Department of CSE
SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VII

17 October 2021

UCS 1712 - Graphics And Multimedia Lab

Exercise 9: 3-Dimensional Projections in C++ using OpenGL

Aim:

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

Set the camera to any position on the 3D space. Have $(0, 0, 0)$ at the center of the screen. Draw X, Y and Z axis. You can use gluPerspective() to perform perspective projection.

Use keyboard functions to rotate and show different views of the object.

Note: Can use built-in functions for 3D transformations.

Code: 3D Projections:

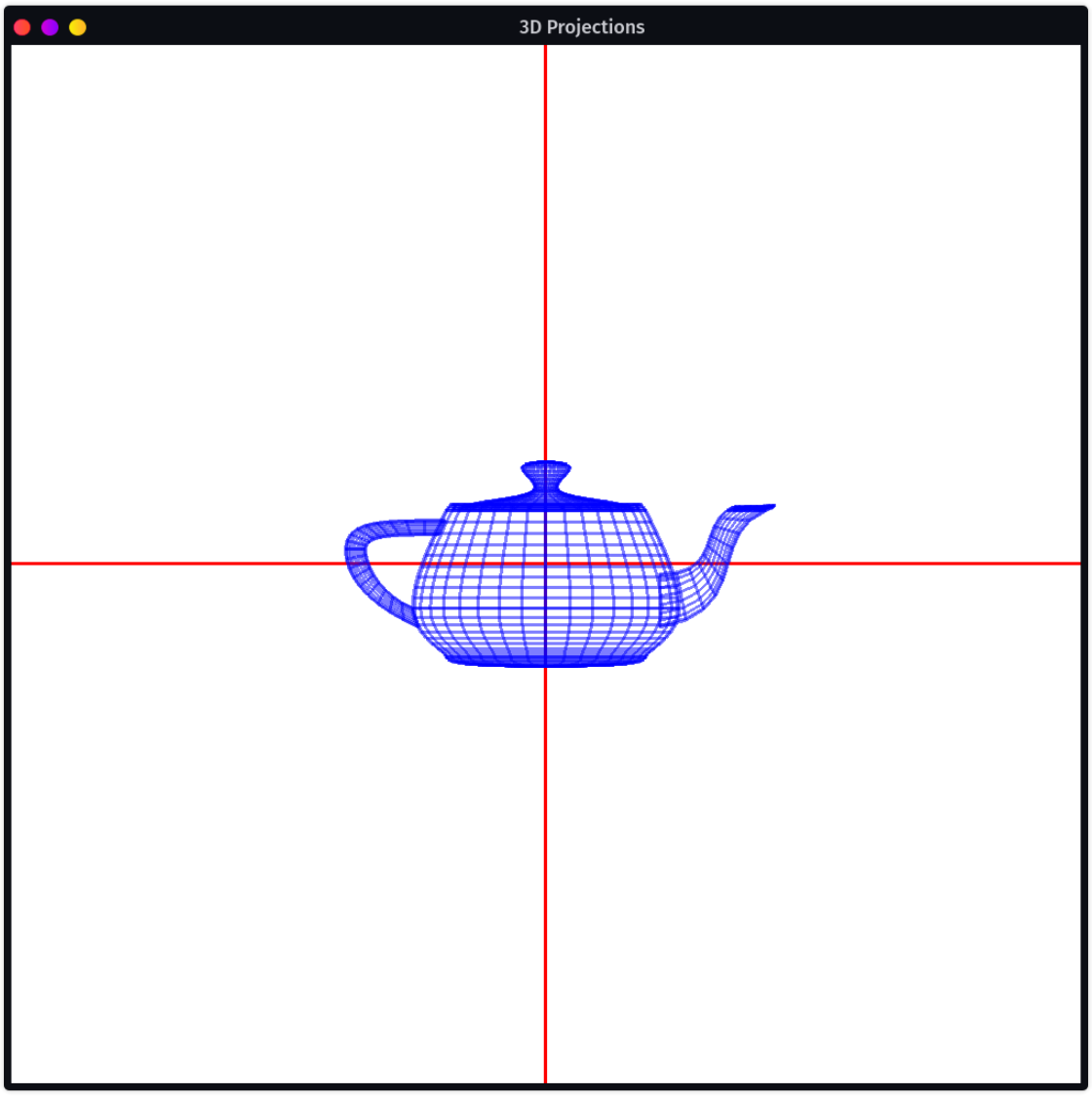
```
/*
 To demonstrate Orthographic Parallel and Perspective Projection using
    OpenGL
 and to also use keyboard functions and show different object views, along
     with
 setting the camera position.
*/

#include<iostream>
#include<cstring>
#include<GL/glut.h>
#include<math.h>

using namespace std;

//Global constants
const float WINDOW_WIDTH = 1000;
const float WINDOW_HEIGHT = 1000;
const float X_MIN = -500;
const float X_MAX = 500;
const float Y_MIN = -500;
const float Y_MAX = 500;
const int FPS = 60;

//Global variables to handle rotation
double x_rotate = 0;
double y_rotate = 0;

//Global variable for projection
bool isOrthoProjection = true;

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

int main(int argc, char **argv){

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    glutCreateWindow("3D Projections");

    //Register the callback functions
    glutDisplayFunc(initializeDisplay);
    glutKeyboardFunc(keyboardKeys);

    //Change to projection mode before applying glOrtho()/gluPerspective()
```

```
46      glMatrixMode(GL_PROJECTION);
47      glLoadIdentity();
48
49      glutMainLoop();
50
51      return 0;
52  }
53
54  void initializeDisplay(){
55      //Initialize display parameters
56
57      glClearColor(1, 1, 1, 1);
58      glClear(GL_COLOR_BUFFER_BIT);
59
60      //Translucency
61      glEnable(GL_BLEND);
62      glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
63
64      //Line width
65      glLineWidth(3);
66
67      //Apply the transformations & drawing on the model view matrix
68      glMatrixMode(GL_MODELVIEW);
69
70      //Draw the X and Y axis
71      drawAxes();
72
73      //Transform only the drawn object, so use the matrix stack accordingly
74      glPushMatrix();
75
76      if(isOrthoProjection){
77          //Parallel Projection
78          glOrtho(-2, 2, -2, 2, -2, 2);
79      } else{
80          //Perspective Projection
81          gluPerspective(120, 1, 0.1, 50); //FoVy = 120, Aspect Ratio = 1
82      }
83
84      gluLookAt(0, 0, 1, 0, 0, 0, 0, 1, 0);   //Camera, Center & Up Vector
85      glRotatef(x_rotate, 1, 0, 0);   //Keyboard based rotations
86      glRotatef(y_rotate, 0, 1, 0);
87
88      glColor4f(0, 0, 1, 0.3);    //Draw the object
89      glutWireTeapot(0.5);
90
91      glPopMatrix();  //Pop the matrix back into the model view stack
92
93      glFlush();
94  }
95
96  void drawAxes(){
```

```
97      //To draw X and Y axis
98
99      glColor3d(1, 0, 0);
100
101     glBegin(GL_LINES);
102
103     glVertex2f(-2, 0);
104     glVertex2f(2, 0);
105
106     glVertex2f(0, -2);
107     glVertex2f(0, 2);
108
109     glEnd();
110     glFlush();
111 }
112
113 void keyboardKeys(unsigned char key, int x, int y){
114     //Callback function for keyboard interactivity
115
116     key = tolower(key);
117
118     switch(key){
119         case 'w':{
120             x_rotate += 5;
121             break;
122         }
123         case 's':{
124             x_rotate -= 5;
125             break;
126         }
127         case 'd':{
128             y_rotate += 5;
129             break;
130         }
131         case 'a':{
132             y_rotate -= 5;
133             break;
134         }
135         case 32:{
136             //Spacebar for changing projections
137             isOrthoProjection = !isOrthoProjection;
138             break;
139         }
140     }
141
142     //Update the display
143     glutPostRedisplay();
144 }
```

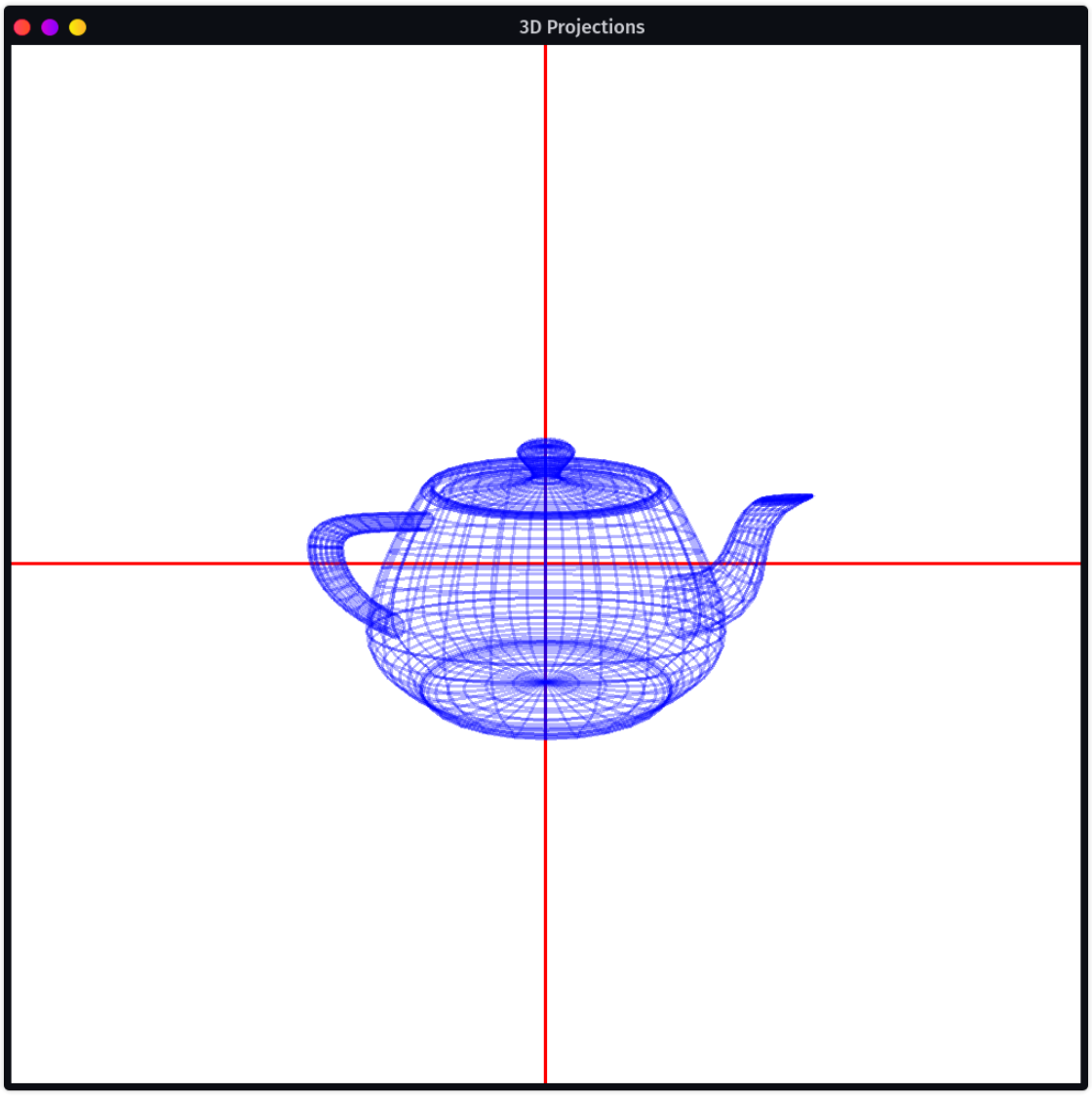
Output: 3D Object - Ortho

Figure 1: 3D Object - Ortho.



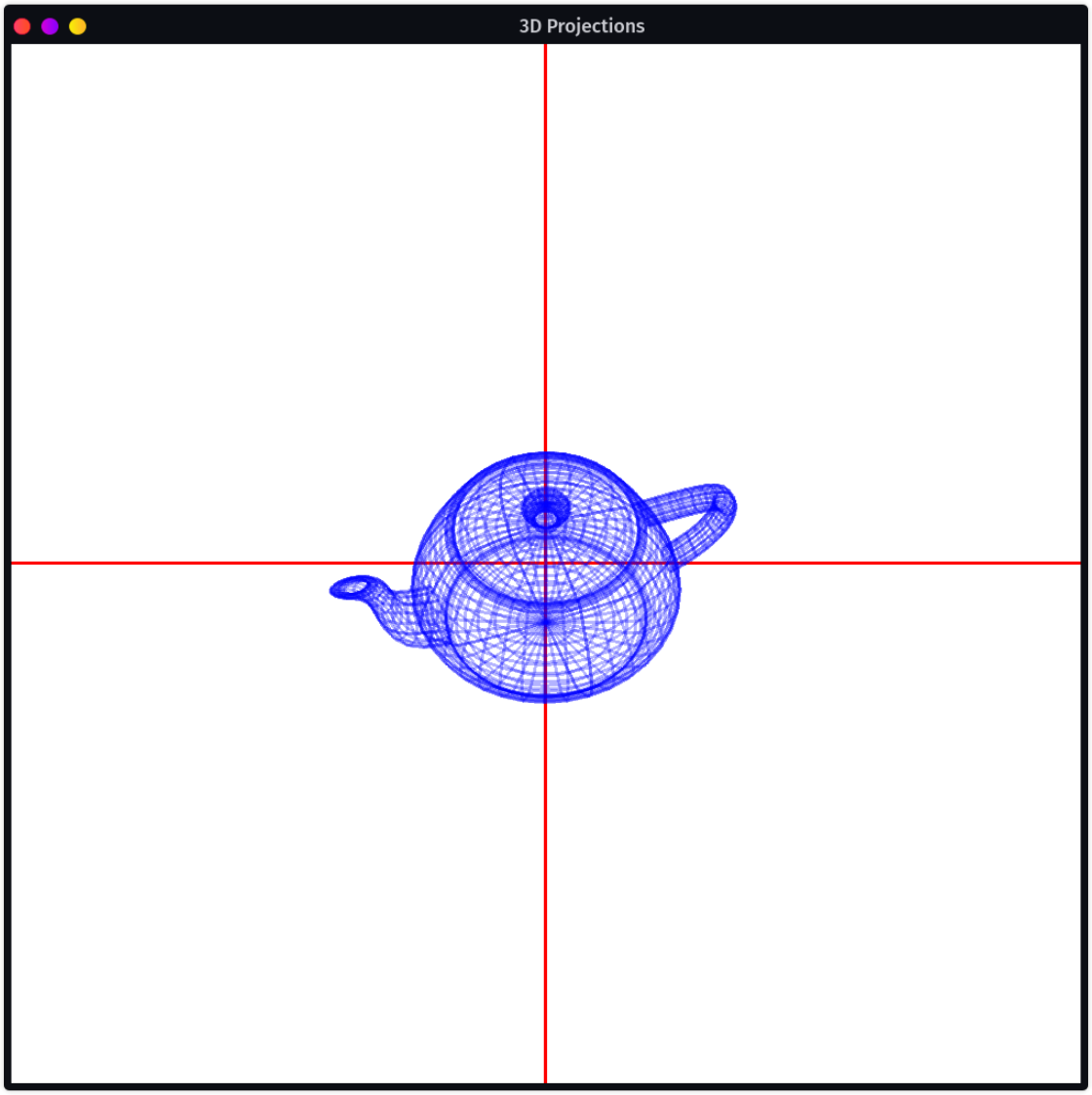
Output: 3D Object - Perspective

Figure 2: 3D Object - Perspective.



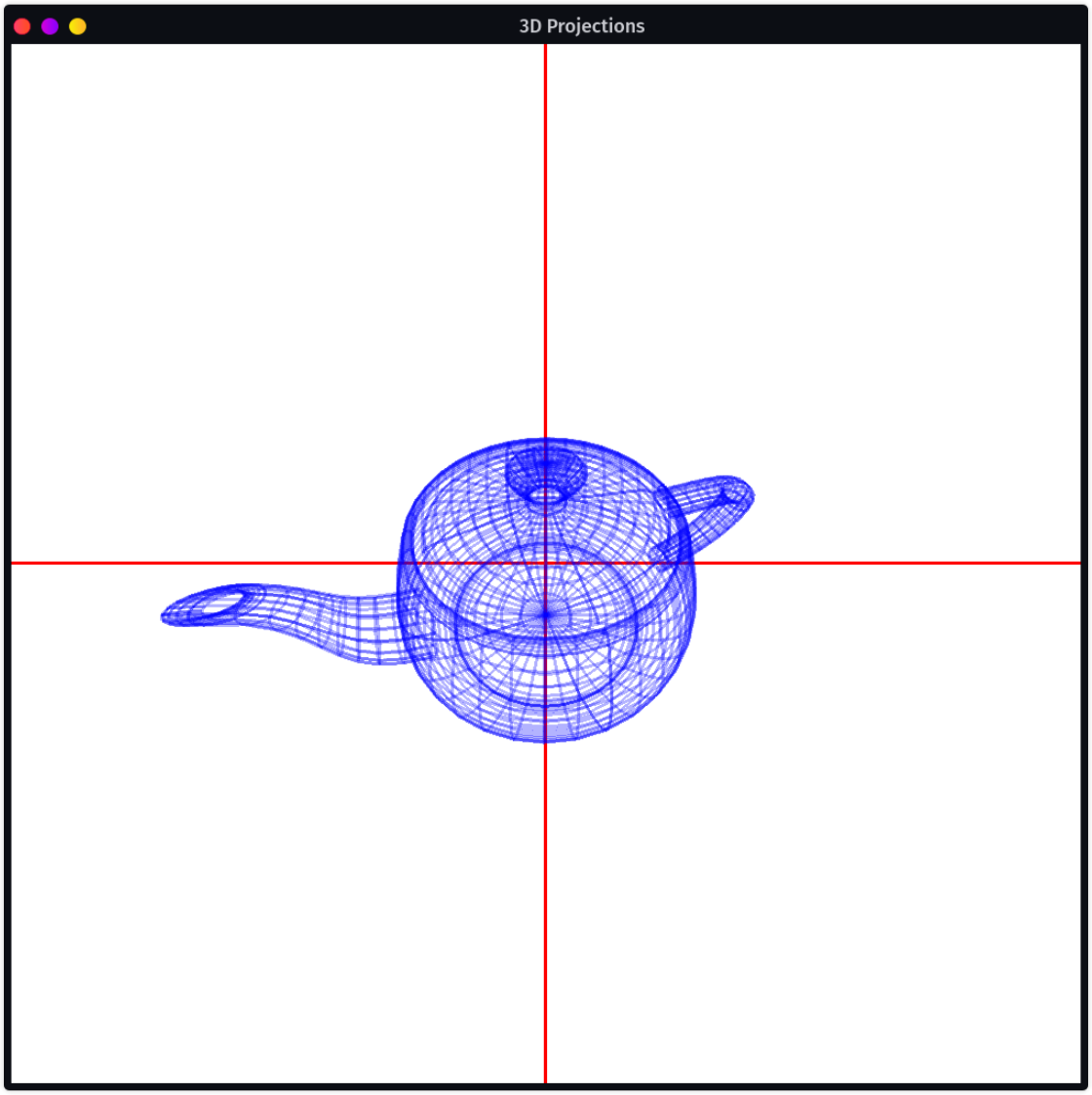
Output: 3D Object - Ortho (Rotated)

Figure 3: 3D Object - Ortho (Rotated).



Output: 3D Object - Perspective (Rotated)

Figure 4: 3D Object - Perspective (Rotated).



Learning Outcome:

- I learnt how to set-up **keyboard functions** for handling different user inputs for rotation & projections using **glutKeyboardFunc()** callback method.
- I learnt how to use inbuilt 3D transformation methods like **glRotatef() and glTranslatef()**.
- I understood the working of **glOrtho() and gluPerspective()** methods.
- I learnt how to set-up camera position using **gluLookAt()** method.
- I understood the usage of **glPushMatrix() & glPopMatrix()**.
- I understood the difference between the matrix modes of **GL_MODELVIEW & GL_PROJECTION**, and when to use which.
- I was able to use the inbuilt **glutWireTeapot()** method to display a 3-D Teapot object and performed parallel and perspective projections upon them, apart from rotating the object in the X and Y directions.
- I learnt about **field of view**, **aspect ratio** and how to properly configure **zNear & zFar** in gluPerspective().
- I learnt how orthographic and parallel projections differ from each other.
- I understood how the **camera, center & up vectors** are configured in gluLookAt().