Department of Computer Science and Engineering

Shivanirudh S G, 185001146, Semester VII

12 October 2021

UCS1712 - Graphics and Multimedia Lab

Exercise 8: 3-Dimensional Projections in C++ using OpenGL

Objective:

Write a menu driven program to perform Orthographic parallel projection and Perspective projection on any 3D object.

Code:

```
#ifndef LOPENGL_H
#define LOPENGL_H

#include <GL/freeglut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <math.h>
#include <stdio.h>
#include<iostream>
#include<vector>
#include<ctime>
#include <unistd.h>
using namespace std;

#endif
```

```h
#ifndef LUTIL_H
#define LUTIL_H

#include "Headers.h"

//Screen Constants
const int SCREEN_WIDTH = 640;
const int SCREEN_HEIGHT = 480;
const int SCREEN_FPS = 60;
const int POINT_SIZE=2;

const double X_MIN = -500;
const double X_MAX = 500;
const double Y_MIN = -500;
const double Y_MAX = 500;

double x_rotate = 0;
double y_rotate = 0;

bool isOrthoProjection = true;

void render();

void keyboardKeys(unsigned char key, int x, int y);

void drawAxes();


#endif
```

```h
#include "Signatures.h"



void render(){
    glClearColor(0, 0, 0, 1);
    glClear(GL_COLOR_BUFFER_BIT);

    //Translucency
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    //Line width
    glLineWidth(3);

    //Apply the transformations & drawing on the model view matrix
    glMatrixMode(GL_MODELVIEW);

    //Draw the X and Y axis
    drawAxes();

    //Transform only the drawn object, so use the matrix stack
    accordingly
    glPushMatrix();

    if(isOrthoProjection){
        //Parallel Projection
        glOrtho(-2, 2, -2, 2, -2, 2);
```

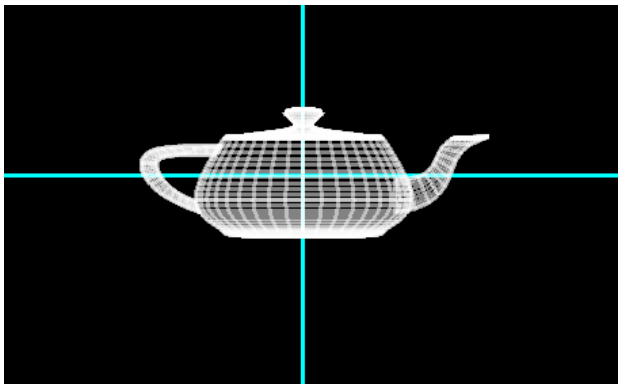
```
28      } else{
29          //Perspective Projection
30          gluPerspective(120, 1, 0.1, 50); //FoVy = 120, Aspect Ratio
         = 1
31      }
32
33      gluLookAt(0, 0, 1, 0, 0, 0, 0, 1, 0);   //Camera, Center & Up
        Vector
34      glRotatef(x_rotate, 1, 0, 0);   //Keyboard based rotations
35      glRotatef(y_rotate, 0, 1, 0);
36
37      glColor4f(1, 1, 1, 0.3);    //Draw the object
38      glutWireTeapot(0.5);
39
40      glPopMatrix();  //Pop the matrix back into the model view stack
41
42      glFlush();
43  }
44
45  void drawAxes(){
46      //To draw X and Y axis
47
48      glColor3d(0, 1, 1);
49
50      glBegin(GL_LINES);
51
52      glVertex2f(-2, 0);
53      glVertex2f(2, 0);
54
55      glVertex2f(0, -2);
56      glVertex2f(0, 2);
57
58      glEnd();
59      glFlush();
60  }
61
62  void keyboardKeys(unsigned char key, int x, int y){
63      //Callback function for keyboard interactivity
64
65      key = tolower(key);
66
67      switch(key){
68          case 'w':{
69              x_rotate += 5;
70              break;
71          }
72          case 's':{
73              x_rotate -= 5;
74              break;
75          }
76          case 'd':{
77              y_rotate += 5;
78              break;
79          }
80          case 'a':{
81              y_rotate -= 5;
82              break;
```

```
        }
        case 32:{
            //Spacebar for changing projections
            isOrthoProjection = !isOrthoProjection;
            break;
        }
    }

    //Update the display
    glutPostRedisplay();
}
```

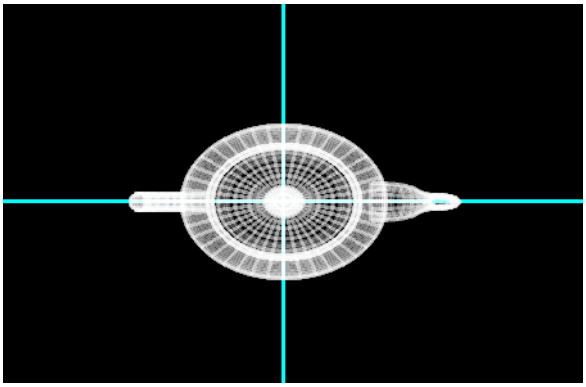
```
#include "Helpers.h"

int main( int argc, char* args[] ){

    glutInit( &argc, args );

    glutInitContextVersion( 2, 1 );

    glutInitDisplayMode( GLUT_SINGLE|GLUT_RGB );
    glutInitWindowSize( SCREEN_WIDTH, SCREEN_HEIGHT );
    glutCreateWindow( "OpenGL" );

    glutDisplayFunc(render);
    glutKeyboardFunc(keyboardKeys);

    //Change to projection mode before applying glOrtho()/
    gluPerspective()
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    glutMainLoop();

    return 0;
}
```

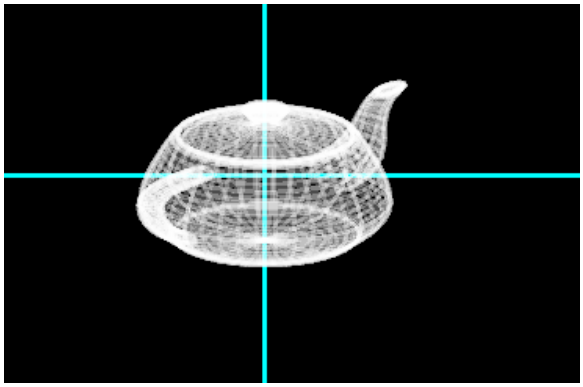
Output:

Orthographic projection:

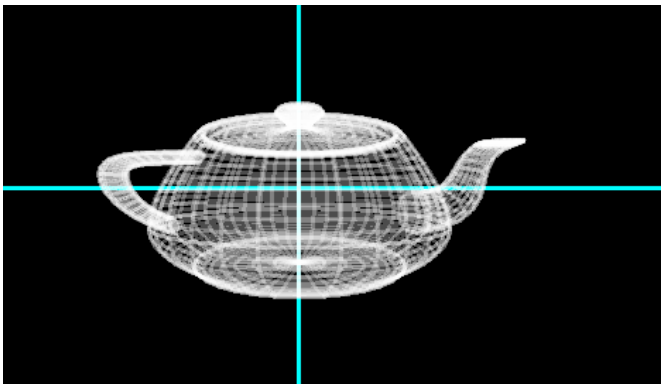


Keyboard operations:





Perspective projection:



Keyboard operations: