CS450/550 Project Report:

Original Proposal Text:

Implementing Disney's Principled Shader

James Leflang

leflangj@oregonstate.edu

Proposed:

Create a working implementation of Walt Disney Animation Studios' Physically-Based Shading in OpenGL.

Intent:

Use this project as an opportunity to take all that we have learned in this course, combine it, and bring it in-line with OpenGL 3.3+ capabilities outside of the fixed-function structures to implement modern shader usage.

Background:

Physically-Based Shading was introduced at SIGGRAPH 2012 to describe the augments to RenderMan during the production of *Wreck-It Ralph*. Walt Disney Animation Studios identified deficiencies in the existing Physically-Based Rendering (referred further as "PBR") pipeline and implemented changes to remedy those deficiencies (see Burley 2012). Since the publication, this model for rendering realistic materials has been implemented as the Principled Shader in Blender, the Disney BRDF in Arnold, and numerous other commercially viable tools.

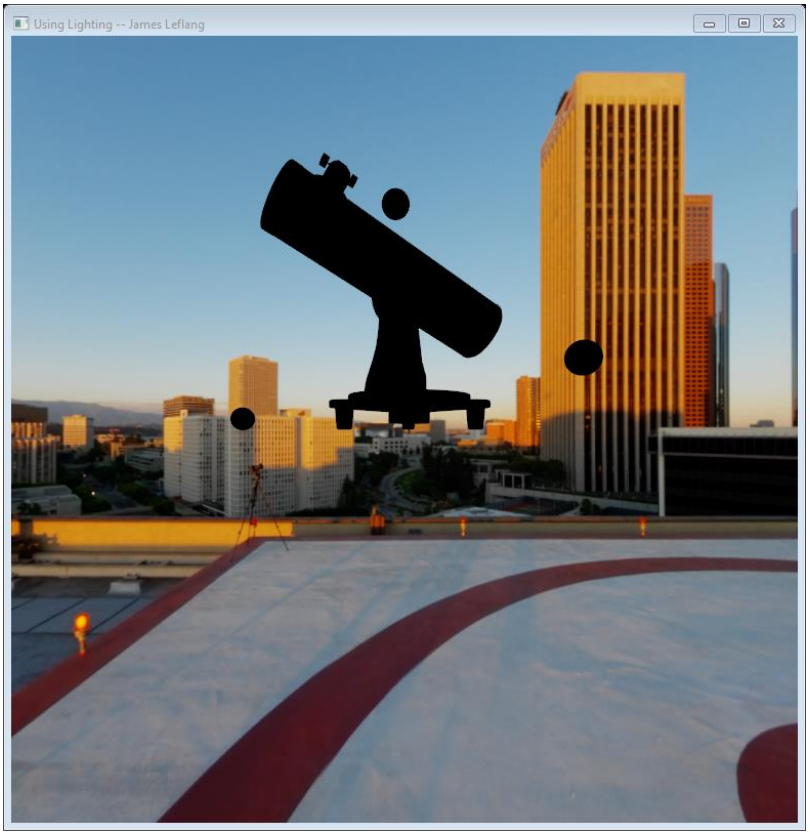
Roadmap:

- Learn and develop a basic PBR rendering engine.
  This has already been started. There are plenty of tutorials on how to implement PBR shaders in OpenGL, abet not using FREEGLUT thus requiring some trial & error to have one-to-one functionality.
- Find and configure suitable models to load.
  This has already been started. This is tricky as I converted loadobjfile.cpp to work with Vertex Attribute Objects, which is a necessary modification to the file for further work. I am now seeking a suitable model to showcase work completed.
- Modify all existing work to match the Principled Shader that has been implemented in Blender.
  This would require the most amount of work as Blender uses the Open Shading Language over GLSL or HLSL. OSL is closest to GLSL so it may not be insurmountable to implement in a short window of time.

Required Third Party Libraries (outside those that we have already used):

- Stb_image.h: this is primarily to load *.hdr files for the environment cubemaps.

Actual Implementation:



Due to external time constraints (unforeseen life commitments that meant I could not spend the time required nor are justifiable for dropping the course), I could not accomplish what I proposed nor do have the textures correctly display. I primarily implemented a MTL parser to load the material textures that were associated in the OBJ file; this became much of my own work and thus is what should be considered as my project. I borrowed shaders that made a cubemap from the learnopengl.com Tutorial on IBL (not the primary focus), and modified the code from the learnopengl.com Tutorial on PBR Lighting (was intended to be the basis for learning PBR to eventually modify into Disney's Principled shader) to work with the provided Shadows shader code to render PBR-ready OBJ's. I added to the PBR shaders an ACES Filmic tonemap (modified to be faster than the WebGL version) to better render the colors. The implementation works on OpenGL 4.5; this project uses features and concepts that were introduced in version 4.2. **Note: for some reason I could not diagnose the reason for the textures being rendered black with the time that I had available. (Steps taken to debug: checked vertex normals and normal maps, changed the shaders by flipping signs, verified that I loaded the texture images correctly into memory using stb_images.h, and ensuring that the textures are loaded into GPU correctly.)**

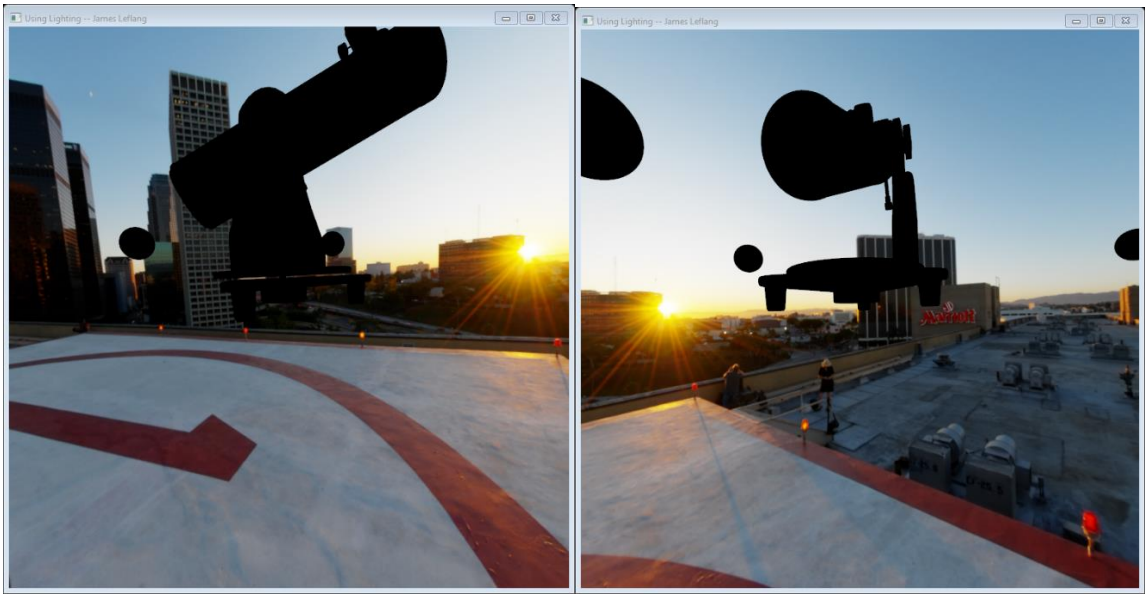
Differences from Proposal:

- **Not Disney's Principled Shader**: This is a disappointment since this was the core of the proposal. I could only implement a basic PBR shader that is borrowed code. I did do some modifications to better understand the borrowed code and to match the lecture material's naming of variables.
- **Most of my own code is in the implementation of the MTL parser and modification of the provided OBJ loader**: This became the bulk of the development time to accurately load the MTL file and associate the texture images with the right objects.

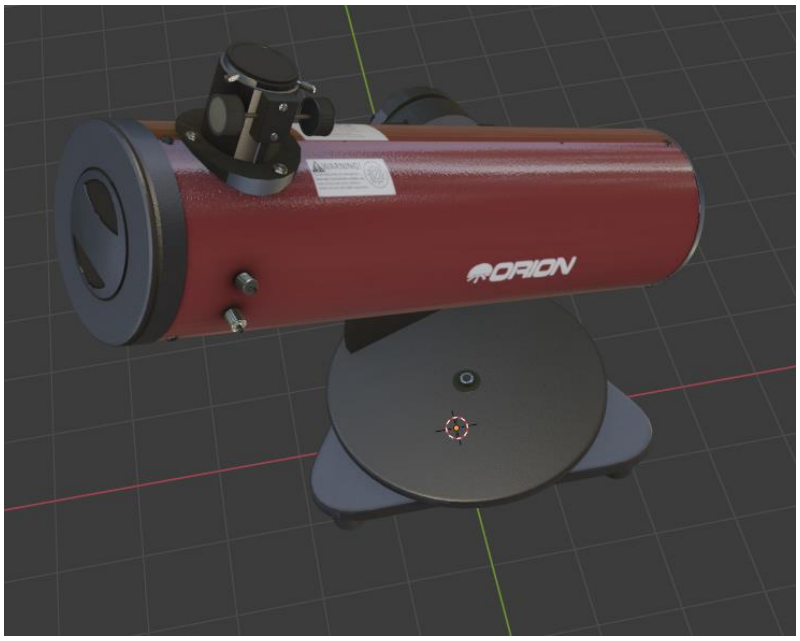
Learnings:

- Implementing PBR pipelines is hard:
  Most of the time is spent understanding how modern OpenGL handles textures and how fragment shaders work with per fragment normals. I set too high of an entry bar for the time that I had available to devote to implementing the most basic form; I should have heeded the warning that this was very hard to do.
- MTL files are very limited in functionality:
  Since the inception of the Wavefront OBJ and MTL formats, they have changed very little and thus suffer from limitations as software and hardware progressed. MTL files specially suffer due to the more complex number of variables that can be modified. There are extensions to the format to compensate, such as the "norm" parameter to denote a Normal texture, but those extensions may or may not be standard across all programs.
- Finding models for PBR is not hard, but insuring they are suitable for this level of project is tricky:
  Many models available online are suited for commercial software (Maya, 3ds Max, mostly Autodesk-made software) and thus make it very unsuitable as they don't convert to OBJ files accurately (majority of the time I found they incorrectly mapped normals upon conversion.)

Images:



What the model is supposed to be (from Blender):



Rendered:

Link to Video:

https://media.oregonstate.edu/media/t/1_6vkx7z2z