# Moteur de jeux

**Compte-rendu TP4**

**Mouvement**

Louis Jean
Master 1 IMAGINE
Université de Montpellier
N° étudiant : 21914083

28 mars 2024

## Table des matières

# 1 Introduction

Dans ce TP, facultatif, nous nous sommes intéressés au déplacement d'un objet à hauteur fixe au dessus du sol.

# 2 Déplacement d'un objet à hauteur fixe du sol

## 2.1 Créer un objet et l'afficher sur la scène

Grâce aux TP précédents, j'ai pu créer une sphère grâce à la classe `Sphere` et l'afficher dans ma scène.

Pour le terrain, j'ai créé une classe `Plane` qui génère les sommets, les triangles, les UVs et les normales de manière explicite, qui hérite de ma classe `Mesh`. On peut contrôler le "bruit" du terrain (ses variations de hauteur) via son paramètre `height_scale`.

```
void Plane::setup_plane() {
    std::srand(static_cast<unsigned int>(std::time(nullptr)));
    // Vertices data
    for (unsigned int z = 0; z <= grid_z; ++z) {
        for (unsigned int x = 0; x <= grid_x; ++x) {
            Vertex vertex;
            float real_x = this->x + x * (size / grid_x);
            float real_z = this->z + z * (size / grid_z);
            vertex.position = glm::vec3(real_x, (std::rand() % 100 / 100.0f)
    * height_scale, real_z);
            vertex.normal = glm::vec3(0, 1, 0); // Normale vers le haut
            vertex.uv = glm::vec2(x / (float)grid_x, z / (float)grid_z);
            this->vertices.push_back(vertex);
        }
    }
    // Indices
    for (unsigned int z = 0; z < grid_z; ++z) {
        for (unsigned int x = 0; x < grid_x; ++x) {
            this->indices.push_back((z    ) * (grid_x + 1) + x);
            this->indices.push_back((z + 1) * (grid_x + 1) + x);
            this->indices.push_back((z + 1) * (grid_x + 1) + x + 1);

            this->indices.push_back((z    ) * (grid_x + 1) + x);
            this->indices.push_back((z + 1) * (grid_x + 1) + x + 1);
            this->indices.push_back((z    ) * (grid_x + 1) + x + 1);
        }
    }
    this->setup_mesh();
}
```

Voici ce que l'on peut observer comme rendu après la première étape effectuée.

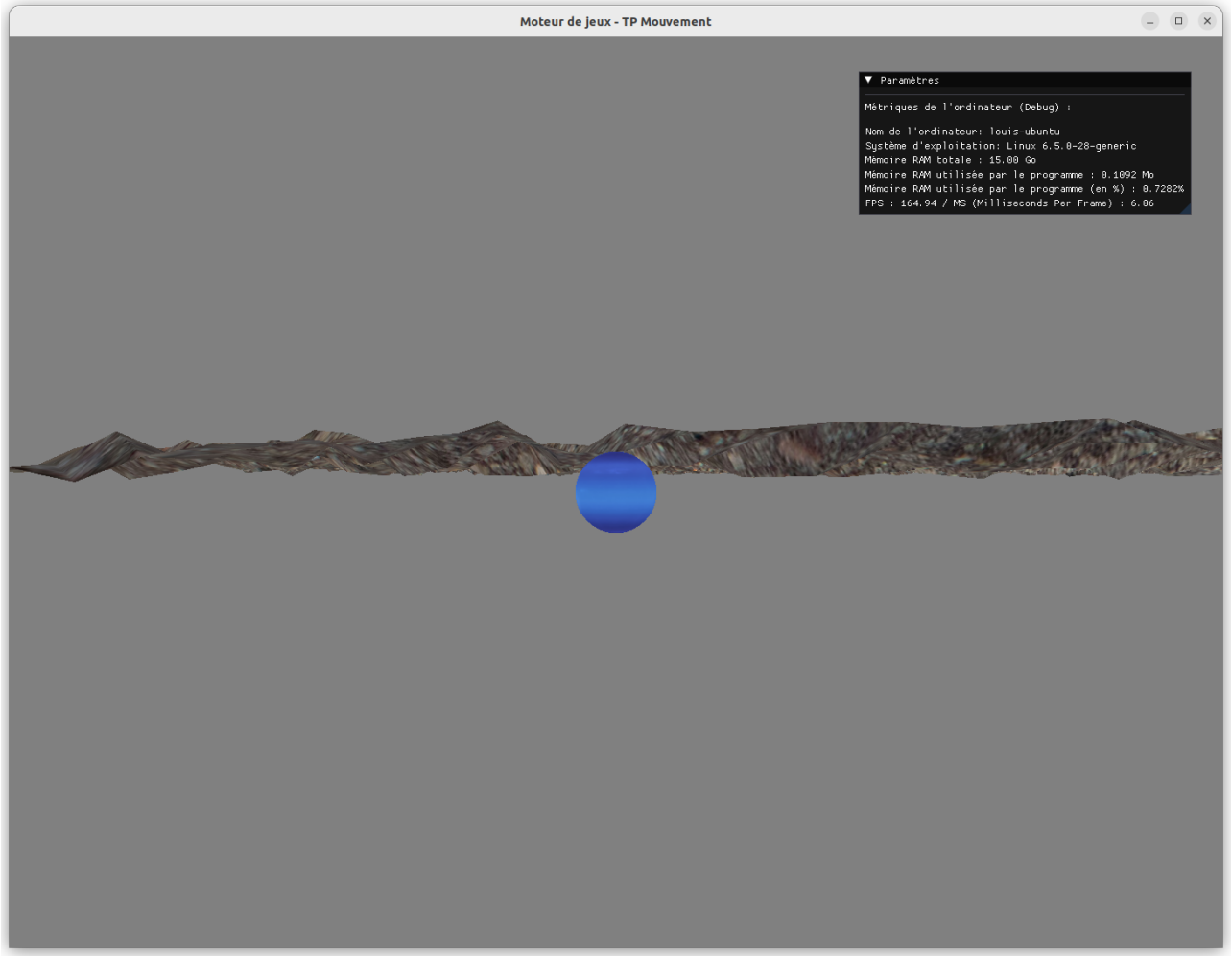



Figure 1: Sphère et terrain affichés dans la scène

Pour l'instant, la sphère ne suit absolument pas la hauteur du plan et passe à travers ce dernier.

## 2.2 Identification de la hauteur du terrain en un point donné

Afin de toujours savoir à quelle hauteur on se trouve lorsque l'on est sur le plan, j'ai écrit une fonction `get_height_at_position`. Cette fonction prend en paramètres une position en $x$ et une position en $y$, puis retrouve les coordonnées dans le plan de ces positions. Cela fonctionne car j'ai créé mon plan comme une grille, je peux donc accéder à chaque cellule avec des calculs simples. Une fois la case dans laquelle l'objet se trouve est identifiée, on récupère les hauteurs des 4 sommets qui composent cette cellule. Ensuite, on interpole ces 4 hauteurs pour approximer la hauteur actuelle à l'endroit où se trouve l'objet.

```
1  float Plane::get_height_at_position(float x, float z) const {
2      float terrain_x = x - this->x;
3      float terrain_z = z - this->z;
4      float grid_square_size_x = this->size / (float)(grid_x);
5      float grid_square_size_z = this->size / (float)(grid_z);
6      int pos_grid_x = (int)(terrain_x / (grid_square_size_x));
7      int pos_grid_z = (int)(terrain_z / (grid_square_size_z));
8      std::cout<<"pos_grid_x "<<pos_grid_x<<"   "<<"grid_x "<<grid_x<<std::
       endl;
9      std::cout<<"pos_grid_z "<<pos_grid_z<<"   "<<"grid_z "<<grid_z<<std::
       endl;
10     if(pos_grid_x > grid_x - 1|| pos_grid_z > grid_z - 1 || pos_grid_x < 0
       || pos_grid_z < 0) {
11         return 0;
12     }
13     float x_inside_square = fmod(terrain_x, grid_square_size_x) /
       grid_square_size_x;
14     float z_inside_square = fmod(terrain_z, grid_square_size_z) /
       grid_square_size_z;
15     float height00 = this->vertices[pos_grid_z * (grid_x + 1) + pos_grid_x].
       position.y;
16     float height10 = this->vertices[pos_grid_z * (grid_x + 1) + pos_grid_x +
        1].position.y;
17     float height01 = this->vertices[(pos_grid_z + 1) * (grid_x + 1) +
       pos_grid_x].position.y;
18     float height11 = this->vertices[(pos_grid_z + 1) * (grid_x + 1) +
       pos_grid_x + 1].position.y;
19     float interpolated_height = height00 * (1 - x_inside_square) * (1 -
       z_inside_square) + height10 * x_inside_square * (1 - z_inside_square) +
       height01 * (1 - x_inside_square) * z_inside_square + height11 *
       x_inside_square * z_inside_square;
20     return interpolated_height;
21 }
```

## 2.3 Mise à jour de la hauteur de l'objet

Pour constamment rechercher la collision et affecter la nouvelle hauteur à l'objet, j'ai créé une classe `PlaneCollider`, qui joue le rôle d'interface entre l'objet et le plan. Cette classe prend une référence à un objet `Plane` en attribut.

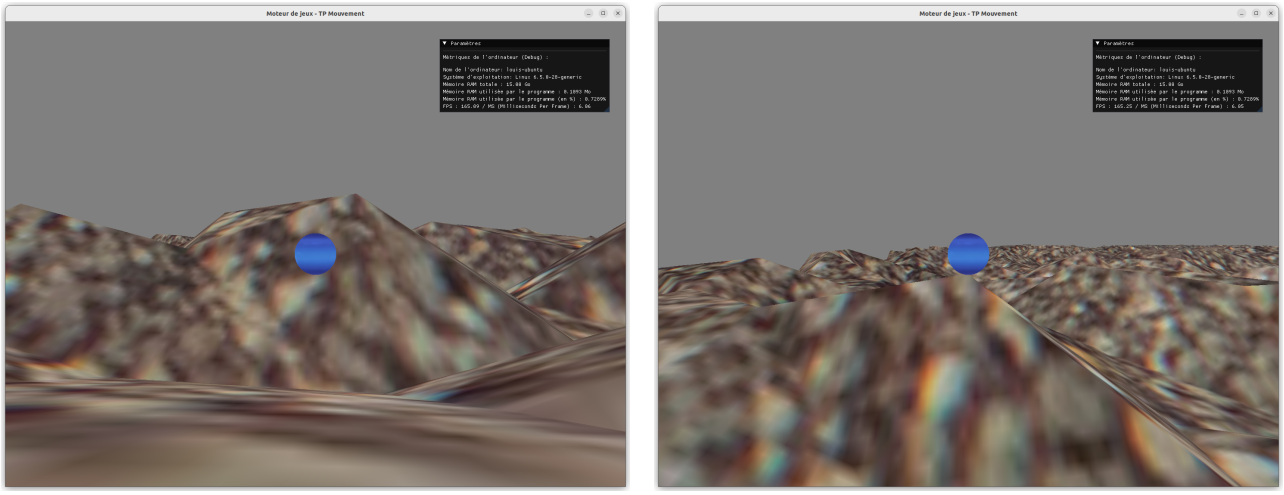
J'ai donc écrit, toujours dans cette classe, une méthode pour vérifier la collision avec un objet (en l'occurrence, une sphère).

```
1 void PlaneCollider::check_collision_with_sphere(float sphere_radius, glm::
    vec3 &sphere_position) {
2     float current_height = plane.get_height_at_position(sphere_position.x,
    sphere_position.z);
3     sphere_position.y = current_height + sphere_radius;
4 }
```

Cette fonction récupère la hauteur du plan, puis l'affecte comme nouvelle coordonnée $y$ de la sphère, en y rajoutant le rayon de la sphère, afin que ce soit bien la base de cette dernière qui soit toujours en contact avec le plan.

Après implémentation nécessaire au fonctionnement de toutes ces méthodes dans le `main` et ajout d'une caméra qui suit la sphère, on peut observer que la sphère se déplace toujours en suivant la hauteur du plan.

**N.B :** Pour déplacer la sphère, veuillez utiliser les touches `Z,Q,S,D`.



(a) Sphère sur le flanc d'un sommet (b) Sphère sur le pic d'un sommet

Figure 2: Sphère se déplaçant sur le plan en suivant sa hauteur

# 3 Conclusion

Dans ce court TP, nous avons implémenté le déplacement d'une sphère sur un plan de hauteur quelconque. Nul doute que je me servirai de cette fonctionnalité dans le prochain TP. Je n'ai malheureusement pas eu le temps d'implémenter le LOD demandé dans ce TP, je le regrette.

Merci pour le temps et l'attention que vous avez consacrés à la lecture de ce compte-rendu.