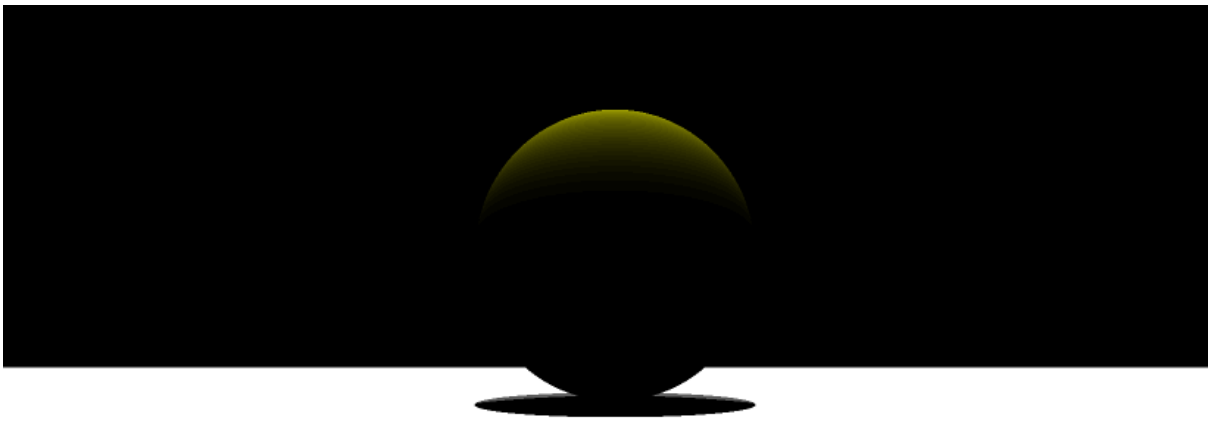
Raytracer

Documentation utilisateur



1. **Réalisé par :**
    - Axel Fradet axel.fradet@epitech.eu
    - Mathieu Robert mathieu1.robert@epitech.eu
    - Kylian Tranchet kylian.tranchet@epitech.eu
    - Théophile Jérôme-Rocher theophile.jerome-rocher@epitech.eu
        - **Promotion Epitech Nantes PGE 2027. 2e année.**

2. **Liens utiles (utilisateur):**
   - Github: https://github.com/Njord201/Raytracer

3. **Le projet:**
   Le projet Raytracer est réalisé par groupe de 4 dans le cadre de la 2e année à Epitech. L'objectif est de faire un logiciel de raytracing, donc créer une image réaliste basée sur une scène fournie par l'utilisateur, pouvant contenir des formes, des jeux de lumière... Le programme lorsqu'exécuté prend en paramètre le lien vers un fichier .cfg (configuration d'une scène) et l'affiche directement sur l'écran.

4. **Supporté par nôtre Raytracer:**
   - **Primitives**
     - Sphère
     - Cylindre infini
     - Plan
     - Cube
     - Cône infini
     - Triangle
     - .OBJ fichiers
       - Matériaux des primitives (flatColor : couleur plate)
   - **Caméra**
     - Résolution
     - Position
     - Rotation
     - FOV (Champ de Vue)
   - **Lumières**
     - Lumière ambiante
     - Lumière diffuse
     - Points lumineux
     - Lumières directionnelles
     - Ombres
   - **Transformations:**
     - Rotation
     - Translation
   - **Scènes**:
     - Chargement d'une scène depuis une scène

5. **Dépendances:**

Veillez à bien répondre à ces dépendances sur votre système afin que le Raytracer fonctionne correctement.

- Make
- G++ (Compilateur C++)
- SDL2
- Git (Avez utilisateur connecté)

6. **Installation + Exécution du Raytracer:**
   - Ouvrez votre terminal
   - Tapez “**git clone** https://github.com/Njord201/Raytracer.git”
   - Tapez “**cd Raytracer/**”
   - Pour compiler proprement le Raytracer tapez “**make re**”
   - Pour l’exécuter, tapez “**./Raytracer {chemin_fichier.cfg}**” en mettant un fichier de configuration valide. Ou **–help** pour l’aide.

7. **Les scènes (configuration):**
    I. **Préambule:**

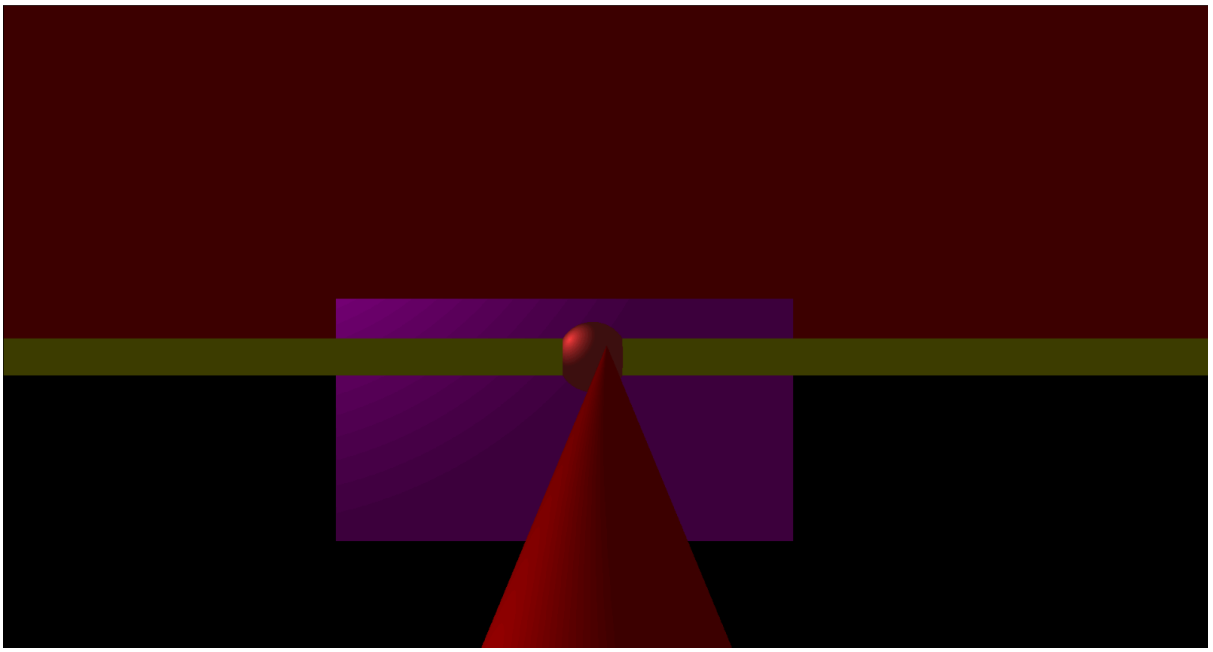
Lors de l'exécution du Raytracer vous fournissez le lien vers un fichier.cfg. Plusieurs exemples sont disponibles dans **tests/files_samples/**.

Vos fichiers de configuration, en anglais, doivent suivre le format de fichier Libconfig, comme celui-ci:

```
camera :
{
    resolution = { width = 1920; height = 1080; };
    position = { x = 0.0; y = -0.0; z = 0.0; };
    rotation = { x = 0.0; y = 0.0; z = 0.0; };
    fieldOfView = 72.0; # In degree
};
# Primitives in the scene
primitives :
{
    # List of spheres
    spheres = (
    { x = 6; y = -5; z = 200; r = 15; material = { type = "flatColor"; color = { r = 255; g = 64; b = 64;};};},
    ) ;

    # List of cylinders
    cylinders = (
    { x = 6; y = -5; z = 200; r = 10; axis="X"; material = { type = "flatColor"; color = { r = 255; g = 255; b = 0;};} },
    );

    #list of cones
    cones = (
        { x = 0; y = 0; z = 100; angle = 45; axis="Y"; material = { type = "flatColor"; color = { r = 255; g = 0; b = 0;};} }
    );

    #list of planes
    planes = (
        { position = 500; axis="Y"; rotation = { x = 0.0; y = 0.0; z = 0.0; }; material = { type = "flatColor"; color = { r = 255; g
= 0; b = 0;};} }
    );

    # List of rectangulars cuboids
    rectangular_cuboids = (
    { minX = -200; minY = -105; minZ = 250; maxX = 45; maxY = 25; maxZ = 260; material = { type = "flatColor"; color = { r = 255; g =
0; b = 255;};}; translation = {x = 100, y = 0, z = 0};},
    );
};
# Light configuration
lights :
{
    ambient = 0.5; # Multiplier of ambient light
    diffuse = 0.6; # Multiplier of diffuse light
    # List of point lights
    point = (
        {x = 200; y = 150; z = 0;},
    );
    # List of directional lights
    directional = (
        {position = {x = 0; y = 500; z = 10;};  direction = {x = 0; y = 0; z = 0;} }
    );
};
imports : {
    scenes = (
        {path = "tests/files_examples/subjects/subject2.cfg"},
    );
};
```

Comme vous pouvez le voir, nous pouvons configurer la caméra, les primitives, les lumières. Les trois sont des impératifs. Les imports sont facultatifs.

Résultat avec notre Raytracer le 10/05/2024 pour cette scène :



II. **Configuration de la caméra:**

```
camera :
{
    resolution = { width = 1920; height = 1080; };
    position = { x = 0.0; y = -0.0; z = 0.0; };
    rotation = { x = 0.0; y = 0.0; z = 0.0; };
    fieldOfView = 72.0; # In degree
};
```

Caméra **“camera”** prend:

- **Résolution**
  - **“width”** (largeur)
  - **“height”** (hauteur)
- **Position**
  - **“x” “y” “z”**
- **Rotation**
  - **“x” “y” “z”**
- Le champ de vision “**fieldOfView**” **prend des degrés.**

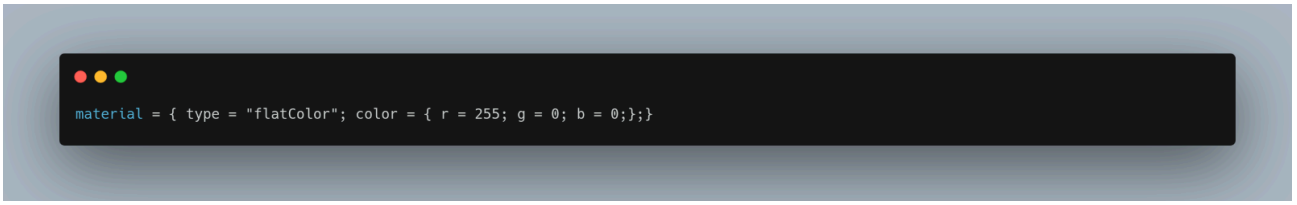
III. Configuration des primitives:

Pour chaque primitives (cylindres, sphères...), vous pouvez en spécifier plusieurs.

```
primitives :
{
    #ici les primitives
};
```

a. Couleur/Matériaux des primitives

Nous supportons la **flatColor**, ou couleur plate. Vous pourrez retrouver ce matériau dans les configurations suivantes.



```
material = { type = "flatColor"; color = { r = 255; g = 0; b = 0;};}
```

Un **material** prend en paramètres:

- **Type** (flatColor disponible).
- **Color,** prenant elle-même **“r” “g” “b”** pour couleur rgb de la flatColor.

b. Sphères

```
spheres = (
    { x = 6; y = -5; z = 200; r = 15; material = { type = "flatColor"; color = { r = 255; g = 64; b = 64;};};},
);
```

Une **sphère** prend en paramètres:

- **Origine “x” “y” “z”**.
- **Radius “r”**.
- **Matériau “material”** (Cf. 7.III.a)**.**

c. **Cylindres**

Un **cylindre** prend en paramètres:

- **Origine “x” “y” “z”**.
- **Radius “r”**.
- **Axis “axis”**.
- **Matériau “material”** (Cf. 7.III.a)**.**

d. **Cônes infinis**

Un **cône** prend en paramètres:

- **Origine “x” “y” “z”**.
- **Angle “angle”** en degrés.
- **Axis “axis”** soit “X” “Y” ou “Z”.
- **Matériau “material”** (Cf. 7.III.a)**.**

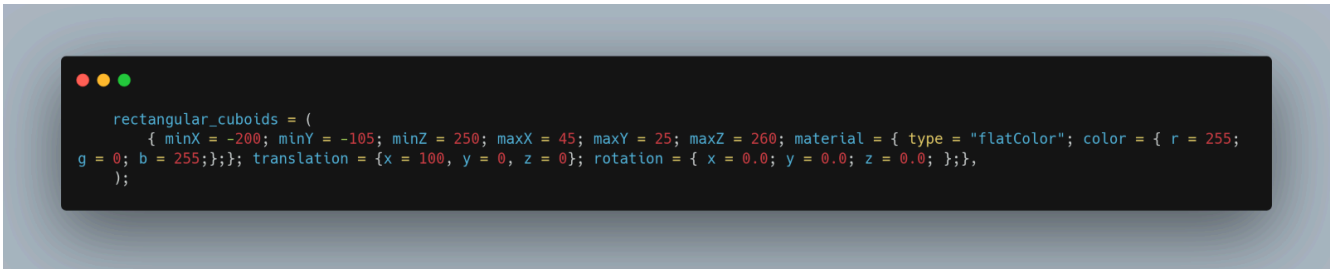
e. **Plans**

Un plan prend en paramètres:

- **Position “position”**.

- **Axis “axis”** soit “X” “Y” ou “Z”.
- **Matériau “material”** (Cf. 7.III.a)**.**

f. Cubes

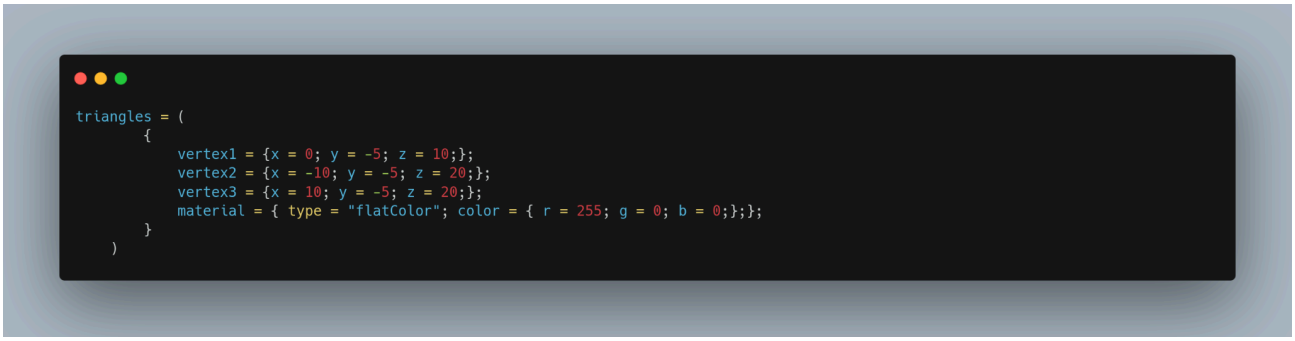


```
rectangular_cuboids = (
    { minX = -200; minY = -105; minZ = 250; maxX = 45; maxY = 25; maxZ = 260; material = { type = "flatColor"; color = { r = 255;
g = 0; b = 255;};}; translation = {x = 100, y = 0, z = 0}; rotation = { x = 0.0; y = 0.0; z = 0.0; };},
);
```

Un cube prend en paramètres:

- **Coordonnées de ses points : “minX” “minY” “minZ” “maxX” “maxY” “maxZ”**.
- **Matériau “material”** (Cf. 7.III.a)**.**
    - *Ici un exemple de translation + rotation*.

g. Triangles

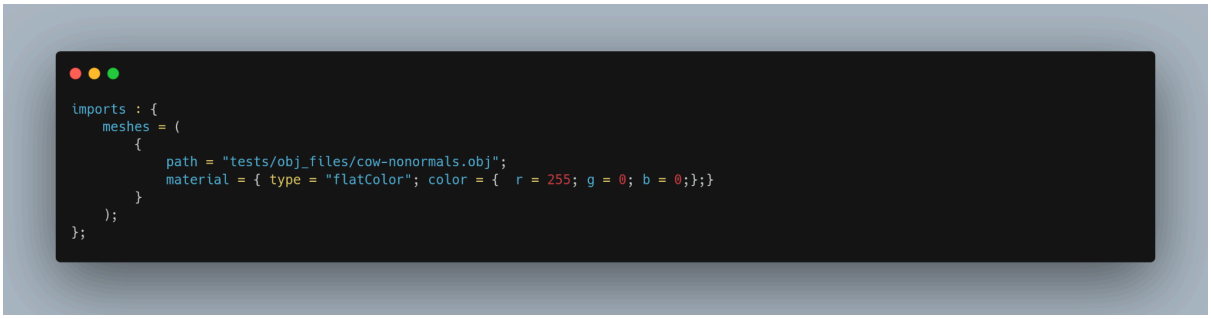


```
triangles = (
    {
        vertex1 = {x = 0; y = -5; z = 10;};
        vertex2 = {x = -10; y = -5; z = 20;};
        vertex3 = {x = 10; y = -5; z = 20;};
        material = { type = "flatColor"; color = { r = 255; g = 0; b = 0;};};
    }
)
```

Un triangle prend en paramètres:

- **“vertex1”** comprenant **“x” “y” “z”** les coordonnées d’un angle.
- **“vertex2”** comprenant **“x” “y” “z”** les coordonnées d’un angle.
- **“vertex3”** comprenant **“x” “y” “z”** les coordonnées d’un angle.
- **Matériau “material”** (Cf. 7.III.a)**.**

h. Fichiers .OBJ



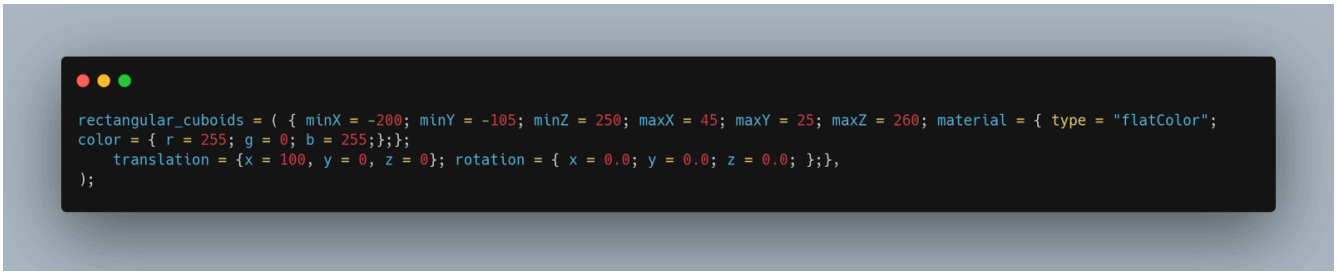
```
imports : {
    meshes = (
        {
            path = "tests/obj_files/cow-nonormals.obj";
            material = { type = "flatColor"; color = {  r = 255; g = 0; b = 0;};}
        }
    );
};
```

Vous pouvez charger un fichier .obj, contenant donc des formes complexes (vache…). Nous ne supportons que les .obj contenant des faces (f) et des vertex (v). L'import d'un .obj se fait dans **"meshes"** dans **"imports".** Chaque mesh prend en paramètres:

- Le lien vers le fichier **"path".**
- Un matériau **"material"** (Cf. 7.III.a)**.**

i. Configuration transformations primitives

Vous pouvez effectivement bouger vos primitives dans l'espace.

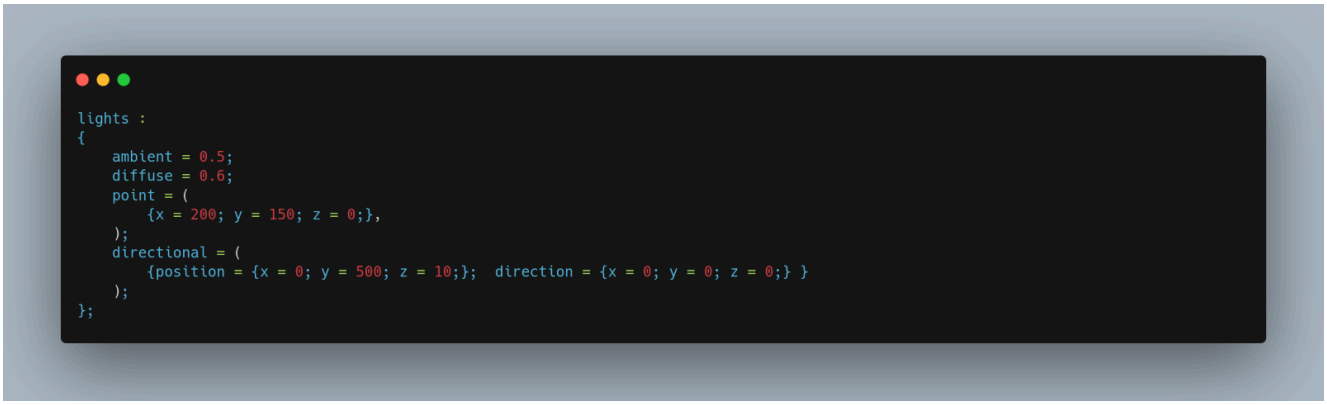


```
rectangular_cuboids = ( { minX = -200; minY = -105; minZ = 250; maxX = 45; maxY = 25; maxZ = 260; material = { type = "flatColor";
color = { r = 255; g = 0; b = 255;};};
    translation = {x = 100, y = 0, z = 0}; rotation = { x = 0.0; y = 0.0; z = 0.0; };},
);
```

Disponibles : **translation** et **rotation**.

- **Rotation** prend **"x" "y" "z"**.
- **Translation** prend **"x" "y" "z"**.

IV. Configuration des lumières:



```
lights :
{
    ambient = 0.5;
    diffuse = 0.6;
    point = (
        {x = 200; y = 150; z = 0;},
    );
    directional = (
        {position = {x = 0; y = 500; z = 10;};  direction = {x = 0; y = 0; z = 0;} }
    );
};
```

Les lumières “**lights”** prennent en paramètres:

- La lumière ambiante, “**ambient**”, un multiplicateur.
- La diffusion de la lumière, “**diffuse”,** un multiplicateur.
- **“point”** soit les point lumineux
    - **Un point prend coordonnées “x” “y” “z”**
- **“directional”** soit les lumières directionnelles
    - Une lumière directionnelle prend en paramètre:
        - Coordonnées **“x” “y” “z”**
        - Direction **“x” “y” “z”**

V. Configuration des imports:

Cette configuration “**imports**” est facultative. Elle permet d’importer une autre scène, un fichier .cfg depuis une scène. Seront importés les paramètres **imports, primitives, lights.**

```
imports : {
    scenes = (
        {path = "tests/files_examples/subjects/subject2.cfg"},
    );
};
```

Imports peut prendre en paramètres des scènes **“scenes“,** possédant chacune un lien ”**path”** vers la configuration.

Sont bien gérées les potentielles boucles infinies, si un fichier s’importe lui-même le programme le détectera.