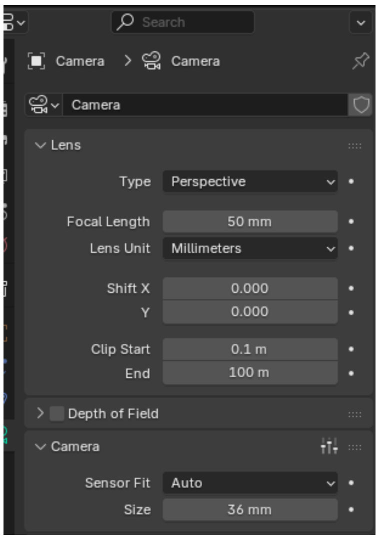
Computer Graphics: Rendering Coursework Report

James Hocking : s2900180

Module 1

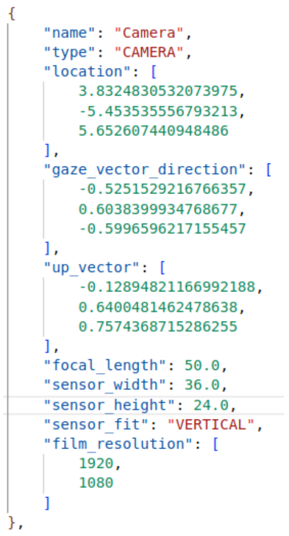
For the first module, a Blender scene exporter has been written that turns a Blender scene into a JSON file. A Camera class has also been implemented that uses the JSON from part 1 and stores the key information of the camera. Furthermore, a function that converts between a pixel coordinate and a ray vector has been written. Finally, a class that reads, updates and writes to `.ppm` has been created. In order to test/debug, the following was used for each section:

- **Blender scene exporter** - to test this, a simple scene with a point light, camera and each type of Mesh was created and moved around. The values in the JSON that was exported was then directly compared to the values in the Blender UI. All the values lined up correctly.



```
Camera > Camera
Camera
Lens
    Type          Perspective
    Focal Length  50 mm
    Lens Unit     Millimeters
    Shift X       0.000
          Y       0.000
    Clip Start    0.1 m
         End      100 m
Depth of Field
Camera
    Sensor Fit    Auto
    Size          36 mm
```

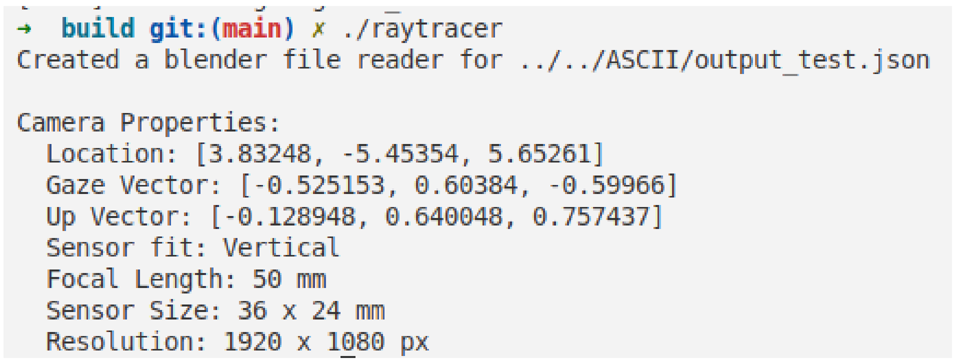
(a) Blender UI showing the camera specification



```
{
    "name": "Camera",
    "type": "CAMERA",
    "location": [
        3.8324830532073975,
        -5.453535556793213,
        5.652607440948486
    ],
    "gaze_vector_direction": [
        -0.5251529216766357,
        0.6038399934768677,
        -0.5996596217155457
    ],
    "up_vector": [
        -0.12894821166992188,
        0.6400481462478638,
        0.7574368715286255
    ],
    "focal_length": 50.0,
    "sensor_width": 36.0,
    "sensor_height": 24.0,
    "sensor_fit": "VERTICAL",
    "film_resolution": [
        1920,
        1080
    ]
},
```

(b) JSON file created, showing alignment with example fields

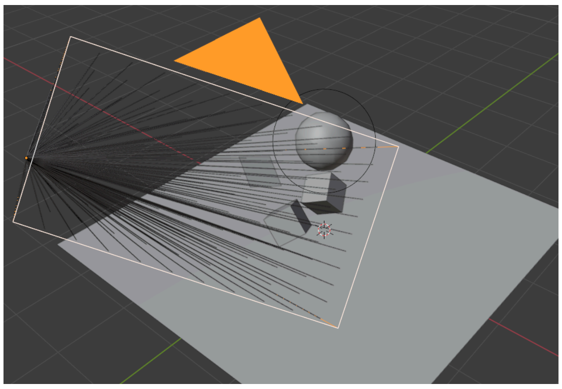
- **Camera space transformations** - for the first part of this, after reading the file the values within C++ were printed to standard output and compared with the values in the JSON file, which lined up.



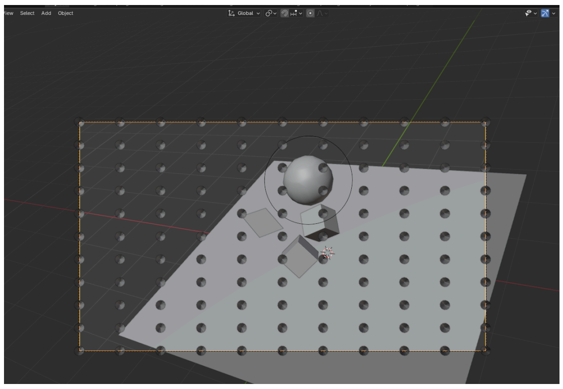
```
➜  build git:(main) ✗ ./raytracer
Created a blender file reader for ../../ASCII/output_test.json

Camera Properties:
  Location: [3.83248, -5.45354, 5.65261]
  Gaze Vector: [-0.525153, 0.60384, -0.59966]
  Up Vector: [-0.128948, 0.640048, 0.757437]
  Sensor fit: Vertical
  Focal Length: 50 mm
  Sensor Size: 36 x 24 mm
  Resolution: 1920 x 1080 px
```

Figure 2: Terminal output after running the JSON file and turning into a C++ object

Finally, for the pixel to ray convertor, the Ray visualizer developed in the Lab 2 was used in order to match the pixel values to the visual ray and show that it corresponded to the exact screen position in the world basis. This converted pixels into a origin and direction which was saved to a `.txt` file and then imported in blender and shown as cylinders.

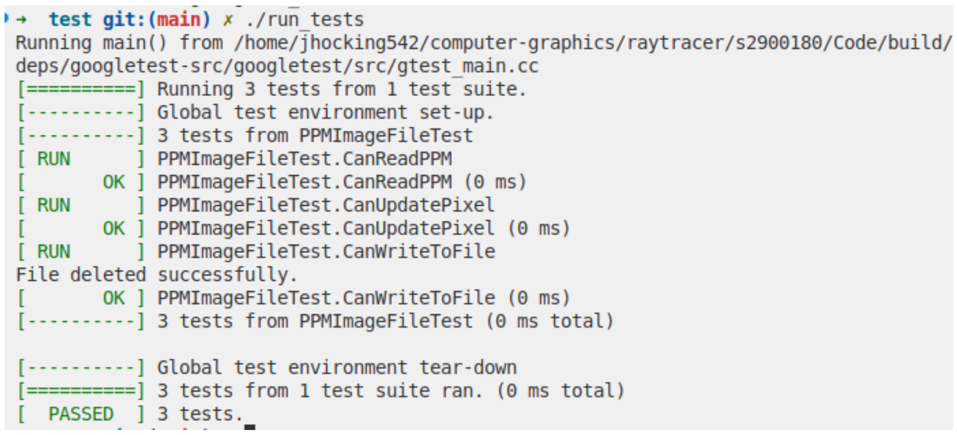


(a) Rays created as cylinders in Blender from the camera origin down to the scene.



(b) Camera View showing that the rays are perfectly aligning with the camera view

- **Image read and write** - for the reading, modification, and writing of the PPM files, both unit tests (`gtest`) and visual checks.



```
➜  test git:(main) ✗ ./run_tests
Running main() from /home/jhocking542/computer-graphics/raytracer/s2900180/Code/build/
deps/googletest-src/googletest/src/gtest_main.cc
[==========] Running 3 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 3 tests from PPMImageFileTest
[ RUN      ] PPMImageFileTest.CanReadPPM
[       OK ] PPMImageFileTest.CanReadPPM (0 ms)
[ RUN      ] PPMImageFileTest.CanUpdatePixel
[       OK ] PPMImageFileTest.CanUpdatePixel (0 ms)
[ RUN      ] PPMImageFileTest.CanWriteToFile
File deleted successfully.
[       OK ] PPMImageFileTest.CanWriteToFile (0 ms)
[----------] 3 tests from PPMImageFileTest (0 ms total)

[----------] Global test environment tear-down
[==========] 3 tests from 1 test suite ran. (0 ms total)
[  PASSED  ] 3 tests.
```

Figure 4: Terminal output showing the three unit tests working. In the future, a more extensive test suite will be created but these checked opening and reading the file, updating pixels, and writing to files.