CSC4140 Assignment VII

Computer Graphics

April 23, 2022

Ray Tracing II

This assignment is 8% of the total mark.

Strict Due Date: 11:59PM, May 1^{th}, 2022

Student ID:

Student Name:

This assignment represents my own work in accordance with University regulations.

Signature:

Overview

In this project, you will add additional features to your ray tracer from Assignment 7, such as more complicated materials, environment lights, or depth of field effects.

NOTE: For this project, you will choose ONE out of Part 1 or Part 2 to complete. In other words, you can choose any combination of THREE parts like the combination (xx, Part 3, Part 4).

However, if you'd like to complete all four parts, please do so! This project is one of the most rewarding to complete fully, as you'll be able to render almost anything at the end: all sorts of fancy materials in different environments, with any camera angle and focus depth.

- Mirror and Glass Materials (20 points)
- Microfacet Material (20 points)
- Environment Light (30 points)
- Depth of Field (30 points)

You will also need to read these articles:

- Experiments, Report and Deliverables

In particular, please read the Experiments, Report, and Deliverables page before beginning the project. There are many deliverables for this project, so please plan accordingly. Several parts of this assignment ask you to compare various methods/implementations, and we don't want you to be caught off guard!

Getting Started

Please consult the Assignment 6 spec for details on the user interface. Here are some old and new command line flags you'll want to pay attention to:

- *-e some_map.exr* loads an environment light *.exr* image file (part 3)
- *-b 1.23* sets the lens radius to 1.23 (part 4)

- *-d 4.56* sets the focal distance to 4.56 (part 4)

Please consider using the CGL Vectors Library to write all your math computations to simplify your code and ensure correctness.

A new mouse interaction: In render mode, after thin lens has been implemented, you can right click on a part of the image and automatically set the appropriate focal distance (autofocus). This is already done for people using pre-built binaries. For people using their own Assignment 6 code, you need to add this function to your *pathtracer.cpp*:

```
1 void PathTracer::autofocus(Vector2D loc) {
2   Ray r = camera->generate\_ray(loc.x / sampleBuffer.w, ...
3           loc.y / sampleBuffer.h);
4   Intersection isect;
5   bvh->intersect(r, \&isect);
6   camera->focalDistance = isect.t;
7 }
```

As well as two new keyboard commands:

- k | l: Increase/Decrease the camera lens radius
- $;$ | $'$: Increase/Decrease the focal distance

Also please use the "cell render" mode to debug, where you can interactively click and drag to select a small region of the screen to be rendered. Press c (after pressing r) to toggle this mode in the GUI.

Integrating Previous Code

For Assignment 7, you will need a working implentation of Assignment 6. You have the option of either porting your code from Assignment 6, or using pre-compiled binaries generated from the staff solution.

Using the Staff Library (Default settings, Recommended)

The project has been configured to default to this settings. If you just cloned the repository, you can directly start building & writing code. You can also use this to check / compare your own Assignment 6 solution.

To use the provided library, set the build option *BUILD_CUSTOM* to *OFF (Default)* in *CMakeLists.txt*. This will tell the generated makefiles to ignore your Assignment 6 code, and

include the library that matches your system instead. Note that the staff binary will generate a watermark on the corner of your image.

Using Your Own Code

To use your own code, set the build option *BUILD_CUSTOM* to *ON* in *CMakeLists.txt*. Then, you'll need to copy over the following files from your previous assignment, and place them in your src/ directory.

- src/scene/sphere.cpp
- src/scene/triangle.cpp
- src/scene/light.cpp
- src/scene/bvh.cpp
- src/scene/bbox.cpp
- src/pathtracer/camera.cpp
- src/pathtracer/bsdf.cpp
- src/pathtracer/pathtracer.cpp

Additionally, you will need to delete all the functions under Assignment 7 code in your *bsdf.cpp*. This includes functions:

- Vector3D MirrorBSDF::f(const Vector3D &wo, const Vector3D &wi)
- Vector3D MirrorBSDF::sample_f(const Vector3D &wo, Vector3D *wi, float *pdf)
- Vector3D GlossyBSDF::f(const Vector3D &wo, const Vector3D &wi)
- Vector3D GlossyBSDF::sample_f(const Vector3D &wo, Vector3D *wi, float *pdf)
- Vector3D RefractionBSDF::f(const Vector3D &wo, const Vector3D &wi)
- Vector3D RefractionBSDF::sample_f(const Vector3D &wo, Vector3D *wi, float *pdf)
- Vector3D GlassBSDF::f(const Vector3D &wo, const Vector3D &wi)
- Vector3D GlassBSDF::sample_f(const Vector3D &wo, Vector3D *wi, float *pdf)
- void BSDF::reflect(const Vector3D &wo, Vector3D *wi)
- bool BSDF::refract(const Vector3D &wo, Vector3D *wi, float ior)

Details

Do not convert or resize your *.png.* files. Use either the command line rendering mode or the $\boxed{S}$ key to save screenshots, not your own OS's utility (like Assignment 6). Keep your images in a subdirectory called images/ in the docs/ directory. We recommend using the -r 480 360 command line flag to set resolution at 480 by 360 for your screenshots.

Please also make sure that your writeup indicates clearly which two (or more!) parts you chose.

Final Note

As mentioned earlier, each of the parts of this project gives you something to work with, and if you finish all of Assignment 7, you can basically render anything. When you're done, feel free to show it off! Make some fancy scenes and materials, or play with the lens! Again, be creative