WEBGL Deferred Renderer and Scene Graph

Amartya Vadlamani

1 Background

Deferred Rendering is a method used to decouple the lighting complexity of a scene from geometric complexity of the same scene. It accomplishes this by first rendering surface information such as the diffuse color, roughness, normal and so on into a geometry buffer.

It then uses this buffer to perform it's lighting calculations.

In this project those lighting calculations are physically based on the Cook-Torrance lighting model. A microfacet BSDF that aims to be more physically plausible than the standard blinn-phong. This not only results in more accurate lighting but also more consistent shading across different lighting setups.

Deferred Rendering however does not take into consideration direct shadows. These are usually performed by rendering the depth of a scene from the light's perspective and then using this light perspective depth to perform a test to see if a fragment is lit or not. This does have the problem that there is no way to filter the shadow map as it doesn't make sense to perform bilinear filtering or downsampling on depth values. You can get around this by storing the depth and depth squared in a two-channel texture and using the ensuing probability distribution to calculate the odds that your fragment is in shadow.

One thing deferred rendering does not accomplish well is the rendering of transparent objects. This is because each fragment can only contain a single value and so can only encode a single surface point. One can get around this by rendering transparent objects in layers, each layer "peeling" off a new depth value behind the current one. This does have an issue in than you now need an arbitrary number of extra buffers in order to render most scenes.

2 Accomplishments

This project can load scene files that encode arbitrary scene graphs. These graphs can contain "OBJ" mesh files with custom PBR materials, lights and cameras.

The objects are rendered using deferred rendering, lit using the Cook-Torrance shading model, and shadowed using variational shadowmapping.

Depth peeling was not implemented due to time constraints.

3 Artifacts

The project is in the form of a webgl application what must be hosted using a webserver that serves index.html in the "src/" folder at the "/".

This can be accomplished by navigating to the "src/" directory and typing "python -m http.server 8000" or "python3 -m http.server 8000" if operating on the lab machines and then navigating to "localhost:8000".

You can then select one of the two example scenes using the interface and load the scene. If this fails, it is due to a bug in the loading code that causes it to start rendering before the load is complete. This can be fixed by waiting approximately 20 seconds and pressing the "Load Scene" button again.

The default scenes contain a sub-surfaced version of the Blender monkey, Suzanne, loaded from an obj file. The materials on demonstration are, rusted copper, painted steel and cardboard.

The scene files are hand-editable and new meshes, textures, materials and scenes can be added to the project simply by putting them in the appropriate directory and adding a scene select in the "index.html" file.

The camera in the scene can also be rotated around the origin by using the arrow keys.

4 References

Deferred Rendering `http://www.codinglabs.net/tutorial_simple_def_rendering.aspx`

Cook-Torrance Broad Strokes `http://www.codinglabs.net/article_physically_based_rendering_cook_torrance.aspx`

Realtime Hard Shadows `https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping`

Variance Shadow Mapping `http://www.punkuser.net/vsm/vsm_paper.pdf`

Additional functions and source code used are attributed at point of usage.