ELEC-A7151 Project documentation

Path Tracing

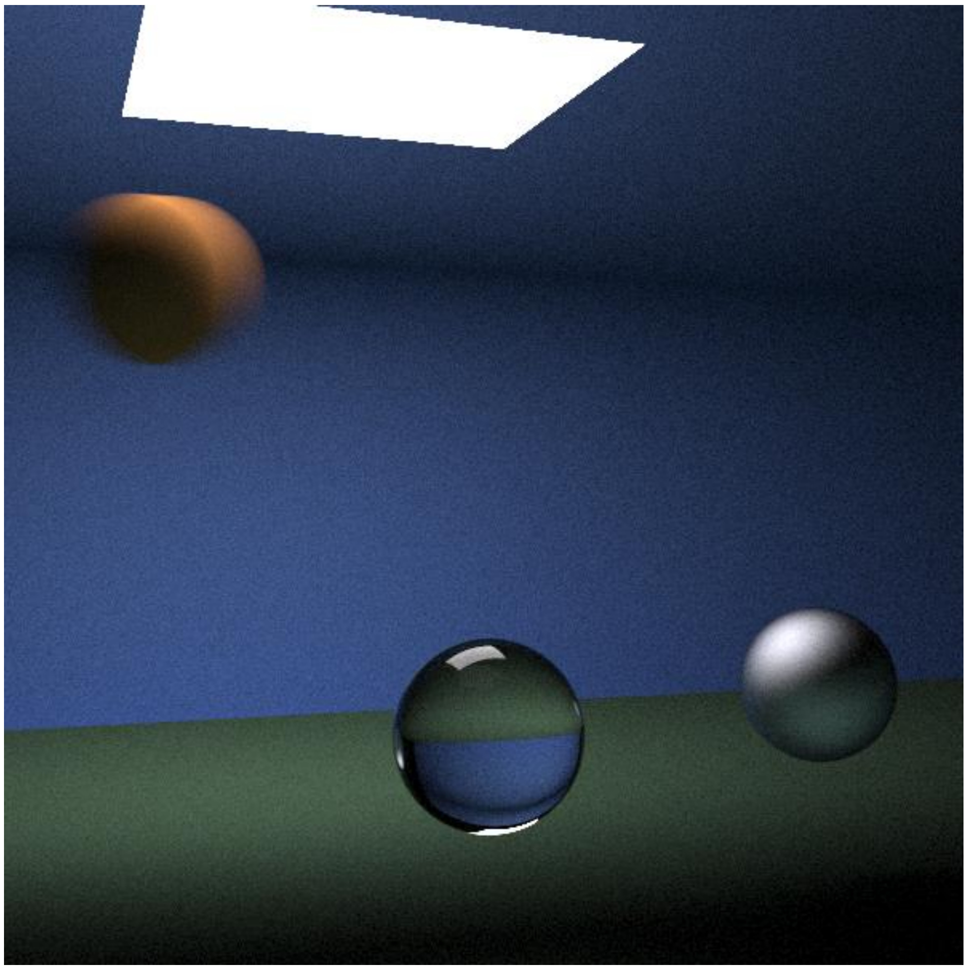
Group:	cpp-pt-4
	Dang Ly
	Teodors Kerimovs
	Verneri Hakkarainen
	Lauri Wilppu
Advisor:	Markus Säynevirta

I. Project overview

This project is a ray tracer with basic features and a graphical user interface for image preview. Ray tracer is used to simulate light rays to make more realistic pictures. Our ray tracer features natural light from the sky or additionally from light source objects. Other features include different shapes: ball, box and rectangle. The camera is modifiable with changeable zoom, position, orientation, field of view and shutter speed for moving objects. The project also features different kinds of materials: matte colors, glossy (metal) colors and glass.

The ray tracer is built on different classes working together: Rays, objects in the world, camera, and list of objects in the world. The rays travel from the camera to the objects like in many other ray tracers. The structure and code of this project is inspired heavily on Peter Shirley's book on the topic. (https://raytracing.github.io/v3/books/RayTracingInOneWeekend.html)

The software we have implemented reads a .txt file and makes a 3D scene out of it. The rendered picture is outputted as .jpg. The GUI lets you see a preview of the world and of the camera position. Software is built using cmake and run through terminal. The rendering of the scene may take several minutes.



An example scene from the ray tracer

II. Software structure

Please check the /doc/ folder for doxy generated software structure

III. Build instructions

Please install cMake 3.20 or higher for building our raytracer project.

1. To clone repo:

```
git clone https://version.aalto.fi/gitlab/kerimot1/cpp-pt-4.git
git submodule update --init
```

2. To build raytracer:

This should create a run_raytracer binary located in /build/

```
cd cpp-pt-4
mkdir build
cd build
cmake -DBUILD_RAYTRACER=ON -DBUILD_TESTS=OFF ..
make
```

3. To create a demo out.jpg in build folder

```
./run_raytracer
```

4. For compiling custom scene from file

```
./run_raytacer ../scenes/cornell_box.txt
```

5. To define a custom scene, please see examples in ../scenes/ and following instructions.
6. To build raytracer's unit tests after:

This should create a run_unit_tests binary located in /build/tests/ folder

```
cd -pt-4
mkdir build
cd build
cmake - cpp DBUILD_RAYTRACER=OFF -DBUILD_TESTS=ON ..
make
```

7. To run unit tests used in test driven development

```
cd tests
./run_unit_tests
```

IV. Scene file documentation

The scene files are .txt files where you write the parameters for the image, camera and a list of objects in a world. The adding of objects is relative to the constructors in the number, order and type of parameters. Here’s an example that showcases and explains the syntax, commands and format of the input file.

Camera and image input have always the same parameters.

Object input consists of object type then material and material parameters and then object parameters in order of the constructor. Please review the doxygen file.

Example of a file.txt:

```
BSCENE  Begin scene. Marks the start of the scene file.
    BIMAGEPARAMS  Begin image parameters.
    aspect_ratio 1.0 / 1.0
        image_width 600
        samples_per_pixel 40
        max_depth 40
    EIMAGEPARAMS End inputting image parameters.

  BCAMERAPARAMS  Begin inputting camera parameters.
        lookfrom 300 300 300
        lookat 0.0 0.0 0.0
        vup 1 0 0
 vfov 20
        aperture 0.01
        dist_to_focus 10.0
        background_color 0 0 0
        skylight 0 Lets the user choose between natural sky light or night time.
    ECAMERAPARAMS  End camera parameter input.

  BADDOBJ  Begin input of a new object to be added to the scene.
        object_sphere  Type of object.
        material metal  Material of object.
         color 1 0 0 Color of material.
         coef 0.5 The added fuzz of metal.
        radius 30.0 Radius of the sphere
        location 20.0 20.0 0.0 Location of the sphere.
```

`EADDOBJ` *End input of this current object.*

```
   BADDOBJ
        object_box
        material diffuse_light
        color 15 15 15
        minpoint 0.0 0.0 0.0
        maxpoint 50.0 50.0 50.0
    EADDOBJ
```

`BADDOBJ`

`object_yz_rect` *Also other orientations available.*

```
        material lambertian
        color 0.50 2 60
        miny 0.0
        maxy 100
minz 0
         maxz 100
         x 0
    EADDOBJ
```

`ESCENE` *Marks the end of the scene file and input.*

V. Testing

The tests are implemented using googletest for fast testing. https://github.com/google/googletest

Material tests

Material tests are simple and test the scatter effect of the materials. Lights are tested just by testing the emitting factor.

Vec3 tests

Vec3 tests test whether our vector class calculates vector calculations correctly.

Camera tests

Camera tests test the camera constructor and that the moving of the camera works correctly.

Utility tests

Utility tests test commonly used functions like the random functions. Random functions are checked to be within the parameters.

Ray tests

Ray tests check that the member functions work correctly and that the ray calculates a correct point at a certain length value along the ray.

VI. Work log

1. Verneri:

Materials: metal, lambertian, light, dielectric.

Utility functions

Raycolors without skylight

Meeting notes

Missing doxy comments

Scene design

Unit tests for test driven development

2. Lauri:

Box

Camera

Render

Rect

Quick troubleshooting errors

Unit tests for test driven development

3. Dang:

Sphere

Anti-aliasing

Finalized the project plan

Implementing GUI

Unit tests for test driven development

4. Teodors:

Vector class

Ray class

Moving sphere

CMake

Scene file reader

Unit tests for test driven development

For more detailed information check the meeting notes.