# Ray Tracing Group 6 - Project Documentation

## Project Overview

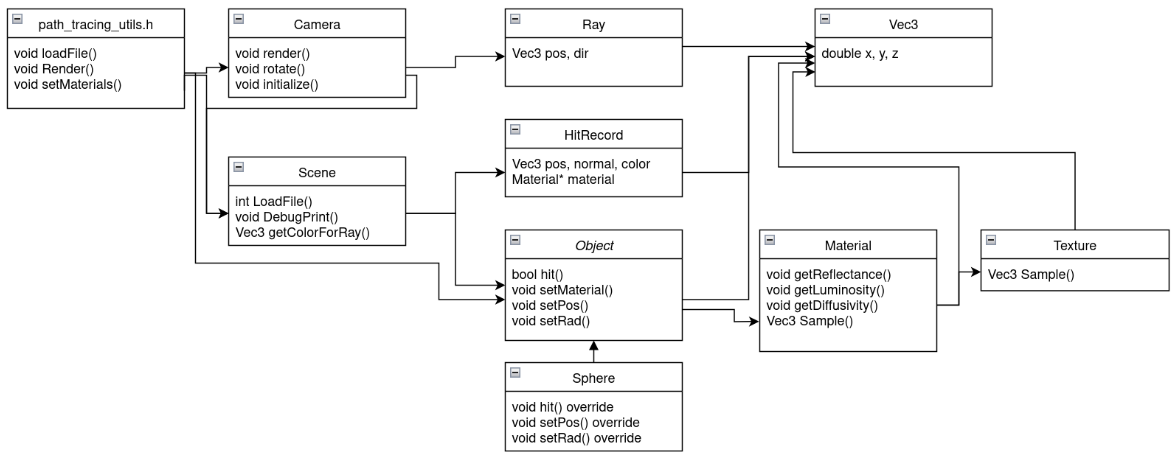
Our software project is about path tracing. In this project you use the GUI interface to read input files containing scene information about spheres which represent planets and stars. After loading the file, you can render a picture of a scene where planets have detailed textures and stars work as lightsource. There is also implemented metal or mirror kind of objects which reflects light. When rendering pictures you can change camera position, render mode, samples per pixel, recursion depth and objects position and other properties.

Shortly scene is created in huge sphere with milky way texture and planets and stars inside of it. Camera shoots rays for every pixel and when they hit objects, rays scatter recursively and finally set the color for each pixel.

## Software structure

The directory structure follows a hybrid out-of-source build configuration, where the backend source files are situated in the /src folder, and the main.cpp files are situated in build directories /gui and /gui_ubuntu for Windows and Linux Ubuntu, respectively.

The main class relationships of the backend are represented in the UML diagram below.



Classes and functions:

- Camera: render(), rotate(), initialize()
- Scene: LoadFile(), DebugPrint(), getColorForRay()
- Ray: pos, dir
- HitRecord: pos, normal, color, material
- Object: hit(), setMaterial(), setPos(), setRad()
- Sphere: hit(), setPos(), setRad()
- Vec3: x, y, z

- Material: getReflectance(), getLuminosity(), getDiffusivity(), Sample()
- Texture: Sample()

External libraries used (included in /libs, no need to install anything): **GLFW, BitMapPlusPlus, stb**

## Instructions for building and using the software

*Building*

The software is tested on Windows and Linux.

**If you are using Windows**, navigate to the folder “gui” with the Developer Command Prompt for Visual Studio. Use the command

```
MSBuild gui_test.sln /p:Configuration=Release
```

The executable should build to the Release folder.

**If you are using Linux or Windows Subsystem for Linux**, navigate to the folder “gui_ubuntu” and run the following commands:

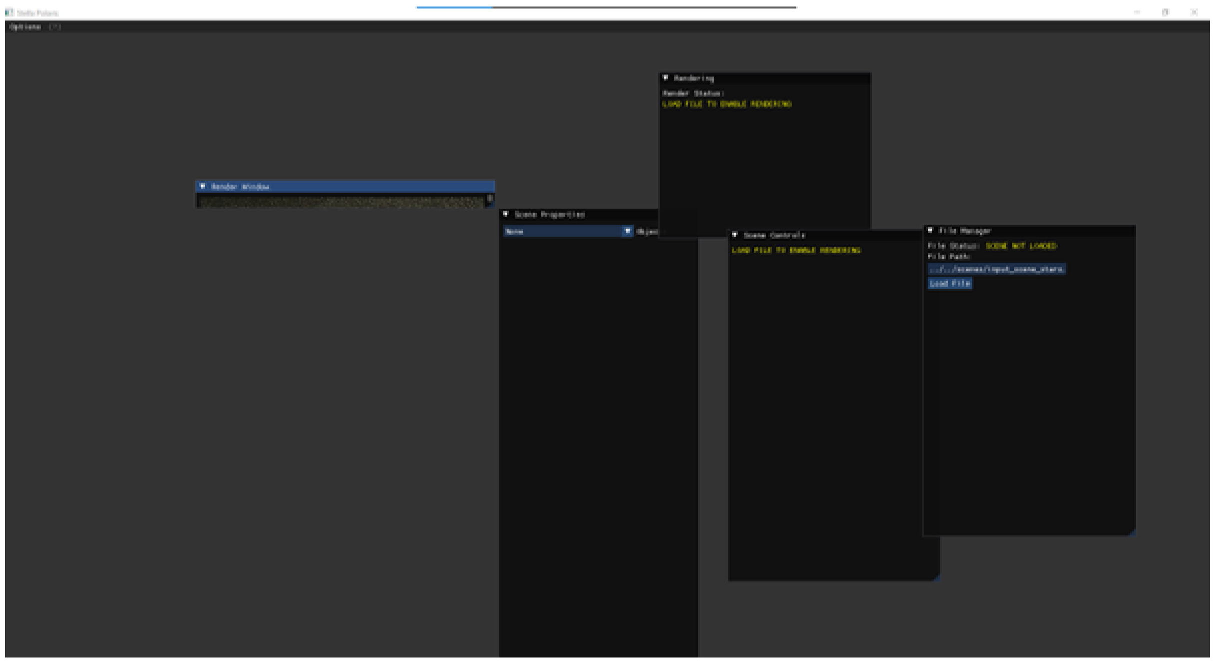
```
cmake .
```

```
cmake --build .
```

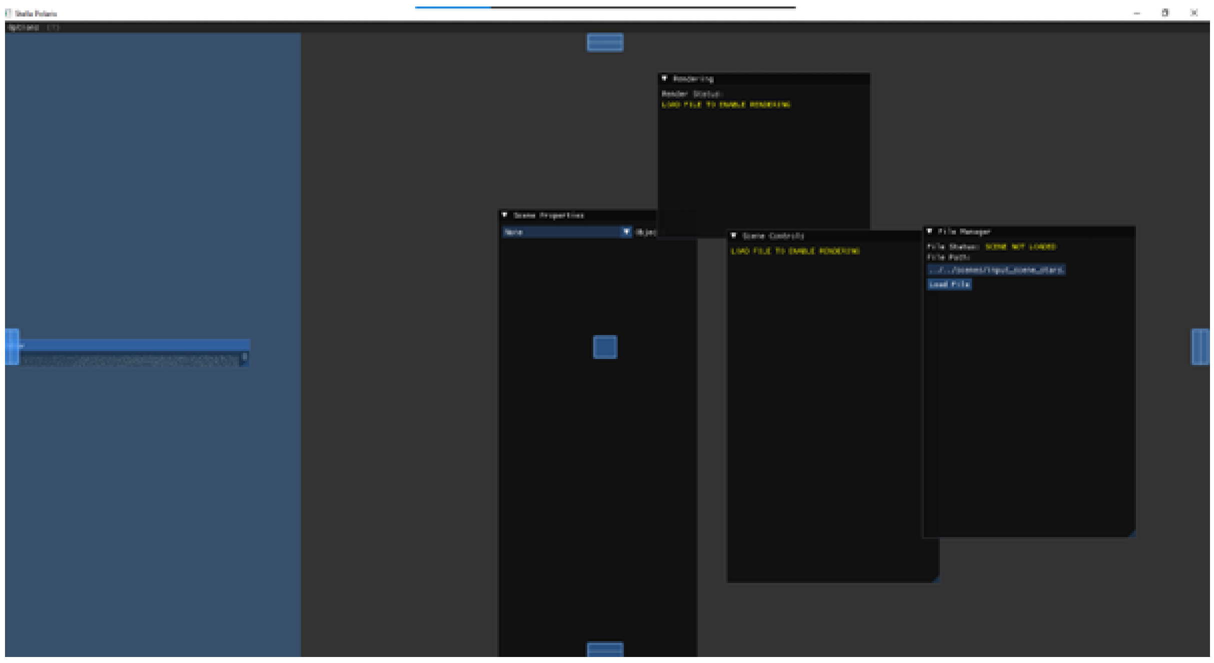
Note that the software uses external libraries. The CMakeLists should be sufficient to locate them. The user shouldn’t need to install anything. In for some reason the glfw library is not found, it can be installed for example with apt-get install libglfw3 and apt-get install libglfw3-dev (requires sudo).

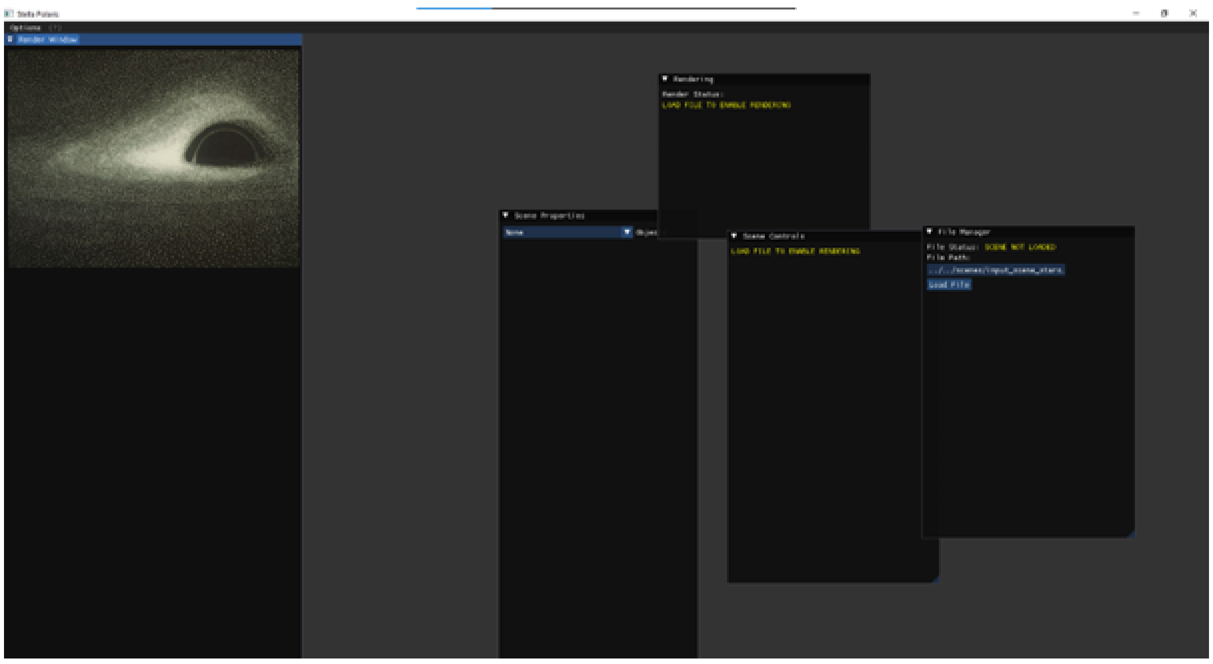
*Setup*

The GUI uses ImGui docking branch. Depending on which platform you build, the software windows might initially be in a state of disarray.



Simply drag the windows from their titles to the symbols that appear and you can organize the setup to your liking. The setup should save automatically.





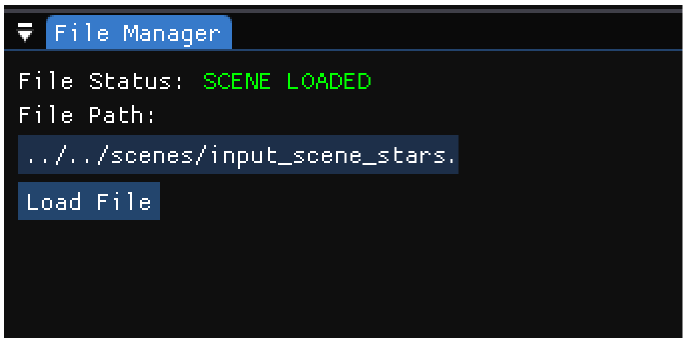
Render Status:
LOAD FILE TO ENABLE RENDERING

LOAD FILE TO ENABLE RENDERING

File Status: SCENE NOT LOADED
File Path:
../../scenes/input_scene_stars.

*File Importing*

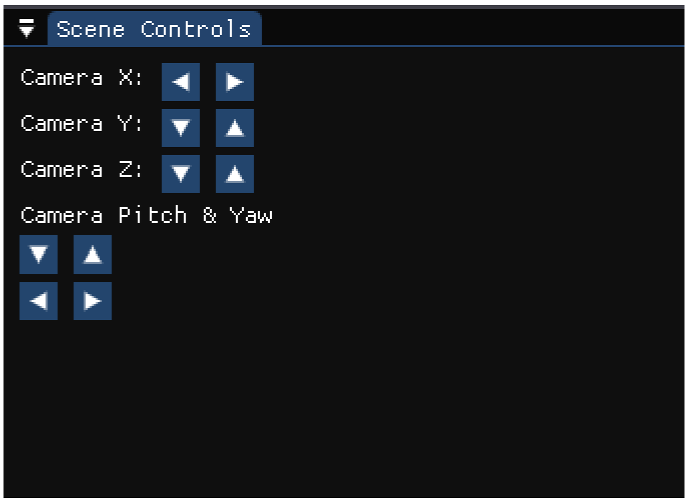
Load scene using the file manager. The predefined path is set up so that it should point to the correct demo file. Upon successful file load, more controls are displayed.



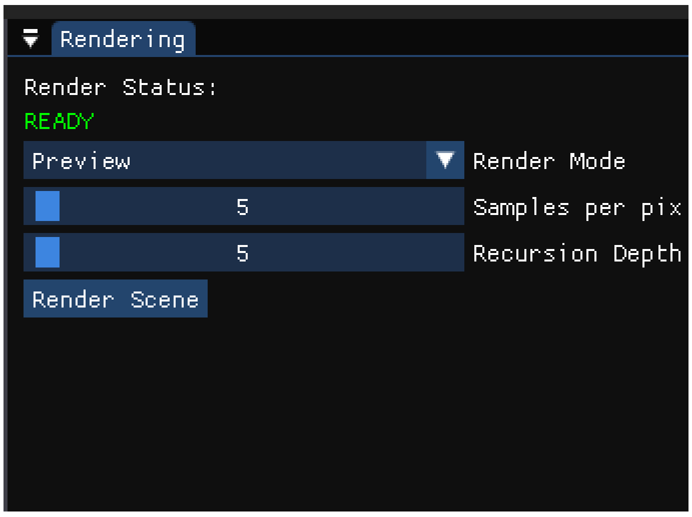
File Status: SCENE LOADED
File Path:
../../scenes/input_scene_stars.

*Rendering*

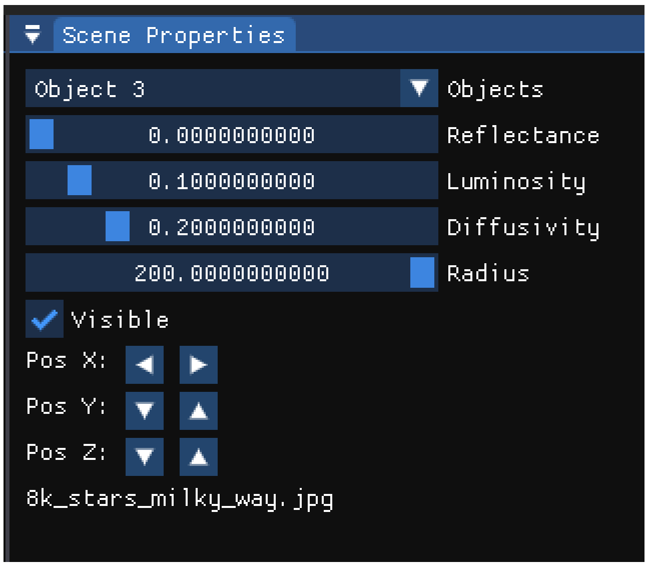
The GUI is very self-evident.



Scene Controls tab allows you to control the camera. It automatically renders a preview at every movement.


Render Status:
READY
Preview — Render Mode
5 — Samples per pix
5 — Recursion Depth

Rendering tab allows you to define the render settings. Preview is for fast rendering, and it always sets samples per pixel and recursion depth to 1, but makes all objects in the scene luminous. Pick “Final” from the dropdown menu to render according to your settings. If you want to have a higher number than the slider allows, hold ctrl and click the slider to input the number manually.


Scene Properties

| | |
|---|---|
| Object 3 | Objects |
| 0.0000000000 | Reflectance |
| 0.1000000000 | Luminosity |
| 0.2000000000 | Diffusivity |
| 200.0000000000 | Radius |

✓ Visible

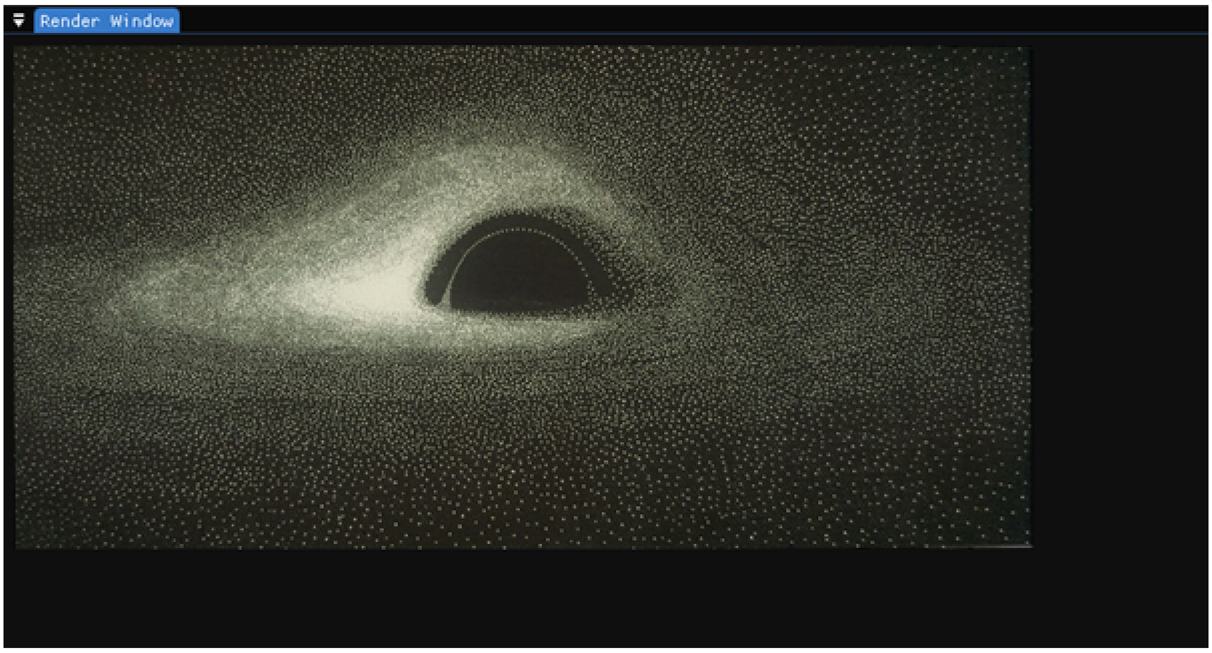
Pos X: ◀ ▶

Pos Y: ▼ ▲

Pos Z: ▼ ▲

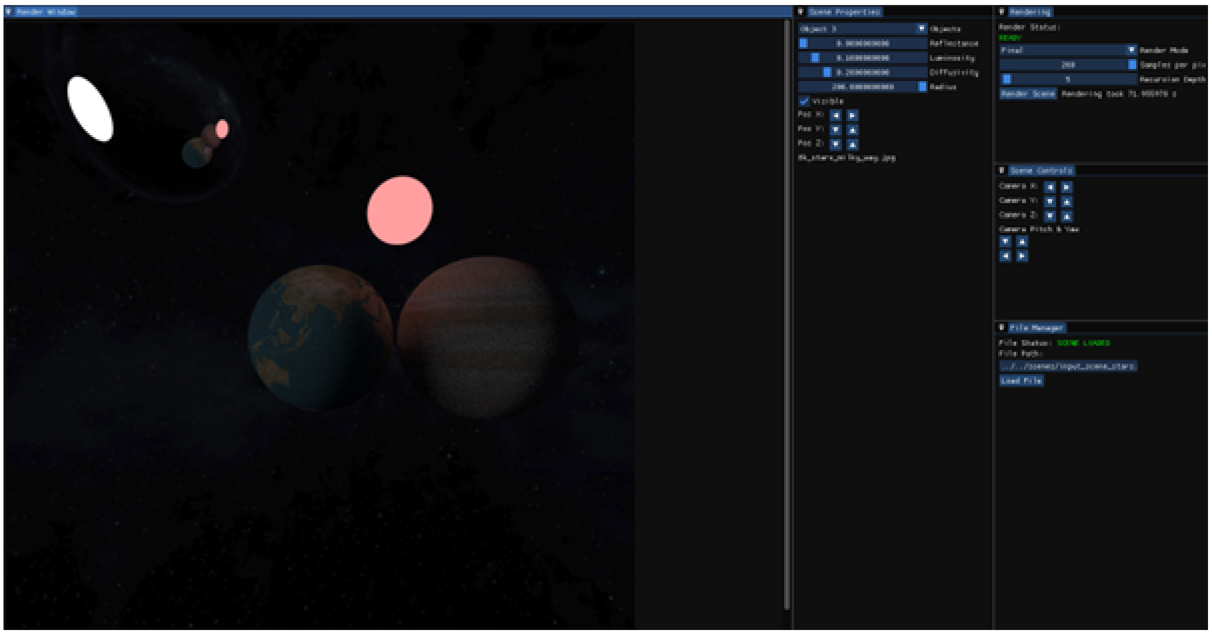
8k_stars_milky_way.jpg

Scene Properties tab allows you to modify the attributes of individual objects. For textured objects, the texture file name is displayed. Since the objects aren't named in the source scene file, object names are generated sequentially.



Render Window should display the placeholder image (first render of a black hole, 1979 © Jean-Pierre Luminet)

At render, the image will be replaced by the rendered one.



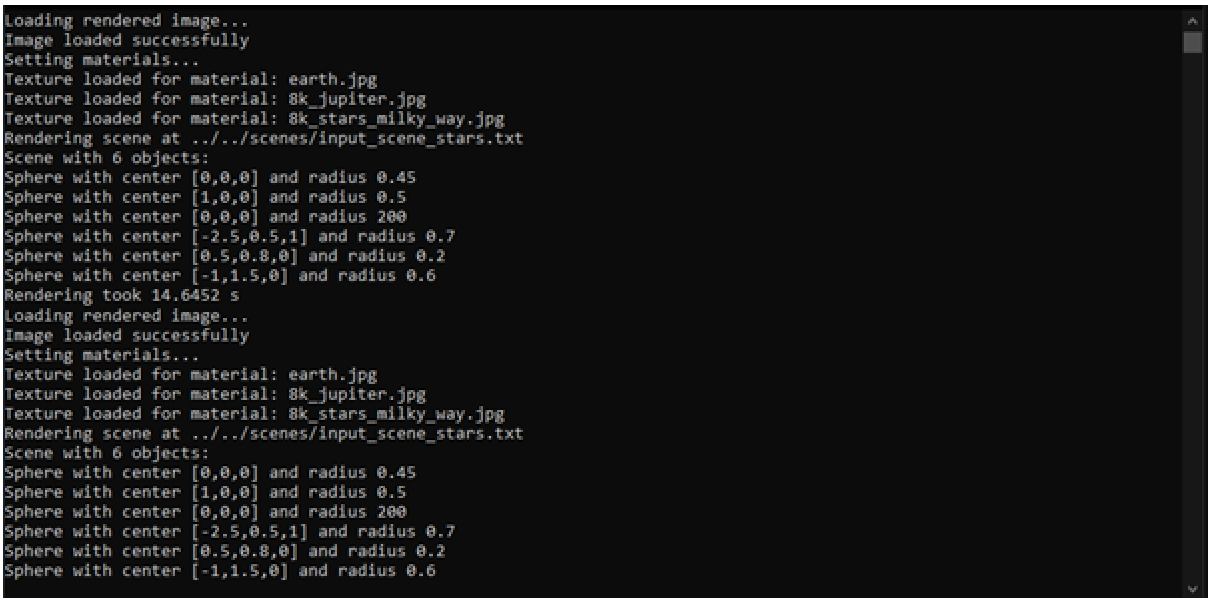
```
Object 3                Objects
0.0000000000            Reflectance
0.1000000000            Luminosity
0.2000000000            Diffusivity
200.0000000000          Radius
Visible
Pos X:
Pos Y:
Pos Z:
8k_stars_milky_way.jpg

Render Status:
READY
Final                   Render Mode
200                     Samples per pix
5                       Recursion Depth
Render Scene  Rendering took 71.955970 s

Camera X:
Camera Y:
Camera Z:
Camera Pitch & Yaw

File Status: SCENE LOADED
File Path:
../../scenes/input_scene_stars
```

The console will display some information about the program as it runs.



```
Loading rendered image...
Image loaded successfully
Setting materials...
Texture loaded for material: earth.jpg
Texture loaded for material: 8k_jupiter.jpg
Texture loaded for material: 8k_stars_milky_way.jpg
Rendering scene at ../../scenes/input_scene_stars.txt
Scene with 6 objects:
Sphere with center [0,0,0] and radius 0.45
Sphere with center [1,0,0] and radius 0.5
Sphere with center [0,0,0] and radius 200
Sphere with center [-2.5,0.5,1] and radius 0.7
Sphere with center [0.5,0.8,0] and radius 0.2
Sphere with center [-1,1.5,0] and radius 0.6
Rendering took 14.6452 s
Loading rendered image...
Image loaded successfully
Setting materials...
Texture loaded for material: earth.jpg
Texture loaded for material: 8k_jupiter.jpg
Texture loaded for material: 8k_stars_milky_way.jpg
Rendering scene at ../../scenes/input_scene_stars.txt
Scene with 6 objects:
Sphere with center [0,0,0] and radius 0.45
Sphere with center [1,0,0] and radius 0.5
Sphere with center [0,0,0] and radius 200
Sphere with center [-2.5,0.5,1] and radius 0.7
Sphere with center [0.5,0.8,0] and radius 0.2
Sphere with center [-1,1.5,0] and radius 0.6
```

## Testing

Unit testing for the modules implemented in this project was done using custom unit tests. For each unit test, a separate folder containing CMakeLists.txt and main.cpp files accompanied with relevant input files was created to allow out-of-source test builds with respect to the source directory. The main.cpp file as well as input files were customised for each test separately.

All major modules were tested in a total of 6 unit test directories:

tests/unit_test1: Scene class and input file loading functionalities.
tests/unit_test2: Object and Sphere class, and scene.hit function.
tests/unit_test3: Camera class.
tests/unit_test4: Interface between Camera and Sphere classes.
tests/unit_test5: Material and Texture classes.
tests/unit_test6: Integration of material and texture specifications to input file.

All tests revealed various shortcomings and bugs, which were subsequently successfully fixed.

## Work log

24.10.2023
-We wrote a project plan during the meeting.

3.11.2023
- Aleksanteri has implemented the base GUI.
- Mikael implemented the first version of proprietary input format for geometric objects and a basic unit test.
- Discussed the distribution of work between everyone.
- Agreed on the use of CMake for cross-platform compatibility.

15.11.2023
- Aleksanteri has advanced the implementation of GUI.
- Mikael has further advanced the vectors class and added the HitRec functionality.
- Henry has implemented the first versions of camera and ray classes.
- Tommi has implemented the material class with the addition of texture class for texture information.

24.11.2023
- Aleksanteri has further advanced the integration of GUI so that building the program now launches an operating window.
- Mikael implemented the render class and render function that renders and saves an image.
- Henry has improved the camera and scene class.
- Tommi has worked on class texture loading and object properties.

29.11.2023
- Aleksanteri has advanced the GUI so that it works with the back-end and actually shows a rendered image.
- Mikael has advanced the render class so that actual image rendering happens.

- Henry has worked on ray functions and material diffusion.
- Tommi has worked on texture loading and material application to object.

1.12.2023
- Aleksanteri has worked on the GUI.
- Mikael has worked on anti-aliasing.
- Henry has worked on material diffusion.
- Tommi has worked on simplifying scene initialization.