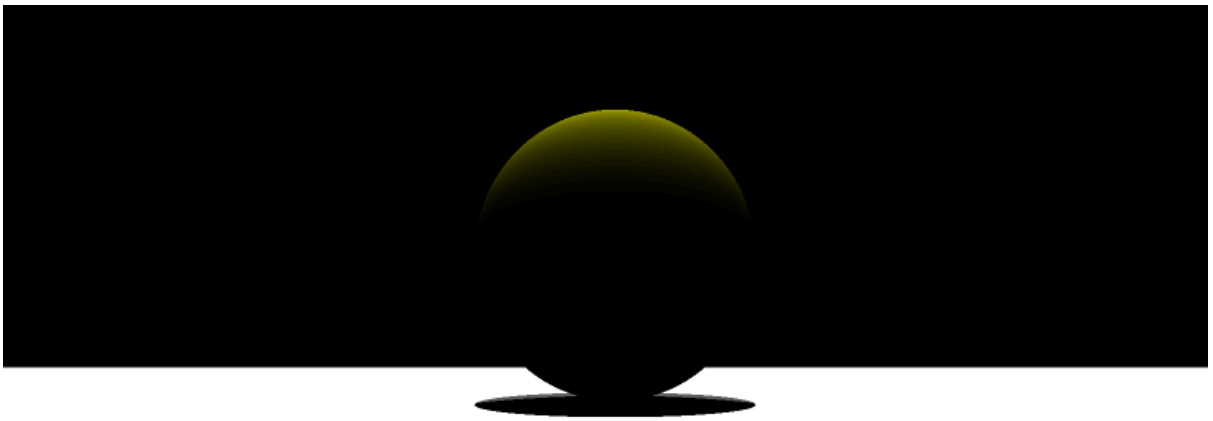
Raytracer

User documentation



1. **Made by:**
   - 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 2027. 2nd year.**

2. **Useful links (users):**
   - Github: https://github.com/Njord201/Raytracer

3. **The project:**
   The Raytracer project is realized in groups of 4 as part of the 2nd year at Epitech. The aim is to create a raytracing software program, creating a realistic image based on a scene provided by the user, which can contain shapes, light effects... When executed, the program takes the link to a .cfg file (scene configuration) and displays it directly on the screen.

4. **Supported by our Raytracer:**
   - **Primitives**
     - Sphere
     - Infinite cylinder
     - Plane
     - Cube
     - Infinite cone
     - Triangle
     - .OBJ files
       - Primitives Materials (flatColor…)
   - **Camera**
     - Resolution
     - Position
     - Rotation
     - FOV (Field Of View)
   - **Lights**
     - Ambient lights
     - Light diffuse multiplier
     - Lights points
     - Directional lights
     - Shadows
   - **Transformations:**
     - Rotation
     - Translation
   - **Scenes**:
     - Load a scene from another (.cfg load from .cfg)

5. **Dependencies:**

Make sure you meet these dependencies on your system so that Raytracer works properly.

- Make
- G++ (C++ compiler)
- SDL2
- Git (Correctly set up)

6. **Install + run Raytracer**
   - Open terminal
   - Type “**git clone** https://github.com/Njord201/Raytracer.git”
   - Type “**cd Raytracer/**”
   - To compile the raytracer type **“make”** or “**make re**”
   - To run, type “**./Raytracer {chemin_fichier.cfg}**” with a valid configuration file. Or –help for help.

7. **Configure scenes:**
    I. **Introduction:**

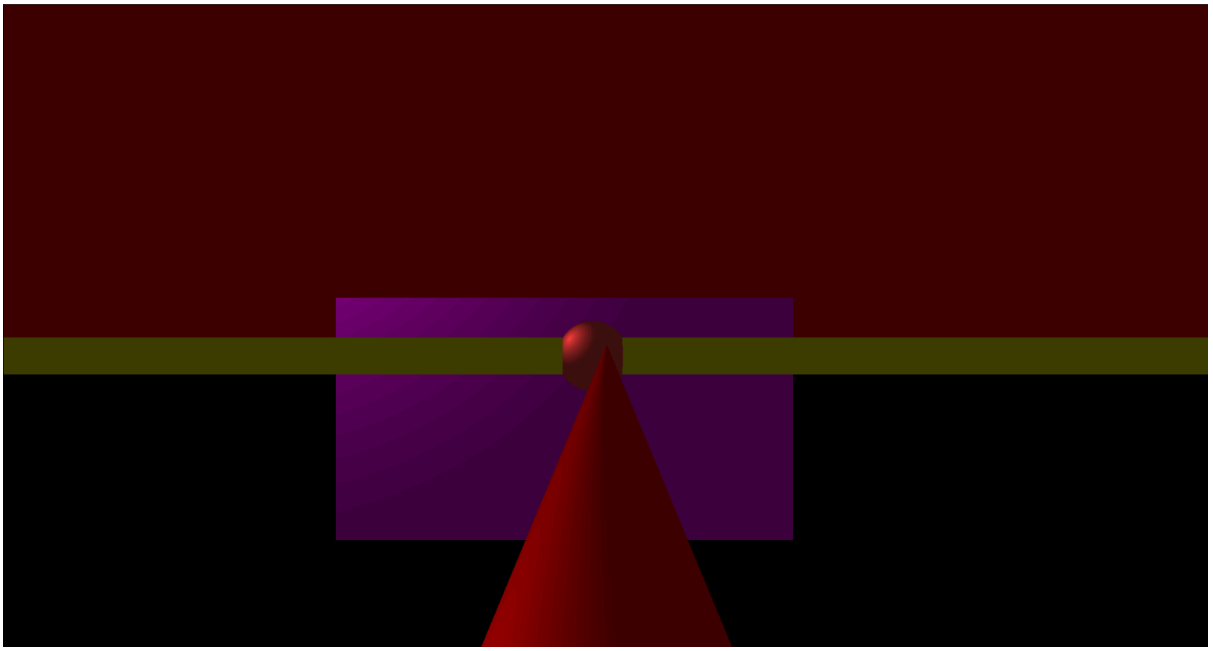
When you run Raytracer, you provide a path to a .cfg file. Several examples are available in tests/files_samples/.

Your configuration files should follow the Libconfig file format, like this one:

```
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"},
    );
};
```

As you can see, we can configure the camera, the primitives and the lights. All three are mandatory. Imports are optional.

Result with our Raytracer on 10/05/2024 for this scene:



II. **Camera configuration:**

```
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
};
```

Camera **“camera”** takes:

- **Resolution**
  - **“width”**
  - **“height”**
- **Position**
  - **“x” “y” “z"**
- **Rotation**
  - **“x” “y” “z”**
- **“fieldOfView” takes degrees.**

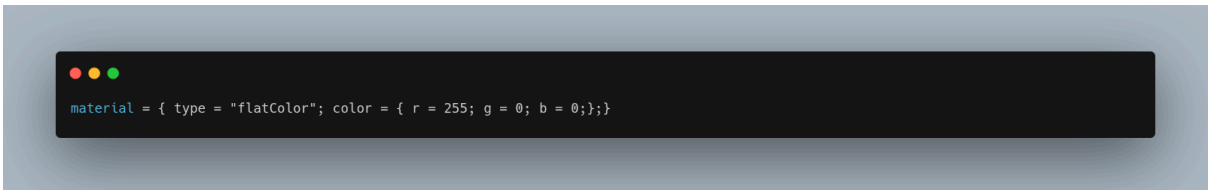
III. Primitives configuration:

For each primitive (cylinder, sphere...), you can specify several.

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

a. Color/Material primitives

We currently support : flatColor. You will find this material used for all primitives right now in the following samples.



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

A **material** needs the following parameters:

- **Type** (flatColor available).
- **Color,** taking as parameter **“r” “g” “b”** for the flatColor rgb.

b. Spheres

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

A **sphere** needs the following parameters:

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

c. **Infinite Cylinders**

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

A **cylinder** needs the following parameters:

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

d. **Infinite cones**

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

An **infinite cone** needs the following parameters:

- **Origin “x” “y” “z”**.
- **Angle “angle”** in degrees.
- **Axis “axis”** : “X” “Y” ou “Z”.
- **Material “material”** (Cf. 7.III.a)**.**

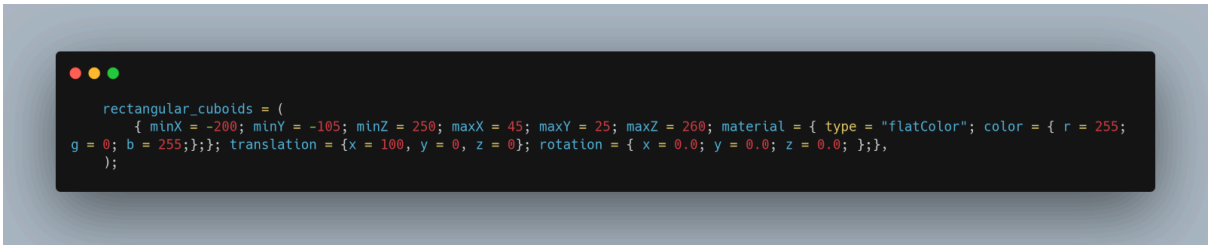
e. **Planes**

```
planes = (
    { position = 500; axis="Y"; material = { type = "flatColor"; color = { r = 255; g = 0; b = 0;};} }
);
```

A **plane** needs the following parameters:

- **Position “position”**.
- **Axis “axis”** : X Y ou Z.
- **Material “material”** (Cf. 7.III.a)**.**

f. Cubes

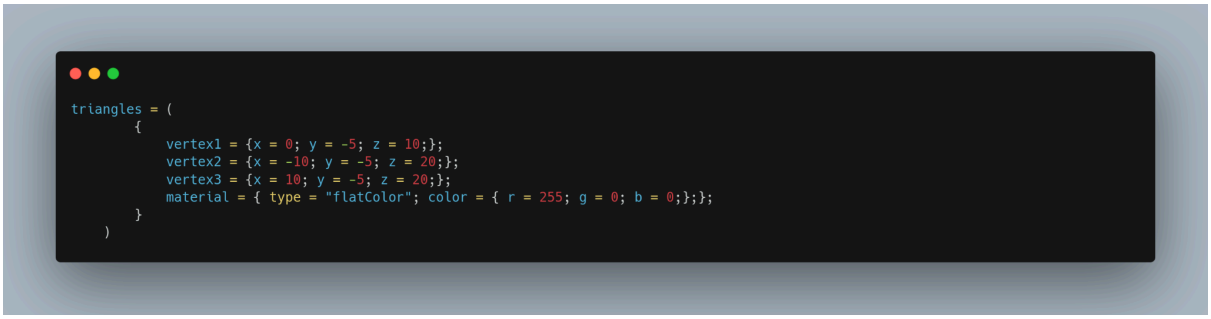


```
    rectangular_cuboids = (
        { minX = -200; minY = -105; minZ = 250; maxX = 45; maxY = 25; maxZ = 260; material = { type = "flatColor"; color = { r = 255;
g = 0; b = 255;};}; translation = {x = 100, y = 0, z = 0}; rotation = { x = 0.0; y = 0.0; z = 0.0; };},
    );
```

A **cube** needs the following parameters:

- **Coordinates : “minX” “minY” “minZ” “maxX” “maxY” “maxZ”**.
- **Material “material”** (Cf. 7.III.a)**.**

a. Triangles

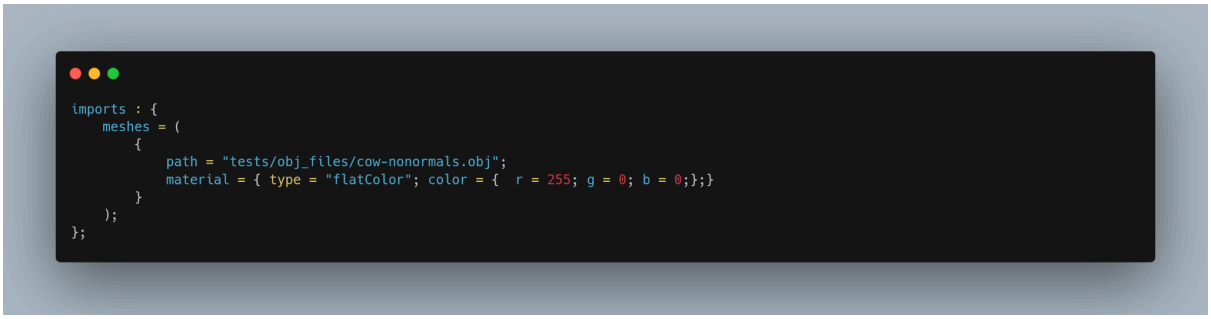


```
triangles = (
        {
            vertex1 = {x = 0; y = -5; z = 10;};
            vertex2 = {x = -10; y = -5; z = 20;};
            vertex3 = {x = 10; y = -5; z = 20;};
            material = { type = "flatColor"; color = { r = 255; g = 0; b = 0;};};
        }
    )
```

A triangle needs the following parameters:

- **“vertex1”** : **“x” “y” “z”** the coordinates of an angle.
- **“vertex2”** : **“x” “y” “z”** the coordinates of an angle.
- **“vertex3”** : **“x” “y” “z”** the coordinates of an angle.
- **A material “material”** (Cf. 7.III.a)**.**

a. .OBJ files



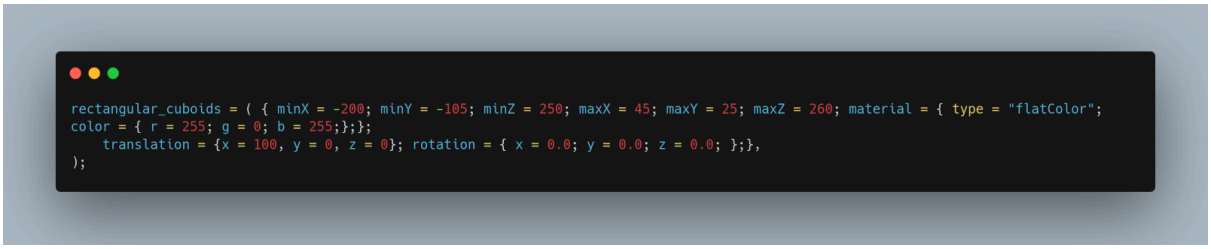
```
imports : {
    meshes = (
        {
            path = "tests/obj_files/cow-nonormals.obj";
            material = { type = "flatColor"; color = {  r = 255; g = 0; b = 0;};}
        }
    );
};
```

You can load an object, a .obj file, allowing you to load complex shapes (cows…). Our .obj reader only supports faces (f) and vertex (v). The import is done in **“meshes”** in **“imports”.** Each mesh takes the following parameters:

- Path to the file **“path”.**
- A material **“material”** (Cf. 7.III.a)**.**

g. Configuration transformations primitives:

You can move your primitives in space.

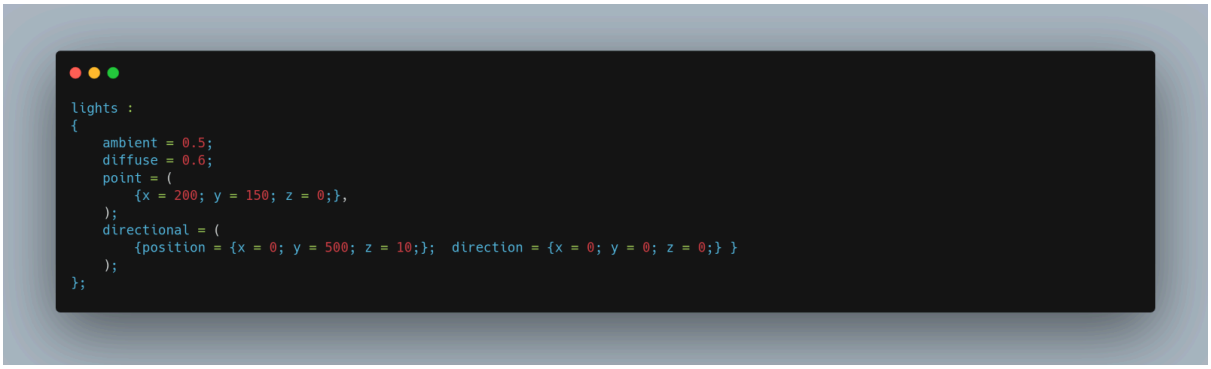


```
rectangular_cuboids = ( { minX = -200; minY = -105; minZ = 250; maxX = 45; maxY = 25; maxZ = 260; material = { type = "flatColor";
color = { r = 255; g = 0; b = 255;};};
    translation = {x = 100, y = 0, z = 0}; rotation = { x = 0.0; y = 0.0; z = 0.0; };},
);
```

Available : **translation** + **rotation**.

- **Rotation** needs “**x**” “**y**” “**z**”.
- **Translation** needs “**x**” “**y**” “**z**”.

IV. Lights configuration:



```
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;} }
    );
};
```

Lights supports these:

- “**ambient**” the ambient light. A multiplier.
- “**diffuse”,** the light diffusion multiplier.
- **“point”**, the lighting points
    - Each **light point** needs **“x” “y” “z”**
- **“directional”** directional lights
    - A **directional light** needs:
        - Coordinates **“x” “y” “z”**
        - Direction **“x” “y” “z”**

V. Loading a scene from another scene:

This "imports" configuration is optional. It allows you to import another scene or a .cfg file from a scene. Parameters such as **imports, primitives and lights** will be imported.

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

Imports can take scenes as parameters, each with a path link to the configuration.

Potential infinite loops are well managed: if a file imports itself, the program will detect it.