

Department of CSE

SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VII

19 July 2021

UCS 1712 - Graphics And Multimedia Lab

Exercise 1: Study of Basic Output Primitives in C++ using OpenGL

Aim:

- To create an output window using OPENGL and to draw the following basic output primitives - POINTS, LINES, LINE STRIP, LINE LOOP, TRIANGLES, QUADS, QUAD STRIP, POLYGON.
- To create an output window and draw a checkerboard using OpenGL.
- To create an output window and draw a house using POINTS, LINES, TRIANGLES and QUADS/POLYGON.

Code: Basic Primitives:

```
1 //Basic output primitives using OpenGL
2 //Shapes: Points, Lines, Line Strips, Line Loops, Triangles, Quads, Quad
  Strips and Polygons
3
4 //Documentation: https://docs.microsoft.com/en-us/windows/win32/opengl/gl-
  functions
5
6 #include <windows.h>
7 #include <GL/glut.h>
8
9 const int WINDOW_WIDTH = 800;
10 const int WINDOW_HEIGHT = 600;
11
12 void initializeDisplay();
13 void displayShapes();
14
15 void displayPoints();
16 void displayLines();
17 void displayLineStrips();
18 void displayLineLoops();
19 void displayTriangles();
20 void displayQuads();
21 void displayQuadStrips();
22 void displayPolygons();
23
24 int main(int argc, char **argv){
25     glutInit(&argc, argv);
26     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
27     glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
28     glutCreateWindow("Basic Shapes");
29     glutDisplayFunc(displayShapes);
30     initializeDisplay();
31     glutMainLoop();
32
33     return 1;
34 }
35
36 void initializeDisplay(){
37     glClearColor(1.0, 1.0, 1.0, 0.0); //The glClearColor function
    specifies clear values for the color buffers.
38     glColor3f(255.0f, 0.0f, 127.0f); //Sets the current color.
39     glPointSize(5); //The glPointSize function
    specifies the diameter of rasterized points.
40     glMatrixMode(GL_PROJECTION); //The glMatrixMode function
    specifies which matrix is the current matrix.
41     glLoadIdentity(); //The glLoadIdentity function
    replaces the current matrix with the identity matrix.
```

```

42     gluOrtho2D(0.0, 800.0, 0.0, 600.0); //The gluOrtho2D function defines
    a 2-D orthographic projection matrix.
43 }
44
45 void displayShapes(){
46     glClear(GL_COLOR_BUFFER_BIT);
47
48     displayPoints();
49     displayLines();
50     displayLineStrips();
51     displayLineLoops();
52     displayTriangles();
53     displayQuads();
54     displayQuadStrips();
55     displayPolygons();
56
57     glFlush();
58 }
59
60 void displayPoints(){
61     //Treats each vertex as a single point.
62     //Vertex n defines point n. N points are drawn.
63
64     glBegin(GL_POINTS);
65
66     glVertex2d(10, 10);
67     glVertex2d(15, 15);
68     glVertex2d(20, 20);
69     glVertex2d(25, 25);
70     glVertex2d(30, 30);
71
72     glEnd();
73 }
74
75 void displayLines(){
76     //Treats each pair of vertices as an independent line segment.
77     //Vertices 2n - 1 and 2n define line n. N/2 lines are drawn.
78
79     glBegin(GL_LINES);
80
81     glVertex2d(0, 0);
82     glVertex2d(800, 600);
83
84     glEnd();
85 }
86
87 void displayLineStrips(){
88     //Draws a connected group of line segments from the first vertex to
    the last.
89     // Vertices n and n+1 define line n. N - 1 lines are drawn.
90

```

```

91     glBegin(GL_LINE_STRIP);
92
93     glVertex2d(100, 100);
94     glVertex2d(200, 200);
95
96     glVertex2d(200, 500);
97     glVertex2d(500, 600);
98
99     glEnd();
100 }
101
102 void displayLineLoops(){
103     //Draws a connected group of line segments from the first vertex to
104     the last,
105     //then back to the first. Vertices n and n + 1 define line n.
106     //The last line, however, is defined by vertices N and 1. N lines are
107     drawn.
108
109     glBegin(GL_LINE_LOOP);
110
111     glVertex2d(650, 250);
112     glVertex2d(750, 250);
113     glVertex2d(750, 350);
114     glVertex2d(650, 350);
115
116     glEnd();
117 }
118
119 void displayTriangles(){
120     //Treats each triplet of vertices as an independent triangle.
121     //Vertices 3n - 2, 3n - 1, and 3n define triangle n. N/3 triangles are
122     drawn.
123
124     glBegin(GL_TRIANGLES);
125
126     glVertex2d(170, 170);
127     glVertex2d(170, 220);
128     glVertex2d(150, 200);
129
130     glEnd();
131 }
132
133 void displayQuads(){
134     //Treats each group of four vertices as an independent quadrilateral.
135     //Vertices 4n - 3, 4n - 2, 4n - 1, and 4n define quadrilateral n. N/4
136     quadrilaterals are drawn.
137
138     glBegin(GL_QUADS);
139
140     glVertex2d(400, 400);
141     glVertex2d(450, 400);

```

```

138     glVertex2d(450, 500);
139     glVertex2d(400, 500);
140
141     glEnd();
142 }
143
144 void displayQuadStrips(){
145     //Draws a connected group of quadrilaterals.
146     //One quadrilateral is defined for each pair of vertices presented
    after the first pair.
147     //Vertices  $2n - 1$ ,  $2n$ ,  $2n + 2$ , and  $2n + 1$  define quadrilateral  $n$ .  $N/2$ 
    - 1 quadrilaterals are drawn.
148     //Note that the order in which vertices are used to construct a
    quadrilateral
149     //from strip data is different from that used with independent data.
150
151     glBegin(GL_QUAD_STRIP);
152
153     glVertex2d(320, 320);
154     glVertex2d(360, 320);
155
156     glVertex2d(320, 360);
157     glVertex2d(360, 360);
158
159     glVertex2d(360, 390);
160     glVertex2d(390, 390);
161
162     glEnd();
163
164 }
165
166 void displayPolygons(){
167     //Draws a single, convex polygon.
168     //Vertices 1 through N define this polygon.
169
170     glBegin(GL_POLYGON);
171
172     glVertex2d(510, 0);
173     glVertex2d(500, 20);
174     glVertex2d(500, 40);
175     glVertex2d(510, 60);
176     glVertex2d(520, 40);
177     glVertex2d(520, 20);
178
179     glEnd();
180 }

```

Code: Checker Board

```
1 //To draw a checkerboard using OpenGL
2
3 #include <windows.h>
4 #include <GL/glut.h>
5
6 const int WINDOW_WIDTH = 800;
7 const int WINDOW_HEIGHT = 600;
8
9 void initializeDisplay();
10 void displayCheckerboard();
11 void drawSquare(GLint x, GLint y, GLint x_step, GLint y_step);
12
13 int CURRENT_COLOR = 0;      //Global variable to keep track of current
    checker color
14
15 int main(int argc, char **argv){
16     glutInit(&argc, argv);           //Initialize glut
17     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB); //Set display mode
18     glutInitWindowPosition(100, 100); //Set Window position
19     glutInitWindowSize(800, 600);    //Set window size
20     glutCreateWindow("OpenGL Checkerboard"); //Create display window
    with title
21
22     initializeDisplay();              //Initialization procedure
23     glutDisplayFunc(displayCheckerboard); //Send graphics to display
    window
24     glutMainLoop();                  //Display everything and
    wait
25
26     return 1;
27 }
28
29 void initializeDisplay(){
30     //Initialize the display parameters
31
32     glClearColor(0, 1, 1, 0);        //Display window color
33     glMatrixMode(GL_PROJECTION);     //Choose projection
34     gluOrtho2D(0, 800, 0, 600);      //Set transformation
35 }
36
37 void displayCheckerboard(){
38     //Displays an 8x8 checkerboard
39
40     glClear(GL_COLOR_BUFFER_BIT);    //Clear display window
41     GLint x, y;
42     GLint x_step = 100, y_step = 75;
```

```

43     //For 8x8 board in an 800x600 window, x_step = 800/8 = 100, y_step =
    600/8 = 75
44
45     for(x = 0; x <= 800; x += x_step){
46         for(y = 0; y <= 600; y += y_step){
47             drawSquare(x, y, x_step, y_step);
48         }
49     }
50
51     glFlush(); //Forces execution of OpenGL functions in finite time.
52 }
53
54 void drawSquare(GLint x, GLint y, GLint x_step, GLint y_step){
55     //Draws a square, given a pair of coordinates and step sizes
56
57     GLint x1, y1, x2, y2, x3, y3, x4, y4;
58
59     //Vertex 1
60     x1 = x;
61     y1 = y + y_step;
62
63     //Vertex 2
64     x2 = x + x_step;
65     y2 = y + y_step;
66
67     //Vertex 3
68     x3 = x + x_step;
69     y3 = y;
70
71     //Vertex 4
72     x4 = x;
73     y4 = y;
74
75     if(CURRENT_COLOR == 0){
76         glColor3f(1, 1, 1); //White color
77         CURRENT_COLOR = 1;
78     }
79     else{
80         glColor3f(0, 0, 0); //Black color
81         CURRENT_COLOR = 0;
82     }
83
84     glBegin(GL_POLYGON);
85
86     glVertex2i(x1, y1);
87     glVertex2i(x2, y2);
88     glVertex2i(x3, y3);
89     glVertex2i(x4, y4);
90
91     glEnd();
92 }

```

Code: House

```
1 //To draw a house using OpenGL
2
3 #include <windows.h>
4 #include <GL/glut.h>
5
6 const int WINDOW_WIDTH = 800;
7 const int WINDOW_HEIGHT = 600;
8
9 void initializeDisplay();
10 void drawHouse();
11
12
13 int main(int argc, char **argv){
14     glutInit(&argc, argv);           //Initialize glut
15     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB); //Set display mode
16     glutInitWindowPosition(100, 100); //Set Window position
17     glutInitWindowSize(800, 600);    //Set window size
18     glutCreateWindow("OpenGL - House"); //Create display window
19     with title
20     initializeDisplay();              //Initialization procedure
21     glutDisplayFunc(drawHouse);      //Send graphics to display
22     glutMainLoop();                  //Display everything and
23     wait
24     return 1;
25 }
26
27 void initializeDisplay(){
28     glClearColor(0.5, 0.1, 1, 0);
29     glMatrixMode(GL_PROJECTION);
30     gluOrtho2D(0, 800, 0, 600);
31 }
32
33 void drawHouse(){
34     glClear(GL_COLOR_BUFFER_BIT);    //Clear display window
35
36     //Ground
37     glColor3f(0.5, 0.3, 0);
38     glBegin(GL_POLYGON);
39
40     glVertex2i(0, 100);
41     glVertex2i(800, 100);
42     glVertex2i(800, 0);
43     glVertex2i(0, 0);
44 }
```



```

45     glEnd();
46
47     //Side Roof
48     glColor3f(0.3, 0.5, 0.8);
49     glBegin(GL_POLYGON);
50
51     glVertex2i(200, 500);
52     glVertex2i(600, 500);
53     glVertex2i(700, 350);
54     glVertex2i(300, 350);
55
56     glEnd();
57
58     //Front Roof
59     glColor3f(0.1, 0.5, 0.0);
60     glBegin(GL_TRIANGLES);
61
62     glVertex2i(200, 500);
63     glVertex2i(100, 350);
64     glVertex2i(300, 350);
65
66     glEnd();
67
68     //Front Wall
69     glColor3f(0.7, 0.2, 0.3);
70     glBegin(GL_POLYGON);
71
72     glVertex2i(100, 350);
73     glVertex2i(300, 350);
74     glVertex2i(300, 100);
75     glVertex2i(100, 100);
76
77     glEnd();
78
79     //Side Wall
80     glColor3f(0.1, 0.2, 0.3);
81     glBegin(GL_POLYGON);
82
83     glVertex2i(300, 350);
84     glVertex2i(700, 350);
85     glVertex2i(700, 100);
86     glVertex2i(300, 100);
87
88     glEnd();
89
90     //Front Door
91     glColor3f(0.7, 0.2, 0.9);
92     glBegin(GL_POLYGON);
93
94     glVertex2i(150, 250);
95     glVertex2i(250, 250);

```

```

96     glVertex2i(250, 100);
97     glVertex2i(150, 100);
98
99     glEnd();
100
101     //Front Door Lock
102     glColor3f(0.3, 0.7, 0.9);
103     glPointSize(15);
104     glBegin(GL_POINTS);
105
106     glVertex2i(170, 170);
107
108     glEnd();
109
110     //Front Door Frame
111     glColor3f(1, 1, 1);
112     glLineWidth(2.5);
113     glBegin(GL_LINES);
114
115     glVertex2i(150, 250);
116     glVertex2i(250, 250);
117
118     glVertex2i(150, 100);
119     glVertex2i(150, 250);
120
121     glVertex2i(250, 100);
122     glVertex2i(250, 250);
123
124
125     glVertex2i(150, 100);
126     glVertex2i(250, 100);
127
128
129     glEnd();
130
131
132     //Pathway
133     glColor3f(0.3, 0.5, 0.7);
134     glLineWidth(5);
135     glBegin(GL_POLYGON);
136
137     glVertex2i(150, 100);
138     glVertex2i(250, 100);
139     glVertex2i(210, 0);
140     glVertex2i(40, 0);
141
142     glEnd();
143
144     //Windows
145
146     //Window - 1

```

```

147     glColor3f(0.2, 0.4, 0.3);
148     glBegin(GL_POLYGON);
149
150     glVertex2i(330, 320);
151     glVertex2i(450, 320);
152     glVertex2i(450, 230);
153     glVertex2i(330, 230);
154
155     glEnd();
156
157     //Window - 2
158     glColor3f(0.2, 0.4, 0.3);
159     glBegin(GL_POLYGON);
160
161     glVertex2i(530, 320);
162     glVertex2i(650, 320);
163     glVertex2i(650, 230);
164     glVertex2i(530, 230);
165
166     glEnd();
167
168     //Window Borders
169
170     //Window - 1
171     glColor3f(0.1, 0.7, 0.5);
172     glLineWidth(5);
173     glBegin(GL_LINES);
174
175     glVertex2i(390, 320);
176     glVertex2i(390, 230);
177     glVertex2i(330, 273);
178     glVertex2i(450, 273);
179
180     glEnd();
181
182     //Window -2
183     glColor3f(0.1, 0.7, 0.5);
184     glLineWidth(5);
185     glBegin(GL_LINES);
186
187     glVertex2i(590, 320);
188     glVertex2i(590, 230);
189     glVertex2i(530, 273);
190     glVertex2i(650, 273);
191
192     glEnd();
193
194     //Decoration
195     glColor3f(0.2, 0.4, 0.2);
196     glPointSize(5);
197     glBegin(GL_POINTS);

```

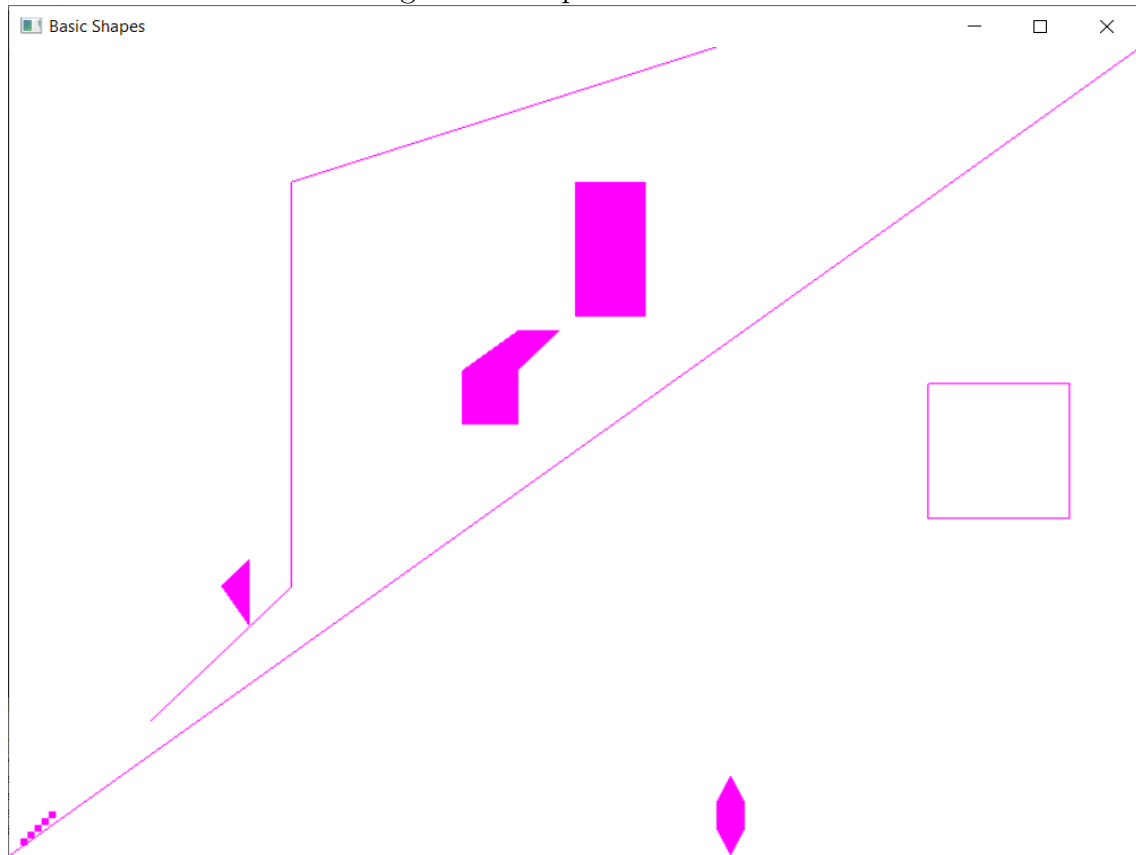
```

198
199     GLint x = 310;
200     for(x; x <= 690; x += 10){
201         glVertex2i(x, 120);
202     }
203
204     glEnd();
205
206     //Hexagonal Sun
207     glColor3f(0.8, 1, 0);
208     glBegin(GL_POLYGON);
209
210     glVertex2i(50, 500);
211     glVertex2i(75, 550);
212     glVertex2i(125, 550);
213     glVertex2i(150, 500);
214     glVertex2i(125, 450);
215     glVertex2i(75, 450);
216
217     glEnd();
218
219
220     glFlush();          //Flush the output to the display
221 }

```

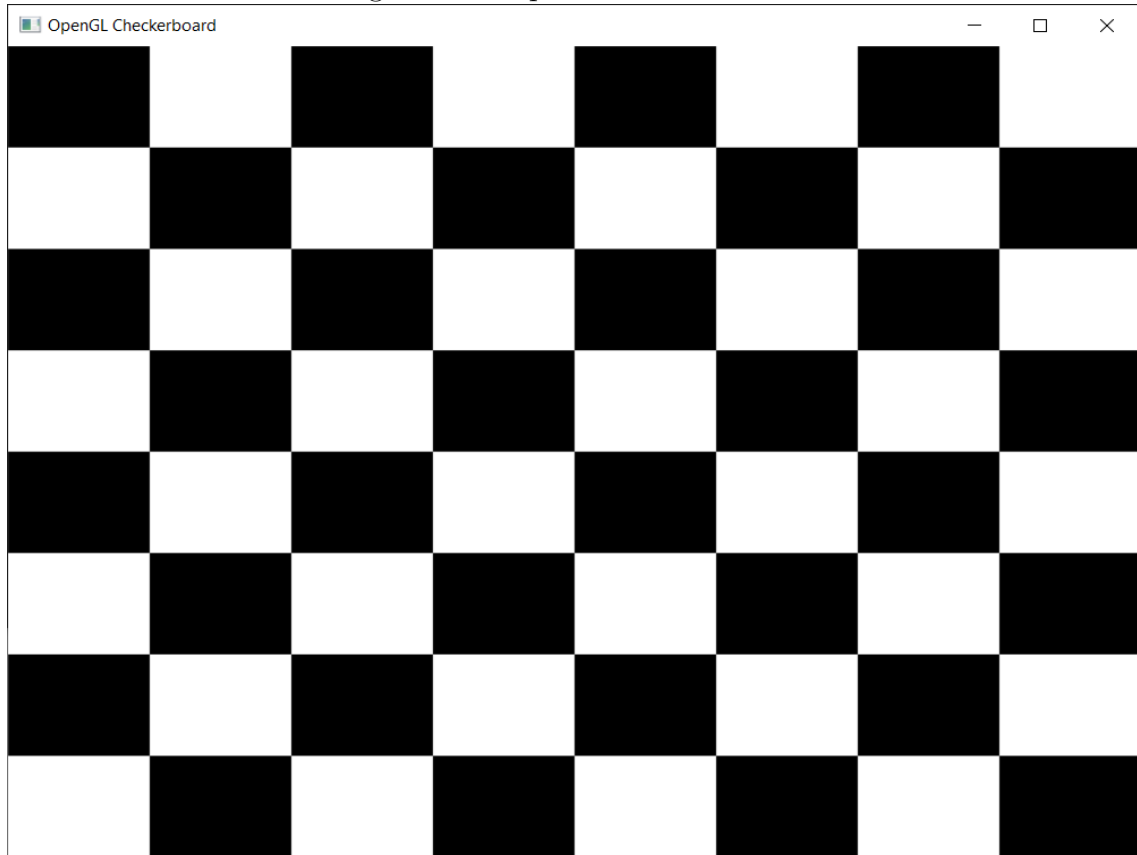
Output: Primitives:

Figure 1: Output: Primitives.



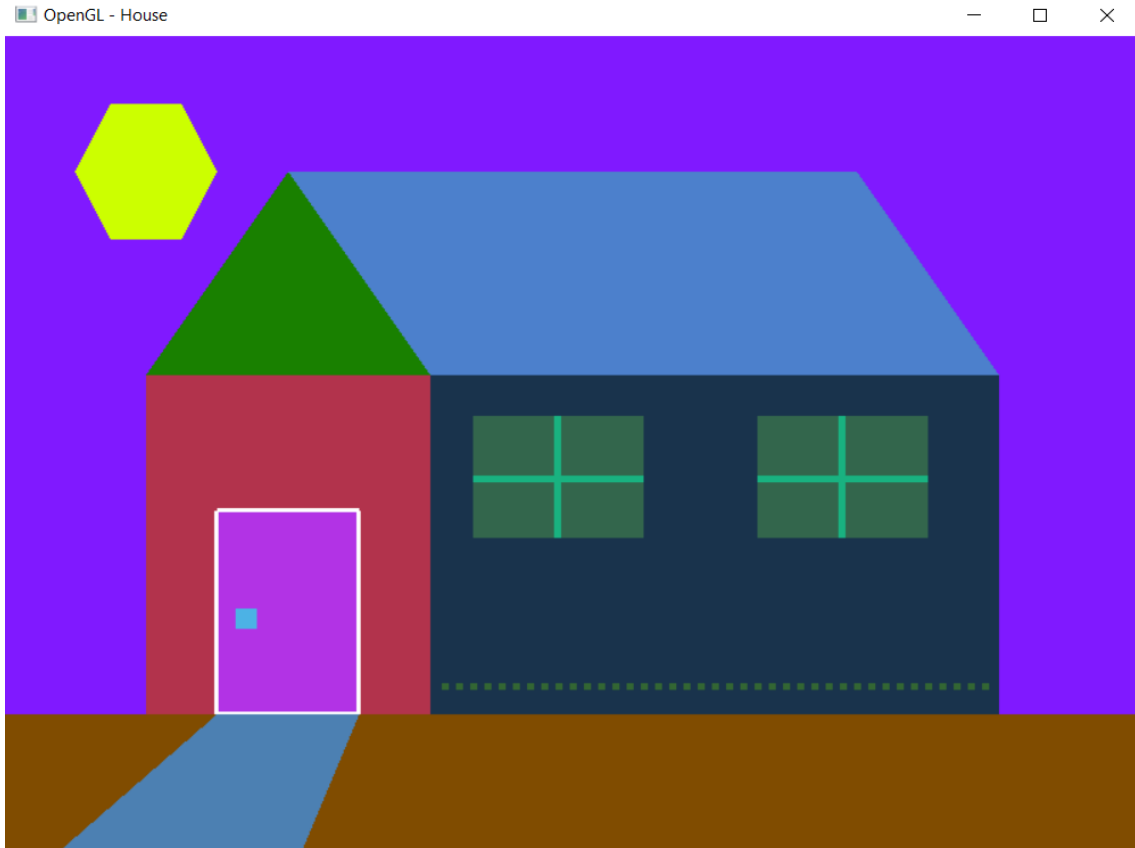
Output: Checker Board:

Figure 2: Output: Checker Board.



Output: House:

Figure 3: Output: House.



Learning Outcome:

- I configured **OpenGL** and **GLUT Framework** on my system using the CodeBlocks IDE.
- I learnt about OpenGL and its usage in the high-performance graphics industry - like creating animations and games.
- I learnt to draw some **primitive output shapes** like points, lines, line strips, line loops, triangles, quads, quad strips and polygons using GLUT's inbuilt functions.
- I understood how to **initialize a new GLUT output display** with colors, matrix mode, window title, window size etc.
- I learnt how these shapes are plotted using the **glVertex2d()** function.
- I was able to construct a basic **8x8 checkerboard** using the inbuilt primitive functions and was able to color the checkerboard appropriately.
- I was able to draw a **simple house** with shapes like triangles, quads and polygons. I was also able to color the house with different shades.
- I understood that the OpenGL uses a **coordinate system** to map the output shapes onto the display window.