Einführung in Visual Computing

Physically Based Rendering with Image Based Lighting in OpenGL

Summary

This project demonstrates a comprehensive setup for a physically based rendering (PBR) pipeline with IBL and Dynamic Lighting using OpenGL. It includes the initialization of essential libraries (GLFW, GLAD, GLM), shader compilation, model and texture loading, camera and input handling, lighting setup, and HDR to cubemap conversion. The project showcases the use of modern OpenGL techniques to achieve realistic rendering of 3D models with advanced lighting and material properties.

Objectives Achieved

- **Camera Implementation:** The Camera class was implemented to simulate an orbiting camera in OpenGL. It was designed to orbit around a target point, manage orientation using quaternions (also to avoid GImbal lock), calculate the view matrix, and process mouse input for rotation and zoom.
- **Physically Based Rendering (PBR):** PBR was integrated to achieve realistic material and lighting effects. The PBR system used multiple textures - albedo, metallic, roughness, normal, and ambient occlusion mapping to define the surface properties of models. The PBR shader leveraged precomputed environment maps, including irradiance and prefiltered maps, to accurately simulate complex lighting interactions and reflections..
- **Image-Based Lighting (IBL):** IBL was implemented to enhance scene realism by simulating complex lighting interactions. This involved converting HDR images into cubemaps, generating irradiance maps for diffuse lighting, and creating prefiltered maps for reflections. Three distinct HDR setups were used to provide diverse lighting conditions, ensuring more accurate and dynamic environmental effects.

Shaders utilized these precomputed textures for realistic lighting in various scenarios.

- **Depth Testing and Seamless Cubemap Sampling:** Depth testing along with seamless cubemap sampling ensured correct rendering of 3D objects and better quality while sampling from cubemaps.
- **Dynamic Lighting:** The project integrated moving lights in conjunction with Image-Based Lighting (IBL) to enhance the realism and depth of the rendered scene. By incorporating dynamic lighting, the system simulates real-world lighting variations, such as changing light sources and their movement over time, which adds complexity.

Problems Encountered

- The steep learning curve of OpenGL itself was a significant challenge. Understanding OpenGL's extensive documentation and debugging its low-level graphics programming aspects demanded considerable time and effort.
- Shader compilation and linking errors were frequent and challenging to debug. Some were extremely difficult and even took days to make it work.
- Properly loading models and textures proved difficult, mainly due to issues with file formats, compatibility, and the use of external libraries.
- Setting up the camera system was difficult, with problems related to incorrect view and projection matrices, and issues with camera movement and controls.
- Precomputing values for multiple HDR maps was challenging due to difficulties in ensuring seamless transitions between environment maps. Additionally, managing and utilizing buffers for HDR maps added complexity to the process.

Source Code

A large portion of the source code was derived from Learn OpenGL, a comprehensive tutorial resource that covers modern OpenGL practices, including shaders and PBR (Physically Based Rendering) with IBL (Image-Based Lighting). Custom headers and starter code were also utilized. Additional support came from YouTube channels such as OpenGL

Tutorials - YouTube and Cherno - YouTube, which provided valuable guidance. Websites like How to program an arcball (orbiting) camera in C++ and OpenGL by NerdHut, and discussions on the Khronos Forums, helped in successfully implementing the arcball camera system.

Instructions

The IBL (Image-Based Lighting) environments can be switched by pressing the numerals 1, 2, and 3 (*numerals below the function keys*). The scene can be navigated using the arcball camera: moving the mouse adjusts the camera's orientation, left-clicking and dragging changes the camera's position, and the scroll wheel alters the distance to the model. Camera properties such as sensitivity, speed, maximum radius, minimum radius etc can be adjusted in the camera class. The main file contains commented-out code for loading a backpack model; uncommenting this code and commenting out the current model (DamagedHelmet) will render the backpack instead. Also note that the moving lights are not adjustable.