Raytracing : User Manual

Introduction

This project will allow the user to create different images with a high degree of visual realism using the ray tracing technique.

Description

> In computer graphics, ray tracing is a rendering technique for generating an image by tracing the path of light as pixels in an image plane and simulating the effects of its encounters with virtual objects. The technique is capable of producing a high degree of visual realism, more so than typical scanline rendering methods, but at a greater computational cost. This makes ray tracing best suited for applications where taking a relatively long time to render can be tolerated, such as in still computer-generated images, and film and television visual effects (VFX), but more poorly suited to real-time applications such as video games, where speed is critical in rendering each frame.Ray tracing is capable of simulating a variety of optical effects, such as reflection and refraction, scattering, and dispersion phenomena (such as chromatic aberration). raytracing

Installation

Linux installation of sdl library

```
sudo apt install libsdl2-2.0-0 libsdl2-gfx-1.0-0 libsdl2-image-2.0-0 libsdl2-mixer-2.0-0 libsdl2-net-2.0-0 libsdl2-ttf-2.0-0
```

Refer to sdl for more details

Usage

In the repertory where there the Makefile type:

```
make
```

Executable:

That will generate the executable called lray.

Options	Description
-n	Determines the level of the program 1 : Will only create the image with the light. 2: Generates a graphic window that allow us to move the camera. 3 : Will launch recursive interactive rays, calculate the shadows and allow us to create mirror and transparent objects.
-i	The path file ***.txt*** that as the description of the scene
-o	The file ***.ppm*** (has to be this format) where the image will be saved. (PS: for the level 2 you can save the image with the key s of the keyboard)
-ps	Determines how many recursive interactive rays it will launch. (PS: It is only active for the level 3 and the bigger the number the waiting time will increase)

For example the user can type

```
./lray -n 3 -i scene.txt -o image.ppm -ps 16
```

Graphic window:

If by any chance the user decides to use the level 2 the following commands to play with the camera are:

Categories	Commands	Description
position	▲ ▼ ◄ ►	Moves the camera position upwards Moves the camera position downwards Moves the camera position leftwards Moves the camera position rightwards

Categories	Commands	Description
look at	i k j l	Moves the camera look at point upwards Moves the camera look at point downwards Moves the camera look at point leftwards Moves the camera look at point rightwards
axis	x y z	Positions the camera in the x-axis Positions the camera in the y-axis Positions the camera in the z-axis
zoom	+ -	Zoom in Zoom out

Input Files format:

As said before the input files have to ***.txt*** the different separator are *{ }* / The colors *red green blue* should always be between 0 and 1. The values of a variable *is* are 0 for false and 1 for true. The text before the *{* can be in upper, lower or both cases.

Categories	Format	Description
image	Image { width height | (optional) red green blue}	the color will determine the background color
camera	Camera {pos_x pos_y pos_z | look_x look_y look_z | axis_x axis_y axis_z | fov | phi | aperture | dist}	the fov should be in degrees and the axis only one can value 1 the others shall value 0
Light	Light {pos_x pos_y pos_z | red green blue | intensity}	the intensity should be very elevated otherwise cannot see the scene 3e+07 as reference
sphere	Sphere {pos_x pos_y pos_z | red green blue | radius |(optional) ambiance_red ambiance_green ambiance_blue diffuse_red diffuse_green diffuse_blue specular_red specular_green specular_blue exponent is_mirror is_transparent}	for the optional phong lighting
cone	Cone {center_x center_y center_z | radius | height | red green blue |(optional) ambiance_red ambiance_green ambiance_blue diffuse_red diffuse_green diffuse_blue specular_red specular_green specular_blue exponent is_mirror is_transparent}	for the optional phong lighting
cylinder	Cylinder {center_x center_y center_z | radius | height | red green blue |(optional) ambiance_red ambiance_green ambiance_blue diffuse_red diffuse_green diffuse_blue specular_red specular_green specular_blue exponent is_mirror is_transparent}	for the optional phong lighting
plane	Plane {pos_x pos_y pos_z | normal_x normal_y normal_z | red green blue |(optional) ambiance_red ambiance_green ambiance_blue diffuse_red diffuse_green diffuse_blue specular_red specular_green specular_blue exponent is_mirror is_transparent}	for the optional phong lighting
cube	Cube {p1_x p1_y p1_z | p2_x p2_y p2_z | p3_x p3_y p3_z | p4_x p4_y p4_z | red green blue | (optional) ambiance_red ambiance_green ambiance_blue diffuse_red diffuse_green diffuse_blue specular_red specular_green specular_blue exponent is_mirror is_transparent}	for the optional phong lighting

Build with

- [ ] sdl - The simple directmedia layer

Authors

- [ ] Ailton LOPES MENDES
- [ ] Fabien LAMBERT-DELAVAQUERIE

Bugs

The transparency does not work that well with the cube it might produce a segmentation fault. The shadows are not perfect. If the command line is not written properly the program will not work correctly. The graphic window since it will reder the image every time it can be very slow.

Conclusion

The user can create different scenes with this version of ray tracing.
He shall respect the format given otherwise there might be some problems.