Documentation for Raytracer configuration files

This document outlines the structure and format of the config.cfg file for configuring the raytracer project in CPP. The configuration file utilises the libconfig++ library for parsing. The file is divided into three main sections: Camera, Primitives, and Lights.

1. Camera configuration

The camera configuration specifies the camera's properties within the scene. The camera settings include resolution, position, rotation, and field of view. The syntax for the camera configuration is as follows:

```
camera :
{
    resolution = { width = 800; height = 600; };
    position = { x = 0; y = -100; z = 20; };
    rotation = { x = 0; y = 0; z = 0; };
    fieldOfView = 90.0; # In degree
};
```

- **resolution**: Sets the resolution of the rendered image. Replace <width> and <height> with the desired width and height in pixels.
- **position**: Sets the position of the camera within the 3D space. Replace <x>, <y>, and <z> with the desired X, Y, and Z coordinates.
- **rotation**: Sets the camera's rotation angles. Replace <x>, <y>, and <z> with the desired rotation angles in degrees.
- **fieldOfView**: Sets the camera's field of view in degrees. Replace <fov> with the desired field of view angle.

2. Primitives' configuration

The primitives' configuration defines the objects within the scene, including spheres and planes. The syntax for defining primitives is as follows:

```
primitives :
{
    spheres = (
        { x = <x>; y = <y>; z = <z>; r = <radius>; color = { r = <red>; g = <green>; b = <blue>; }; },
        # More sphere definitions here...
    );
    planes = (
        { axis = "<axis>"; position = <position>; color = { r = <red>; g = <green>; b = <blue>; }; },
        # More plane definitions here...
    );
    # More primitives here...
};
```

1. Spheres

- **x, y, z**: The X, Y, and Z coordinates of the sphere's centre.
- **r**: The sphere's radius.
- **colour**: The sphere's colour, defined by the RGB values <red>, <green>, and <blue>.

2. Planes

- **axis**: The axis which the plane is perpendicular to. Choose from "X", "Y", or "Z".
- **position**: The position of the plane along the specified axis.
- **colour**: The plane's colour, defined by the RGB values <red>, <green>, and <blue>.

3. Lights configuration

The lights configuration specifies the lighting within the scene, including ambient light, diffuse light, point lights, and directional lights. The syntax for defining lights is as follows:

```
lights :
{
    ambient = <ambient>; # Multiplier of ambient light
    diffuse = <diffuse>; # Multiplier of diffuse light
    point = (
        { x = <x>; y = <y>; z = <z>; },
        # More point light definitions here...
    );
    directional = (
        # More directional light definitions here...
    );
};
```

- **ambient**: The multiplier of the ambient light intensity. Replace <ambient> with the desired value.
- **diffuse**: The multiplier of the diffuse light intensity. Replace <diffuse> with the desired value.

3. Point lights

Point lights are light sources that emit light in all directions from a single point in the 3D space. The syntax for defining point lights is as follows:

```
point = (
    { x = <x>; y = <y>; z = <z>; },
    # More point light definitions here...
);
```

- **x, y, z**: The X, Y, and Z coordinates of the point light's position.

4. Directional lights

Directional lights are light sources that emit parallel light rays in a specific direction. They can be considered as infinitely distant light sources, such as sunlight. The syntax for defining directional lights is as follows:

```
directional = (
{ x = <x>; y = <y>; z = <z>; },
# More directional light definitions here...
);
```

- **x, y, z**: The X, Y, and Z components of the directional light's direction vector. The vector should be normalized.

4. Transformations

The transformations configuration allows you to apply translations and rotations to the primitives in the scene. The syntax for defining transformations is as follows:

```
transformations:
{
    # List of transformations
    translations = (
        { index = <index>; x = <x>; y = <y>; z = <z>; },
        # More translation definitions here...
    );
    rotations = (
        { index = <index>; angle = <angle>; x = <x>; y = <y>; z = <z>; },
        # More rotation definitions here...
    );
};
```

1. Translation

Translations are used to move primitives along the X, Y, and Z axes. The syntax for defining translations is as follows:

```
translations = (
    { index = <index>; x = <x>; y = <y>; z = <z>; },
    # More translation definitions here...
);
```

- **index**: The index of the primitive to be translated, where the first primitive has an index of 0, the second has an index of 1, and so on.
- **x, y, z**: The translation distances along the X, Y, and Z axes.

2. Rotations

Rotations are used to rotate primitives around a specified axis. The syntax for defining rotations is as follows:

```
rotations = (
    { index = <index>; angle = <angle>; x = <x>; y = <y>; z = <z>; },
    # More rotation definitions here...
);
```

- **index**: The index of the primitive to be rotated, where the first primitive has an index of 0, the second has an index of 1, and so on.
- **angle**: The rotation angle in degrees.
- **x, y, z**: The X, Y, and Z components of the rotation axis. The axis should be normalized.