DigiPen
INSTITUTE OF TECHNOLOGY

CS 350 |

Assignment 1| Deferred + Forward Shading

Files (submit archive) due

- Week 4
- By midnight

Remember: The major point of this assignment is for you to learn how to submit your homework correctly. You should strive to follow all of the directions exactly as specified in the handouts. The syllabus that was handed out during the first day of class also contains important information on how to submit your homework. If you are still unsure of how to do something, you should ask for help, either from myself or from a tutor in the lab or from another student.

This assignment focuses on building a Deferred + Forward Shading framework to be used in CS 350.

1. The rendering algorithm that we have implemented so far falls under the so-called "Forward Rendering" algorithm as we implement the rendering in **one pass** over the geometry. The goal of this assignment is to implement a **hybrid rendering pipeline** that employs a Deferred Rendering method for the objects of interest, coupled with the usual Forward Rendering for auxiliary rendering (debug draw, vertex and face normals etc.)

2. The rendering algorithm can be specified as below as pseudo-code

```
Render()
{
      // "FBO1" is user defined FBO
      Render_Deferred_Objects(FBO1);

      // Important: Do not forget this
      // Copy FBO1's depth buffer into default
      // OpenGL depth buffer

      if( bCopyDepthInfo ) // For debug purposes
            Copy_Depth_Info( FBO1, Default FBO );

      // Forward Rendering to draw simple objects
      Render_Debug_Objects();
}
```

Render_Deferred_Objects()

a. Implement the Deferred Shading pipeline for rendering the OBJ files using FBO's (let's call this FBO, FBO1.)
b. **First pass:** Generate the G-buffer using multiple render targets (Refer CS 300 Assignment on Dynamic Reflection & Refraction)
c. The render targets will hold the information to implement Phong Shading in the second (Lighting) pass.
d. **Second Pass:** Use the render targets from Pass 1 as input textures to the Lighting Pass. The Vertex Shader will be a straight pass-through to render a Full Screen Quad (FSQ). The Fragment Shader will implement the Phong Shading, but reading the input values for material / environment properties from the textures.

e. Use Uniform Blocks (as before) to pass light information.
f. At the end of the Lighting Pass, the default FBO will show the color information from the objects, but the default depth buffer will be a constant (**Q: Why?**)

Copy_Depth_Info()

a. IMPORTANT – Copy over the depth buffer information from FBO1 into the default depth buffer (**Q: What is the default FBO in OpenGL?**)
b. Toggle this capability with input from user (GUI). Your output **should look incorrect** if the depth information is not copied over.

Render_Debug_Objects()

a. Now draw the face normal and vertex normal using the Forward Rendering method. Essentially, your code to render these objects should be unchanged from CS 300.
b. Do not apply Phong Shading to debug draw objects.
c. In future assignments, we will expand our repertoire of "debug" objects to include bounding volumes, trees, and rays.

Scene Setup:

- The scene setup will involve two sets of objects – deferred rendered and forward rendered
    - **Deferred Object(s):** Load objects corresponding to the power plant sections and place the camera in the center of this object. You may use your CS 300 OBJ loader to draw the objects, or use a 3rd-party library (such as Assimp (https://www.assimp.org/). You can find a very useful tutorial for integrating Assimp into your projects here: https://learnopengl.com/Model-Loading/Assimp.)
    - **Forward-rendered Objects**: Vertex and face normals will be rendered using forward rendering. Use GL_LINES to draw these objects
- Render the models using Phong Illumination Model (from CS300). As a reference, you can use the CS 300 Assignment 2 as a starting point and use the Phong Shading method. It is recommended to **remove Blinn Shading, Phong Lighting, and Environment Mapping from your implementation.**
- Create a first-person person camera (observer) (from CS250)

- Ability to move the camera around (use standard WASD key bindings)

Suggested Weekly breakdown:

- **Week 1:** Refactor CS 300 code to separate rendering for OBJs and Debug Objects into two separate functions. Refer pseudo code from handout for solution. Integrate a WASD camera (from CS 250) for looking around the scene.
- **Week 2:** Implement Deferred Rendering for OBJs. Turn off debug draw. It is a good idea to visualize the individual render targets as final color on the FSQ.
- **Week 3:** Combine Deferred output with debug draw output.
- **Week 4:** Debugging and making your code submission-ready.

Please refer to your course syllabus for submission guidelines.

Please refer to the references on Moodle for details about implementation.

GRADING SHEET

Implementation Point	Grade	Points obtained	Comments
Scene generation	**10%**		
Objects loaded correctly from the files and displayed in proper position	10%		
Deferred Shading	**50%**		
G-buffer populated correctly	30%		
Lighting pass setup for FSQ	10%		
Implementation of lighting pass with Phong Shading	10%		
Forward Shading	**10%**		
Rendering vertex/face normals using forward rendering	10%		
Interactivity	**20%**		
Camera movement (WASD)	5%		
GUI option to toggle depth copying	5%		
Ability to visualize individual render targets on FSQ	10%		
Miscellaneous issues	**10%**		**Automatic zero grade on assignment if application does**
Missing information in README			
Application does not compile			
Application cannot be executed			

Scene setup incorrect		**not satisfy these requirements.**
Total	100%	