CS 550 | Fall 2023

**Project 3**

## “Lighting"

By

**Pushpak Katkhede**

Email – [katkhedp@oregonstate.edu](mailto:katkhedp@oregonstate.edu)

**OUSID – 934453040**

Instructor – Prof Mike Bailey

* **Description 🡪**

**Video Link:** [**Click Here**](https://media.oregonstate.edu/media/t/1_6gtadv12)

**Display Lists:**

We created Display display lists using TeapotDL, PawnDL, DogDL, and GridDL Display Lists: The function begins by creating four separate display lists for rendering 3D models: a teapot, a pawn chess piece, a dog, and a grid. These models are loaded from external files (e.g., "teapot.obj," "pawn.obj," etc.) using the LoadObjFile function. The display lists are created using glGenLists, allowing for efficient rendering of these objects in the future.

**Keyboard Specs:**

* Key Input Handling: When a key is pressed, the function receives the corresponding character. It checks for specific key presses and responds accordingly.
* Changing Projection: The 'O' key switches the projection mode between orthographic and perspective. This affects how the scene is viewed.
* Light Source Selection: The 'P' and 'S' keys allow the user to switch between a point light source and a spot light source. This can alter how light interacts with the objects in the scene.
* Changing Object Color: The 'W', 'R', 'G', 'B', and 'Y' keys change the color of the light source. This is accomplished by selecting a color from a predefined set.
* Quitting the Application: The 'Q' key and the 'Escape' key both initiate the quit sequence, closing the application gracefully.

**Displaying:**

We started our application by clearing the framebuffer and using the glClear function along with our predefined background color to erase the previous frame. After that, we set up the view and projection matrices. The projection matrix chose how 3D coordinates were projected onto the 2D screen, while the view matrix was dependent on the position of the camera. The projection matrix was configurable. Next, using user interaction and animations as a guide, we applied model transformations like rotations and scaling. Lastly, we used glCallList to efficiently render 3D objects from display lists, such as a teapot, pawn, dog, and grid. Our interactive 3D application was built on these features together, which made for a dynamic and visually appealing user experience.

* **Screenshots 🡪**

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**