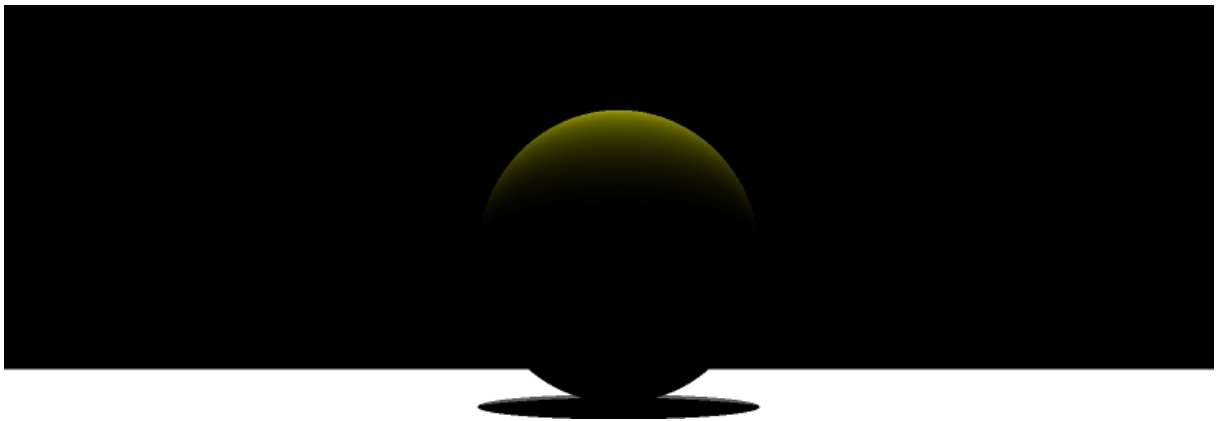




# Raytracer

## User documentation



### 1. Directed by :

- Axel Fradet [axel.fradet@epitech.eu](mailto:axel.fradet@epitech.eu)
- Mathieu Robert [mathieu1.robert@epitech.eu](mailto:mathieu1.robert@epitech.eu)
- Kylian Tranchet [kylian.tranchet@epitech.eu](mailto:kylian.tranchet@epitech.eu)
- Théophile Jérôme-Rocher [theophile.jerome-rocher@epitech.eu](mailto:theophile.jerome-rocher@epitech.eu)
  - Epitech Nantes PGE 2027. 2nd year.

## 2. Useful links (user):

- Github: <https://github.com/Njord201/Raytracer>

## 3. The project:

The Raytracer project is being carried out in groups of 4 as part of the 2nd year at Epitech. The aim is to create raytracing software, i.e. to create a realistic image based on a scene supplied by the user, which can contain shapes, lighting effects, etc. When executed, the programme takes the link to a .cfg file (scene configuration) as a parameter and displays it directly on the screen.

## 4. Supported by our Raytracer

- **Primitives**
  - Sphere
  - Infinite cylinder
  - Plan
  - Cubes
  - Infinite cones
    - Primitive materials (flatColor: flat colour)
- **Camera**
  - Resolution
  - Position
  - Rotation
  - FOV (Field of View)
- **Lights**
  - Ambient light
  - Diffused light
  - Light points
  - Directional lights
  - Shadows
- **Transformations:**
  - Rotation
  - Translation
- **Scenes:**
  - Loading a scene from a scene

## 5. Dependencies:

Make sure that these dependencies are met on your system so that the Raytracer works correctly.

- Make
- G++ (C++ compiler)
- SDL2
- Git (Have user logged in)

## 6. Installing + running Raytracer:

- Open your terminal
- Type "**git clone** <https://github.com/Njord201/Raytracer.git>"
- Type "**cd Raytracer/**"
- To compile the Raytracer properly, type "**make re**".
- To run it, type "**./Raytracer {path\_file.cfg}**" and enter a valid configuration file. Or **-help** for help.

## 7. Scenes (configuration):

### I. Preamble

When you run Raytracer, you provide a link to a .cfg file. Several examples are available in **tests/files\_samples/**.

Your configuration files, in English, must 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;};}; }
  );

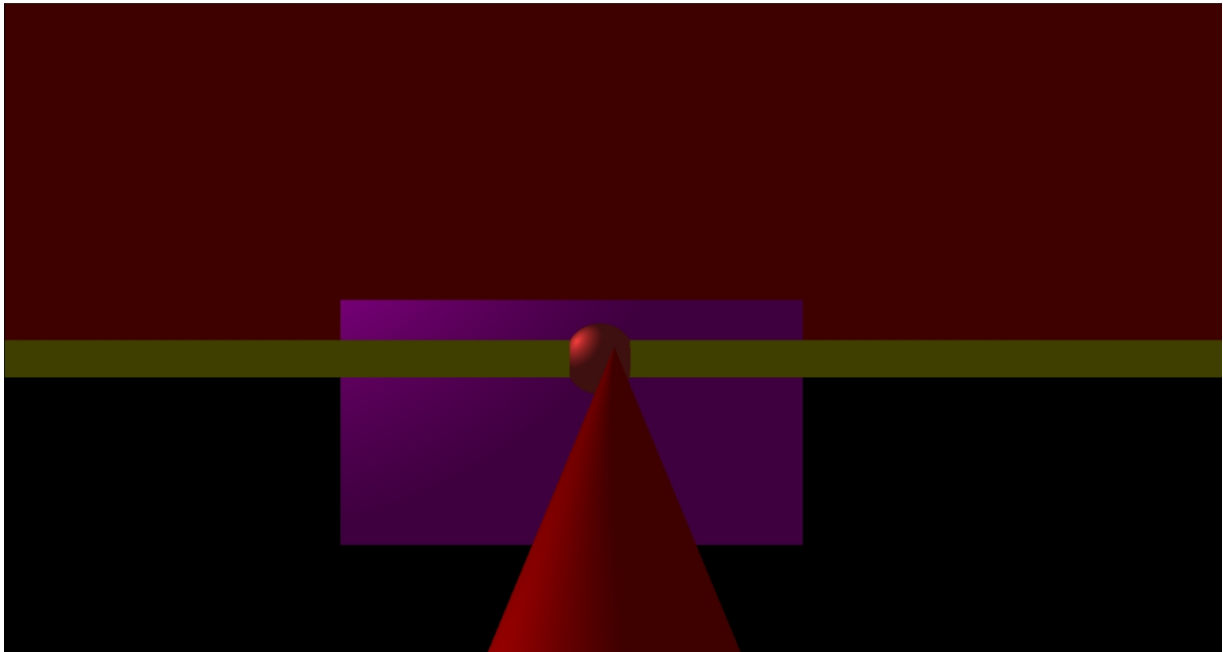
  # List of rectangular cuboids
  rectangular_cuboids = (
    { minX = -200; minY = -105; minZ = 250; maxX = 45; maxY = 25; maxZ = 260; material = { type = "flatColor"; color = { r = 255; g = 0; b = 255;};}; translation = {x = 100, y = 0, z = 0;}; },
  );
};

# 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. The 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 takes:

- **Resolution**
  - Width
  - Height
- **Position**
  - X Y Z
- **Rotation**
  - X Y Z
- The "**fieldOfView**" field of view **takes** °.

### III. Configuring primitives:

For each primitive (cylinders, spheres, etc.), you can specify several.

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

#### a. Colour/Materials of the primitives

We support flatColor. You can find this material in the following configurations.

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

A **material** takes parameters:

- **Type** (flatColor available).
- **Color**, itself taking "r" "g" "b" for the rgb colour of the flatColor.

#### b. Spheres

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

A **sphere** takes parameters:

- **Origin "x" "y" "z"**.
- **Radius "r"**.
- **Material** (Cf. 7.c.i).

### c. Cylinders

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

A **cylinder** takes parameters:

- **Origin "x" "y" "z".**
- **Radius "r".**
- **Axis.**
- **Material** (Cf. 7.c.i).

### d. Infinite cones

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

A **cone** takes parameters:

- **Origin "x" "y" "z".**
- **Angle** in °.
- **Axis "axis"** either X Y or Z.
- **Material** (Cf. 7.c.i).

### e. Plans

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

A **plan** takes into account the parameters:

- **Position.**

- **Axis "axis"** either X Y or Z.
  - **Material** (Cf. 7.c.i).
- 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 takes parameters:

- **Coordinates of its points:** minX minY minZ maxX maxY maxZ.
- **Material** (Cf. 7.c.i).
  - *Here is an example of translation + rotation.*

#### g. **Configuring primitive transformations:**

You can actually 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 in **translation** and **rotation**.

- **Rotation** takes x y z.
- **Translation** takes x y z.

#### IV. **Configuring the lights:**



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

The **lights** take parameters:

- **Ambient** light, a multiplier.
- The diffusion of light, "**diffuse**", a multiplier.
- "**point**" or light points
  - A point takes coordinates "**x**" "**y**" "**z**".
- "directional lights"
  - A directional light takes the parameter:
    - **x**" "**y**" "**z**" coordinates
    - Direction "**x**" "**y**" "**z**"

## V. Import configuration

This "**imports**" configuration is optional. It is used to import another scene or a .cfg file from a scene. The **import** parameters, **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, so if a file imports itself the program will detect it.*