

CS 550 | Fall 2023

Project 3

“Lighting”

By

Pushpak Katkhede

Email – katkhedp@oregonstate.edu

OUSID – 934453040

Instructor – Prof Mike Bailey

❖ Description →

Video Link: [Click Here](#)

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.

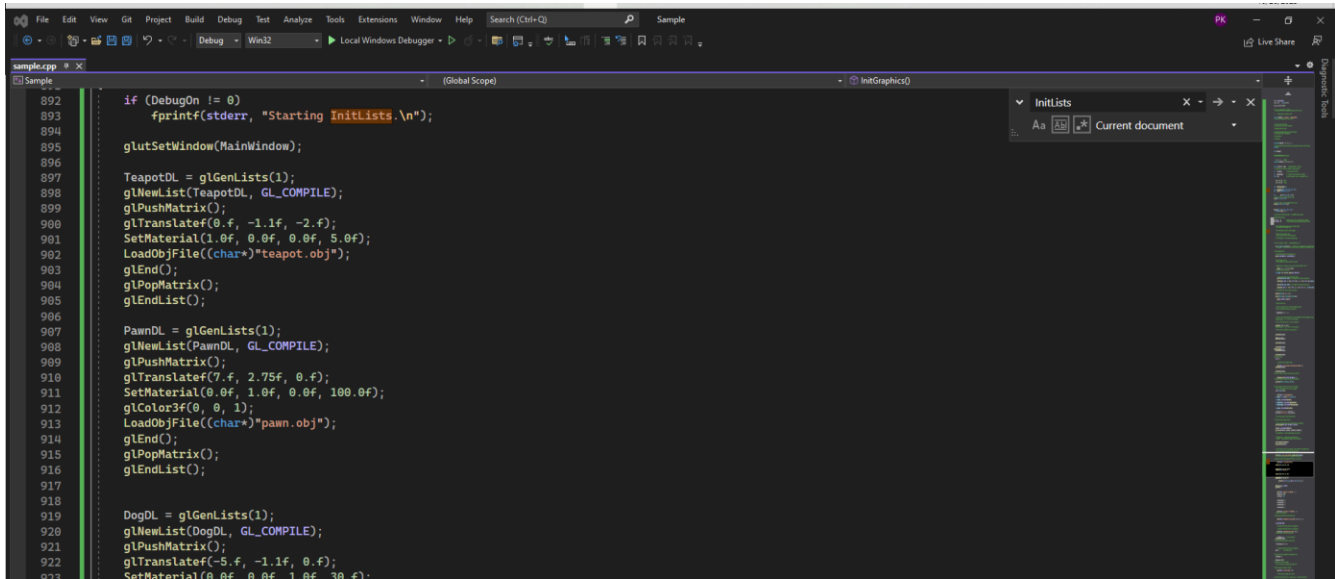
Keyboard Specs:

- **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.

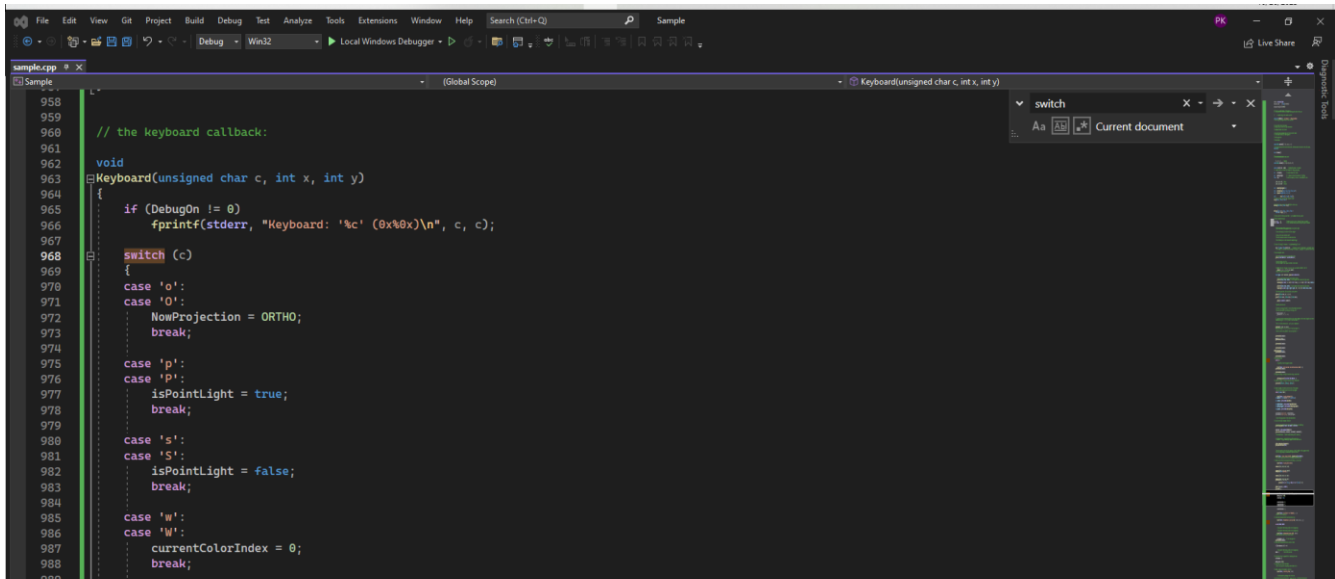
❖ Screenshots →



```
892     if (DebugOn != 0)
893         fprintf(stderr, "Starting InitLists.\n");
894
895     glutSetWindow(MainWindow);
896
897     TeapotDL = glGenLists(1);
898     glNewList(TeapotDL, GL_COMPILE);
899     glPushMatrix();
900     glTranslatef(0.f, -1.1f, -2.f);
901     SetMaterial(1.0f, 0.0f, 0.0f, 5.0f);
902     LoadObjFile((char*)"teapot.obj");
903     glEnd();
904     glPopMatrix();
905     glEndList();
906
907     PawnDL = glGenLists(1);
908     glNewList(PawnDL, GL_COMPILE);
909     glPushMatrix();
910     glTranslatef(7.f, 2.75f, 0.f);
911     SetMaterial(0.0f, 1.0f, 0.0f, 100.0f);
912     glColor3f(0, 0, 1);
913     LoadObjFile((char*)"pawn.obj");
914     glEnd();
915     glPopMatrix();
916     glEndList();
917
918     DogDL = glGenLists(1);
919     glNewList(DogDL, GL_COMPILE);
920     glPushMatrix();
921     glTranslatef(-5.f, -1.1f, 0.f);
922     SetMaterial(0.0f, 0.0f, 1.0f, 30.f);
923     LoadObjFile((char*)"dog.obj");
924     glEnd();
925     glPopMatrix();
926     glEndList();
```

InitLists

Aa .*. Current document



```
958
959
960 // the keyboard callback:
961
962 void
963 Keyboard(unsigned char c, int x, int y)
964 {
965     if (DebugOn != 0)
966         fprintf(stderr, "Keyboard: '%c' (0x%08x)\n", c, c);
967
968     switch (c)
969     {
970     case 'o':
971     case 'O':
972         NowProjection = ORTHO;
973         break;
974
975     case 'p':
976     case 'P':
977         isPointLight = true;
978         break;
979
980     case 's':
981     case 'S':
982         isPointLight = false;
983         break;
984
985     case 'w':
986     case 'W':
987         currentColorIndex = 0;
988         break;
989     }
```

Keyboard(unsigned char c, int x, int y)

switch

Aa .*. Current document

