLFW_KEY_K) == GLFW_PRESS) move = glm::vec3(0, 0, 1);
if(glfwGetKey(window, GLFW_KEY_I) == GLFW_PRESS) move = glm::vec3(0, 0, -1);
if(glfwGetKey(window, GLFW_KEY_J) == GLFW_PRESS) move = glm::vec3(-1, 0, 0);
if(glfwGetKey(window, GLFW_KEY_L) == GLFW_PRESS) move = glm::vec3(1, 0, 0);
```

3 Collision

On calcule alors les collisions entre notre objet et le terrain grâce à la fonction suivante. Nous n'avons réussi à appréhender l'utilisation de calculs du triangle intersecté et donc des coordonées barycentriques. Nous avons opté pour une solution qui nous semblaient plus simple, mais dont finalement les résultats ne semblent pas être concluants.

```
// Evaluates collisions between an entity and the terrain
bool computeCollisions(Entity &e, Entity &terrain, const char* pathHeightMap, float offset, float &moveY)
{
    glm::vec3 centerA = e.transform.getLocalPosition();
    int height, width, nbChannels;
    unsigned char *data = stbi_load(pathHeightMap, &width, &height, &nbChannels, 0);
    // Cartesian equation terrain
    Square* s = dynamic_cast<Square*>(terrain.meshes[0]);
    glm::vec3 normal = s->normal();
    glm::vec3 point = s->bottomLeft();
    float a = normal.x; float b = normal.y; float c = normal.z;
    float d = -(a * point.x) - (b * point.y) - (c * point.z);
    // Projection;
    float lambda = (a * centerA.x + b * centerA.y + c * centerA.z + d)/(a*a + b*b + c*c);
```

```
    glm::vec3 projection = glm::vec3(centerA.x - lambda*a, centerA.y - lambda*b, centerA.z - lambda*c);
    float u,v;
    glm::vec3 X = s->bottomRight() - s->bottomLeft();
    glm::vec3 Y = s->upLeft() - s->bottomLeft();
    glm::vec3 p_bottom = projection - s->bottomLeft();
    glm::vec3 projP_X = (glm::dot(p_bottom, X)/glm::length(X) * glm::length(X))*X;
    glm::vec3 projP_Y = (glm::dot(p_bottom, Y)/glm::length(Y) * glm::length(X))*Y;
    u = glm::length(projP_X)/glm::length(X);
    v = glm::length(projP_Y)/glm::length(Y);
    unsigned char *texel = data + (int)(v + width * u) * nbChannels;
    unsigned char heightTexel = texel[0];
    moveY = ((float)(heightTexel)/255.0f + offset) - centerA.y;
    return (centerA.y >= 0 && centerA.y < ((float)(heightTexel)/255.0f + offset));
}
```

Ainsi, dans la boucle principale on a :

```
character.Draw(programID);
float moveY = 0.0f;
bool collides = computeCollisions(character, terrain,
"../data/textures/Heightmap_Mountain.bmp", 3.5, moveY);
move += glm::vec3(0.0f, moveY, 0.0f);
character.transform.setLocalPosition(
character.transform.getLocalPosition() +
(characterSpeed * deltaTime * move));
character.updateSelfAndChild();
move = glm::vec3(0.0f);
```

4 Niveau de détails

On peut charger plusieurs maillages dans nos entités.

```
class LODEntity : public Entity
{
private:
    int current;
public:
    LODEntity(const char* path, unsigned int type) : Entity(path, type), current(0){}
    void nextLevel(){(current+1 >= this->meshes.size())? current = 0 : current ++;}
    void setCurrentLevel(int level){current = level;}
    bool isMaxLevel(){return current == this->meshes.size()-1;}
    void Draw(GLuint shaderID){this->meshes[current]->draw(shaderID);}
};

/* in main */
LODEntity character("../data/models/icosphere.off", 1);
character.addMesh("../data/models/icosphere1.off", 1);
character.addMesh("../data/models/icosphere2.off", 1);

```

```
//...

glm::vec3 camPos = getCamPosition();
if(glm::length(character.transform.getLocalPosition() - camPos) < 10)
character.setCurrentLevel(0);
else if(glm::length(character.transform.getLocalPosition() - camPos) < 15)
character.setCurrentLevel(1);
else character.setCurrentLevel(2);
```

On obtient alors les résultats suivants.

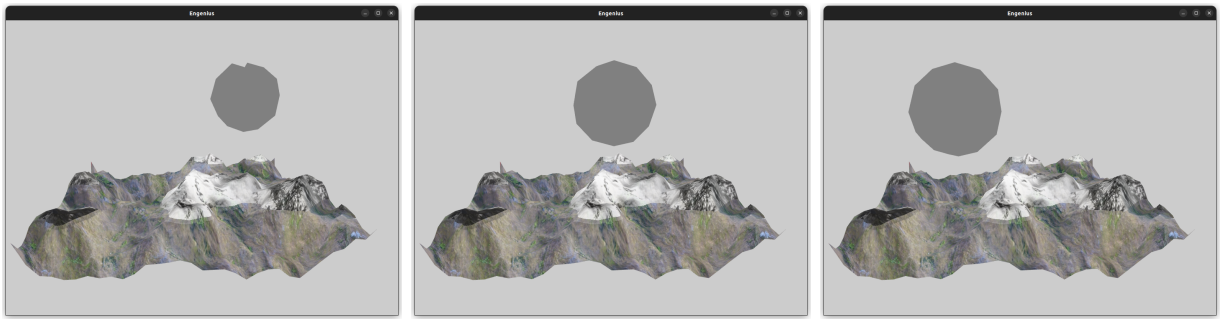

Figure 1: Déplacement de l'objet en fonction de la hauteur du terrain

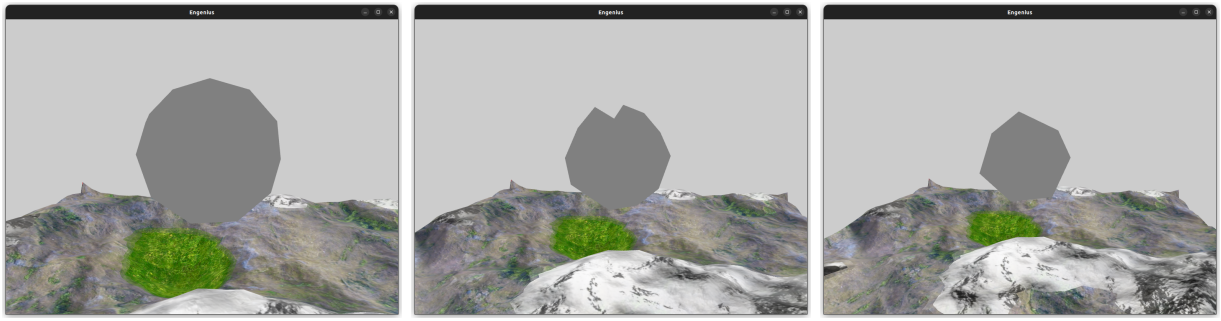

Figure 2: Niveaux de détails