Department of CSE
SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VII

27 October 2021

UCS 1712 - Graphics And Multimedia Lab

Exercise 10: Creating a 3D Scene in C++ using OpenGL

Aim:

Write a C++ program using OpenGl to draw atleast four 3D objects. Apply lighting and texture and render the scene. Apply transformations to create a simple 3D animation.

OpenGL Functions to use:

- glShadeModel()
- glMaterialfv()
- glLightfv()
- glEnable()
- glGenTextures()
- glTexEnvf()
- glBindTexture()
- glTexParameteri()
- glTexCoord2f()

Note: Use built-in transformation functions.

Code: 3D Scene:

```
 /*
Write a C++ program using Opengl to draw atleast four 3D objects. Apply lighting and texture and
render the scene. Apply transformations to create a simple 3D animation.
*/

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

const float WINDOW_WIDTH = 800;
const float WINDOW_HEIGHT = 800;
const int FPS = 60;

//Global variables for handling animation
float translate_x = 0;
int frame = 0;
int direction = 1;

using namespace std;

void initializeDisplay();
void renderAnimation(int val);
void setLights();
void setMaterialParams(float aR, float aG, float aB, float dR, float dG, float dB, float sR, float sG, float sB, float shiny);

int main(int argc, char **argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    glutCreateWindow("3D Animation");

    glutDisplayFunc(initializeDisplay);
    glutTimerFunc(1000/FPS, renderAnimation, 0);

    glEnable(GL_DEPTH_TEST);

    setLights();

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(40, 1, 4, 20);

    glMatrixMode(GL_MODELVIEW);
```

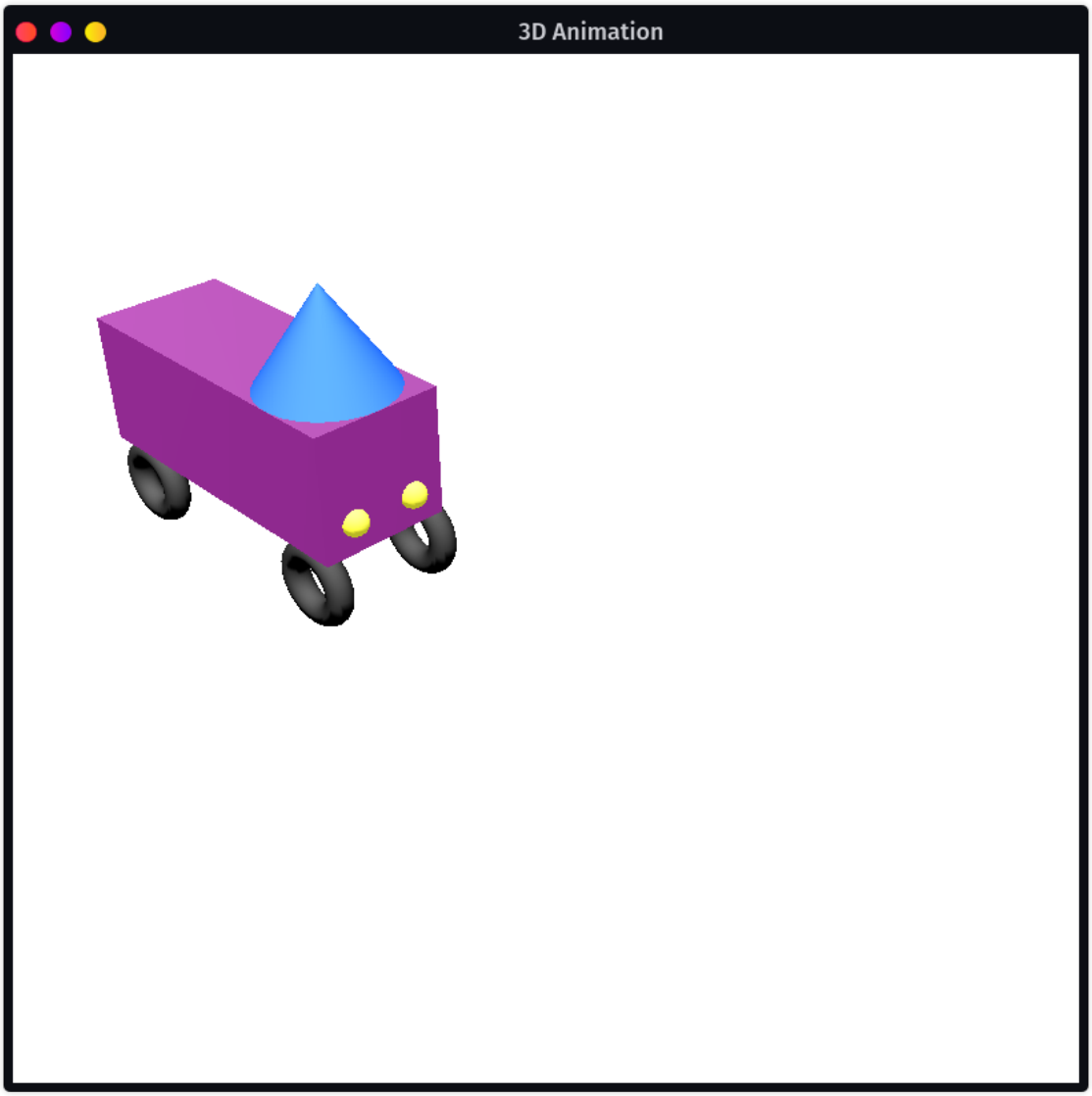
```
46      glLoadIdentity();
47      gluLookAt(5, 5, 5, 0, 0, 0, 0, 1, 0);
48
49      glutMainLoop();
50  }
51
52  void initializeDisplay(){
53      glClearColor(1, 1, 1, 1);
54      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
55
56      glMatrixMode(GL_MODELVIEW);
57
58      glColor4f(0, 0, 0, 0.3);
59
60      //Draw a car
61
62      glPushMatrix();
63      glTranslatef(translate_x, 0, 0);
64
65      //Body
66      glPushMatrix();
67      glScalef(5, 2, 2);
68      setMaterialParams(0.75, 0.2, 0.75, 1, 1, 1, 1, 1, 1, 100);
69      glutSolidCube(0.5);
70      glPopMatrix();
71
72      //Wheels
73      setMaterialParams(0, 0, 0, 0, 0, 0, 1, 1, 1, 1);
74
75      //Back Left
76      glPushMatrix();
77      glTranslatef(-1.25, -1, 0.25);
78      glutSolidTorus(0.1, 0.25, 30, 30);
79      glPopMatrix();
80
81      //Front Left
82      glPushMatrix();
83      glTranslatef(0.75, -1, 0.25);
84      glutSolidTorus(0.1, 0.25, 30, 30);
85      glPopMatrix();
86
87      //Back Right
88      glPushMatrix();
89      glTranslatef(-1.25, -1, -0.7);
90      glutSolidTorus(0.1, 0.25, 30, 30);
91      glPopMatrix();
92
93      //Front Right
94      glPushMatrix();
95      glTranslatef(0.75, -1, -0.7);
96      glutSolidTorus(0.1, 0.25, 30, 30);
```

```
97      glPopMatrix();
98
99      //Headlights
100     setMaterialParams(1, 1, 0.2, 1, 1, 1, 1, 1, 0.2, 300);
101
102     //Left
103     glPushMatrix();
104     glTranslatef(1.25, -0.25, 0.25);
105     glutSolidSphere(0.1, 30, 30);
106     glPopMatrix();
107
108     //Right
109     glPushMatrix();
110     glTranslatef(1.25, -0.25, -0.25);
111     glutSolidSphere(0.1, 30, 30);
112     glPopMatrix();
113
114     //Roof Hat
115     glPushMatrix();
116
117     glRotatef(270, 1, 0, 0);
118     glTranslatef(0.75, 0, 0.5); //Order important. Rotate->Translate
119     //Last command acted on first in OpenGL, thus rotate about fixed point
        here
120
121     setMaterialParams(0, 0.25, 1, 0, 0.5, 1, 1, 1, 1, 1);
122     //setMaterialParams(0, 0.5, 1, 0, 0.5, 1, 1, 1, 1, 50);
123     glutSolidCone(0.5, 0.75, 30, 30);
124
125     glPopMatrix();
126
127     glPopMatrix();
128
129     glFlush();
130     glutSwapBuffers();  //Swap the offscreen buffer to screen
131 }
132
133 void renderAnimation(int val){
134     //Render an animation frame by frame
135
136     frame = (frame % FPS) + 1;
137
138     if(frame % 5 == 0){
139         translate_x += (0.04 * direction);
140         glutPostRedisplay();
141     }
142
143     if(translate_x >= 1.40 || translate_x <= -3.40){
144         direction *= -1;
145     }
146
```

```
147     //Call the timer function again to keep animating
148     glutTimerFunc(1000/FPS, renderAnimation, 0);
149 
150 }
151 
152 void setMaterialParams(float aR, float aG, float aB, float dR, float dG,
        float dB, float sR, float sG, float sB, float shiny){
153     //Set material's ambient, diffuse and specular component colors, along
        with
154     //the shininess of the material
155 
156     float ambient[3] = {aR, aG, aB};
157     float diffuse[3] = {dR, dG, dB};
158     float specular[3] = {sR, sG, sB};
159 
160     glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, ambient);
161     glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, diffuse);
162     glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, specular);
163     glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, shiny);
164 }
165 
166 void setLights(){
167     glShadeModel(GL_SMOOTH);    //Enable smooth shading of objects
168 
169     //Set modelview matrix for the lighting
170     glMatrixMode(GL_MODELVIEW);
171     glLoadIdentity();
172 
173     float lightPosition[] = {0.0, 10.0, 5.0};
174     float lightColor[] = {0.5, 0.5, 0.5};
175     float ambientColor[] = {0.3, 0.3, 0.3};
176     float spotDirection[] = {-1.0, -1.0, -1.0};
177 
178     glEnable(GL_LIGHTING);
179     glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);
180 
181     glEnable(GL_LIGHT0);
182     glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
183     glLightfv(GL_LIGHT0, GL_AMBIENT, lightColor);
184     glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor);
185     glLightfv(GL_LIGHT0, GL_SPECULAR, lightColor);
186     glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 37.0);
187     glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spotDirection);
188     glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 1);
189 }
```

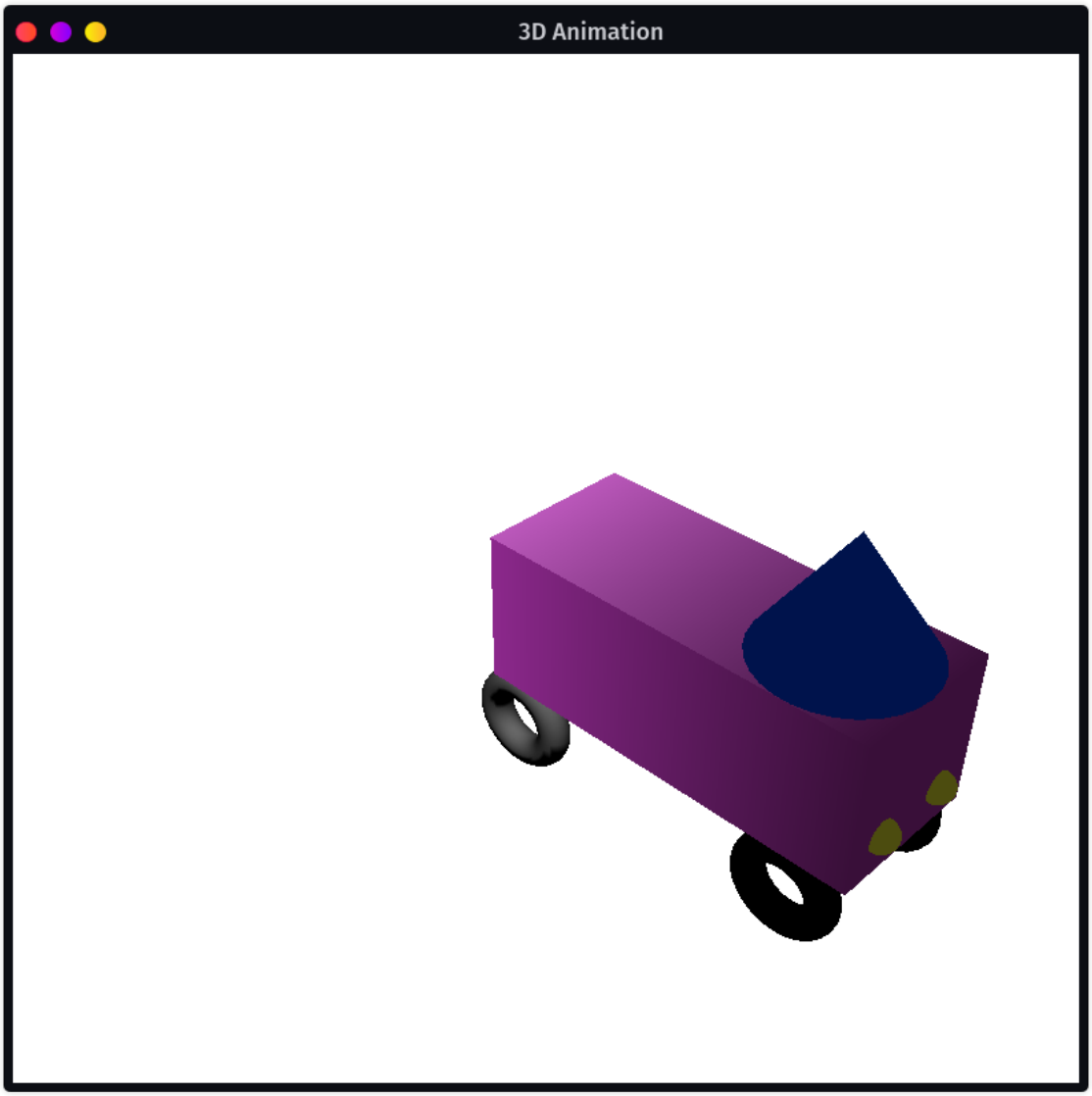
Output: Scene - 1

Figure 1: Scene - 1.



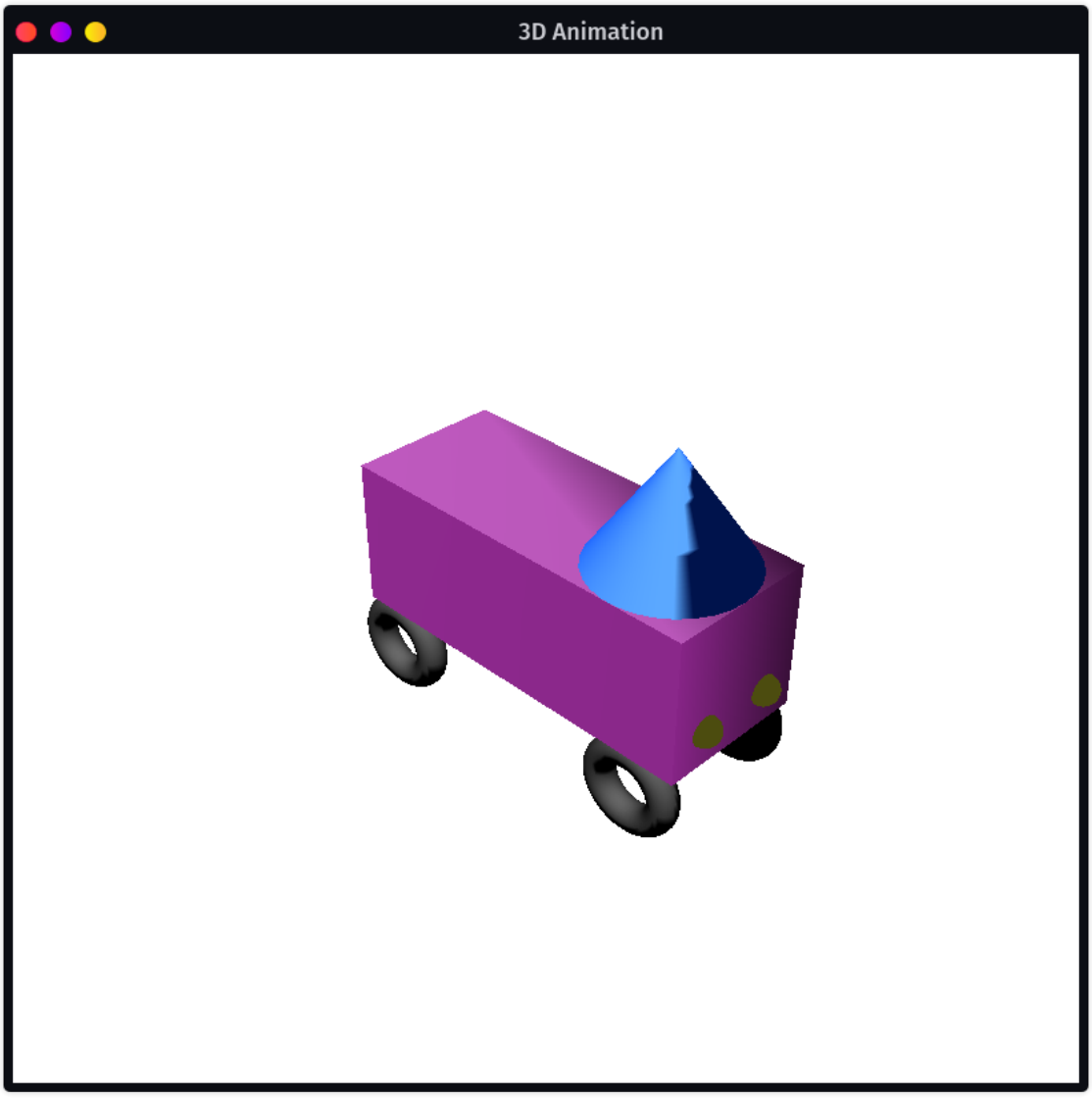
Output: Scene - 2

Figure 2: Scene - 2.



Output: Scene - 3

Figure 3: Scene - 3.



Learning Outcome:

- I learnt how to use in-built 3-D object drawing methods for generating a **Torus, Cube, Cone and Sphere**.
- I was able to apply in-built 3-D transformations to modify the objects & position them appropriately.
- I used these functions to draw a primitive 3D Car.
- I understood how to set different material parameters and transformations for different objects using the **glPushMatrix() and glPopMatrix()** methods.
- I understood how to animate the car object (perform translation) using an **FPS counter & translation variables along the X-axis** and **glutPostRedisplay()** to redraw the scene.
- I learnt how to define **material parameters** for **Ambient, Diffuse and Specular** components.
- I learnt how to set-up a basic lighting model & added **Ambient, Diffuse and Specular** components to it.
- I implemented **spot lighting and spot directionality** for the light using **GL_SPOT_DIRECTION & GL_SPOT_CUTOFF**.
- I was able to animate the car in and out of the lighting area.