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:

```
//Basic output primitives using OpenGL
//Shapes: Points, Lines, Line Strips, Line Loops, Triangles, Quads, Quad
    Strips and Polygons

//Documentation: https://docs.microsoft.com/en-us/windows/win32/opengl/gl-
    functions

#include <windows.h>
#include <GL/glut.h>

const int WINDOW_WIDTH = 800;
const int WINDOW_HEIGHT = 600;

void initializeDisplay();
void displayShapes();

void displayPoints();
void displayLines();
void displayLineStrips();
void displayLineLoops();
void displayTriangles();
void displayQuads();
void displayQuadStrips();
void displayPolygons();

int main(int argc, char **argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    glutCreateWindow("Basic Shapes");
    glutDisplayFunc(displayShapes);
    initializeDisplay();
    glutMainLoop();

    return 1;
}

void initializeDisplay(){
    glClearColor(1.0, 1.0, 1.0, 0.0);   //The glClearColor function
    specifies clear values for the color buffers.
    glColor3f(255.0f, 0.0f, 127.0f);    //Sets the current color.
    glPointSize(5);                    //The glPointSize function
    specifies the diameter of rasterized points.
    glMatrixMode(GL_PROJECTION);        //The glMatrixMode function
    specifies which matrix is the current matrix.
    glLoadIdentity();                   //The glLoadIdentity function
    replaces the current matrix with the identity matrix.
    gluOrtho2D(0.0, 800.0, 0.0, 600.0); //The gluOrtho2D function defines a 2-D orthographic projection matrix.
}

void displayShapes(){
    glClear(GL_COLOR_BUFFER_BIT);

    displayPoints();
    displayLines();
    displayLineStrips();
    displayLineLoops();
    displayTriangles();
    displayQuads();
    displayQuadStrips();
    displayPolygons();

    glFlush();
}

void displayPoints(){
    //Treats each vertex as a single point.
    //Vertex n defines point n. N points are drawn.

    glBegin(GL_POINTS);

    glVertex2d(10, 10);
    glVertex2d(15, 15);
    glVertex2d(20, 20);
    glVertex2d(25, 25);
    glVertex2d(30, 30);

    glEnd();
}

void displayLines(){
    //Treats each pair of vertices as an independent line segment.
    //Vertices 2n - 1 and 2n define line n. N/2 lines are drawn.

    glBegin(GL_LINES);

    glVertex2d(0, 0);
    glVertex2d(800, 600);

    glEnd();
}

void displayLineStrips(){
    //Draws a connected group of line segments from the first vertex to the last.
    // Vertices n and n+1 define line n. N - 1 lines are drawn.

    glBegin(GL_LINE_STRIP);

    glVertex2d(100, 100);
    glVertex2d(200, 200);

    glVertex2d(200, 500);
    glVertex2d(500, 600);

    glEnd();
}

void displayLineLoops(){
    //Draws a connected group of line segments from the first vertex to
   the last,
    //then back to the first. Vertices n and n + 1 define line n.
    //The last line, however, is defined by vertices N and 1. N lines are
   drawn.

    glBegin(GL_LINE_LOOP);

    glVertex2d(650, 250);
    glVertex2d(750, 250);
    glVertex2d(750, 350);
    glVertex2d(650, 350);

    glEnd();
}

void displayTriangles(){
    //Treats each triplet of vertices as an independent triangle.
    //Vertices 3n - 2, 3n - 1, and 3n define triangle n. N/3 triangles are
    drawn.

    glBegin(GL_TRIANGLES);

    glVertex2d(170, 170);
    glVertex2d(170, 220);
    glVertex2d(150, 200);

    glEnd();
}

void displayQuads(){
    //Treats each group of four vertices as an independent quadrilateral.
    //Vertices 4n - 3, 4n - 2, 4n - 1, and 4n define quadrilateral n. N/4
   quadrilaterals are drawn.

    glBegin(GL_QUADS);

    glVertex2d(400, 400);
    glVertex2d(450, 400);
    glVertex2d(450, 500);
    glVertex2d(400, 500);

    glEnd();
}

void displayQuadStrips(){
    //Draws a connected group of quadrilaterals.
    //One quadrilateral is defined for each pair of vertices presented after the first pair.
    //Vertices 2n - 1, 2n, 2n + 2, and 2n + 1 define quadrilateral n. N/2 - 1 quadrilaterals are drawn.
    //Note that the order in which vertices are used to construct a quadrilateral
    //from strip data is different from that used with independent data.

    glBegin(GL_QUAD_STRIP);

    glVertex2d(320, 320);
    glVertex2d(360, 320);

    glVertex2d(320, 360);
    glVertex2d(360, 360);

    glVertex2d(360, 390);
    glVertex2d(390, 390);

    glEnd();

}

void displayPolygons(){
    //Draws a single, convex polygon.
    //Vertices 1 through N define this polygon.

    glBegin(GL_POLYGON);

    glVertex2d(510, 0);
    glVertex2d(500, 20);
    glVertex2d(500, 40);
    glVertex2d(510, 60);
    glVertex2d(520, 40);
    glVertex2d(520, 20);

    glEnd();
}
```

Code: Checker Board

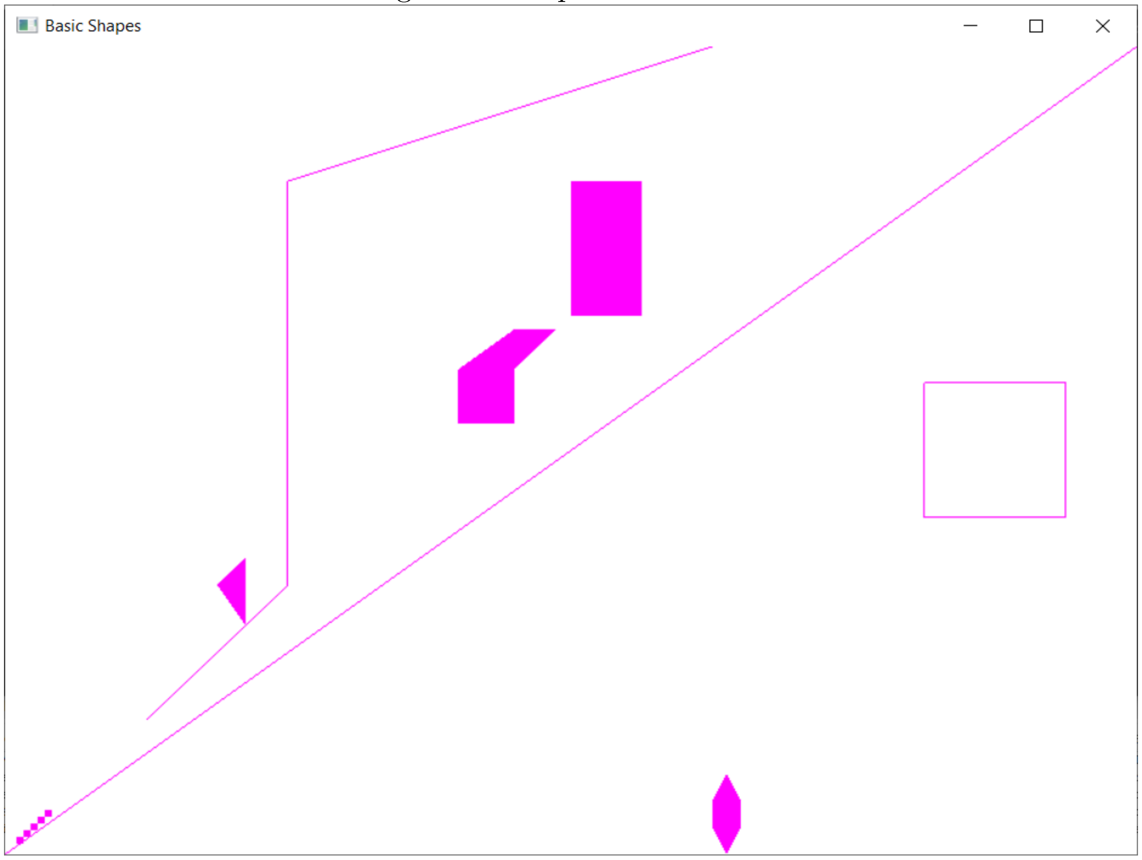
```c
//To draw a checkerboard using OpenGL

#include <windows.h>
#include <GL/glut.h>

const int WINDOW_WIDTH = 800;
const int WINDOW_HEIGHT = 600;

void initializeDisplay();
void displayCheckerboard();
void drawSquare(GLint x, GLint y, GLint x_step, GLint y_step);

int CURRENT_COLOR = 0;      //Global variable to keep track of current checker color

int main(int argc, char **argv){
    glutInit(&argc, argv);                        //Initialize glut
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);    //Set display mode
    glutInitWindowPosition(100, 100);             //Set Window position
    glutInitWindowSize(800, 600);                 //Set window size
    glutCreateWindow("OpenGL Checkerboard");      //Create display window with title

    initializeDisplay();                          //Initialization procedure
    glutDisplayFunc(displayCheckerboard);         //Send graphics to display window
    glutMainLoop();                               //Display everything and wait

    return 1;
}

void initializeDisplay(){
    //Initialize the display parameters

    glClearColor(0, 1, 1, 0);       //Display window color
    glMatrixMode(GL_PROJECTION);    //Choose projection
    gluOrtho2D(0, 800, 0, 600);     //Set transformation
}

void displayCheckerboard(){
    //Displays an 8x8 checkerboard

    glClear(GL_COLOR_BUFFER_BIT);   //Clear display window
    GLint x, y;
    GLint x_step = 100, y_step = 75;
    //For 8x8 board in an 800x600 window, x_step = 800/8 = 100, y_step =
   600/8 = 75

    for(x = 0; x <= 800; x += x_step){
        for(y = 0; y <= 600; y += y_step){
            drawSquare(x, y, x_step, y_step);
        }
    }

    glFlush();  //Forces execution of OpenGL functions in finite time.
}

void drawSquare(GLint x, GLint y, GLint x_step, GLint y_step){
    //Draws a square, given a pair of coordinates and step sizes

    GLint x1, y1, x2, y2, x3, y3, x4, y4;

    //Vertex 1
    x1 = x;
    y1= y + y_step;

    //Vertex 2
    x2 = x + x_step;
    y2 = y + y_step;

    //Vertex 3
    x3 = x + x_step;
    y3 = y;

    //Vertex 4
    x4 = x;
    y4 = y;

    if(CURRENT_COLOR == 0){
        glColor3f(1, 1, 1);     //White color
        CURRENT_COLOR = 1;
    }
    else{
        glColor3f(0, 0, 0);     //Black color
        CURRENT_COLOR = 0;
    }

    glBegin(GL_POLYGON);

    glVertex2i(x1, y1);
    glVertex2i(x2, y2);
    glVertex2i(x3, y3);
    glVertex2i(x4, y4);

    glEnd();
}
```

Code: House

```c
//To draw a house using OpenGL

#include <windows.h>
#include <GL/glut.h>

const int WINDOW_WIDTH = 800;
const int WINDOW_HEIGHT = 600;

void initializeDisplay();
void drawHouse();


int main(int argc, char **argv){
    glutInit(&argc, argv);                      //Initialize glut
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);  //Set display mode
    glutInitWindowPosition(100, 100);           //Set Window position
    glutInitWindowSize(800, 600);               //Set window size
    glutCreateWindow("OpenGL - House");         //Create display window with title

    initializeDisplay();                        //Initialization procedure
    glutDisplayFunc(drawHouse);                 //Send graphics to display window
    glutMainLoop();                             //Display everything and wait

    return 1;
}

void initializeDisplay(){
    glClearColor(0.5, 0.1, 1, 0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0, 800, 0, 600);
}

void drawHouse(){
    glClear(GL_COLOR_BUFFER_BIT);       //Clear display window

    //Ground
    glColor3f(0.5, 0.3, 0);
    glBegin(GL_POLYGON);

    glVertex2i(0, 100);
    glVertex2i(800, 100);
    glVertex2i(800, 0);
    glVertex2i(0, 0);

    glEnd();

    //Side Roof
    glColor3f(0.3, 0.5, 0.8);
    glBegin(GL_POLYGON);

    glVertex2i(200, 500);
    glVertex2i(600, 500);
    glVertex2i(700, 350);
    glVertex2i(300, 350);

    glEnd();

    //Front Roof
    glColor3f(0.1, 0.5, 0.0);
    glBegin(GL_TRIANGLES);

    glVertex2i(200, 500);
    glVertex2i(100, 350);
    glVertex2i(300, 350);

    glEnd();

    //Front Wall
    glColor3f(0.7, 0.2, 0.3);
    glBegin(GL_POLYGON);

    glVertex2i(100, 350);
    glVertex2i(300, 350);
    glVertex2i(300, 100);
    glVertex2i(100, 100);

    glEnd();

    //Side Wall
    glColor3f(0.1, 0.2, 0.3);
    glBegin(GL_POLYGON);

    glVertex2i(300, 350);
    glVertex2i(700, 350);
    glVertex2i(700, 100);
    glVertex2i(300, 100);

    glEnd();

    //Front Door
    glColor3f(0.7, 0.2, 0.9);
    glBegin(GL_POLYGON);

    glVertex2i(150, 250);
    glVertex2i(250, 250);
    glVertex2i(250, 100);
    glVertex2i(150, 100);

    glEnd();

    //Front Door Lock
    glColor3f(0.3, 0.7, 0.9);
    glPointSize(15);
    glBegin(GL_POINTS);

    glVertex2i(170, 170);

    glEnd();

    //Front Door Frame
    glColor3f(1, 1, 1);
    glLineWidth(2.5);
    glBegin(GL_LINES);

    glVertex2i(150, 250);
    glVertex2i(250, 250);

    glVertex2i(150, 100);
    glVertex2i(150, 250);

    glVertex2i(250, 100);
    glVertex2i(250, 250);


    glVertex2i(150, 100);
    glVertex2i(250, 100);


    glEnd();


    //Pathway
    glColor3f(0.3, 0.5, 0.7);
    glLineWidth(5);
    glBegin(GL_POLYGON);

    glVertex2i(150, 100);
    glVertex2i(250, 100);
    glVertex2i(210, 0);
    glVertex2i(40, 0);

    glEnd();

    //Windows

    //Window - 1
    glColor3f(0.2, 0.4, 0.3);
    glBegin(GL_POLYGON);

    glVertex2i(330, 320);
    glVertex2i(450, 320);
    glVertex2i(450, 230);
    glVertex2i(330, 230);

    glEnd();

    //Window - 2
    glColor3f(0.2, 0.4, 0.3);
    glBegin(GL_POLYGON);

    glVertex2i(530, 320);
    glVertex2i(650, 320);
    glVertex2i(650, 230);
    glVertex2i(530, 230);

    glEnd();

    //Window Borders

    //Window - 1
    glColor3f(0.1, 0.7, 0.5);
    glLineWidth(5);
    glBegin(GL_LINES);

    glVertex2i(390, 320);
    glVertex2i(390, 230);
    glVertex2i(330, 273);
    glVertex2i(450, 273);

    glEnd();

    //Window -2
    glColor3f(0.1, 0.7, 0.5);
    glLineWidth(5);
    glBegin(GL_LINES);

    glVertex2i(590, 320);
    glVertex2i(590, 230);
    glVertex2i(530, 273);
    glVertex2i(650, 273);

    glEnd();

    //Decoration
    glColor3f(0.2, 0.4, 0.2);
    glPointSize(5);
    glBegin(GL_POINTS);

    GLint x = 310;
    for(x; x <= 690; x += 10){
        glVertex2i(x, 120);
    }

    glEnd();

    //Hexagonal Sun
    glColor3f(0.8, 1, 0);
    glBegin(GL_POLYGON);

    glVertex2i(50, 500);
    glVertex2i(75, 550);
    glVertex2i(125, 550);
    glVertex2i(150, 500);
    glVertex2i(125, 450);
    glVertex2i(75, 450);

    glEnd();


    glFlush();      //Flush the output to the display
}
```

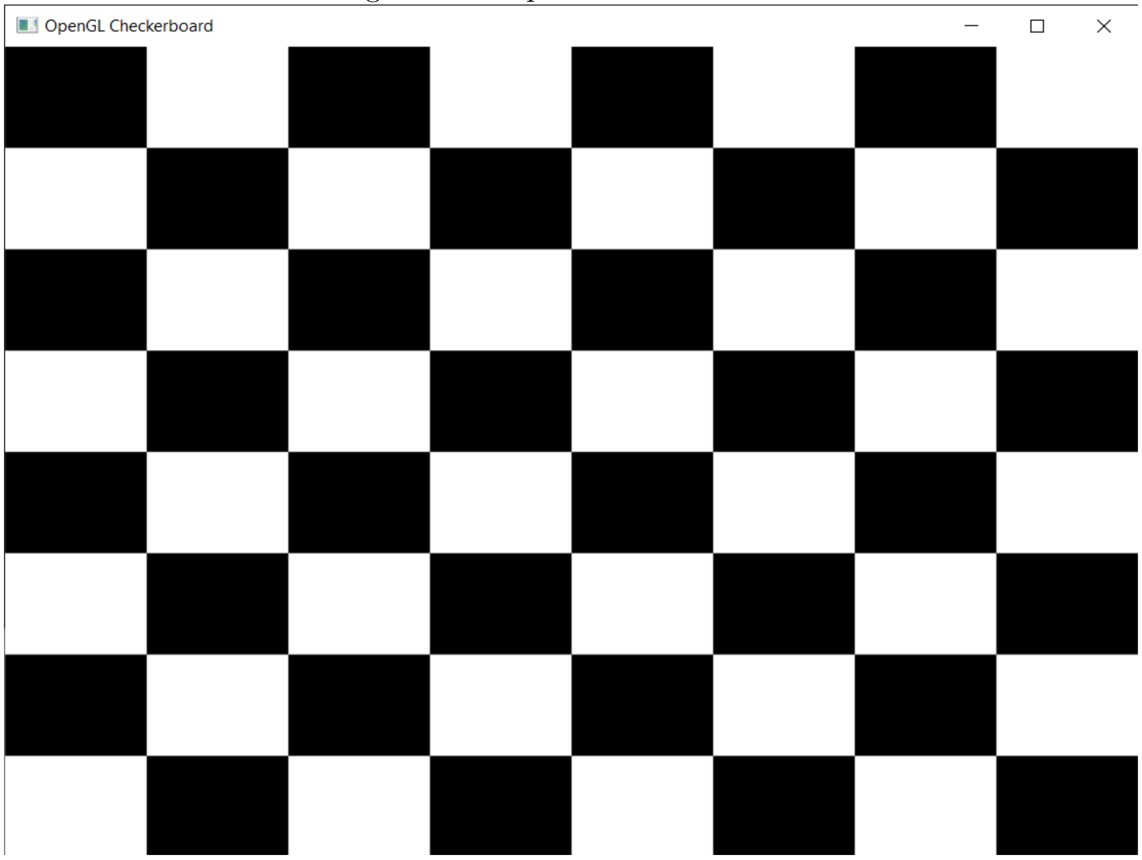
Output: Primitives:

Figure 1: Output: Primitives.



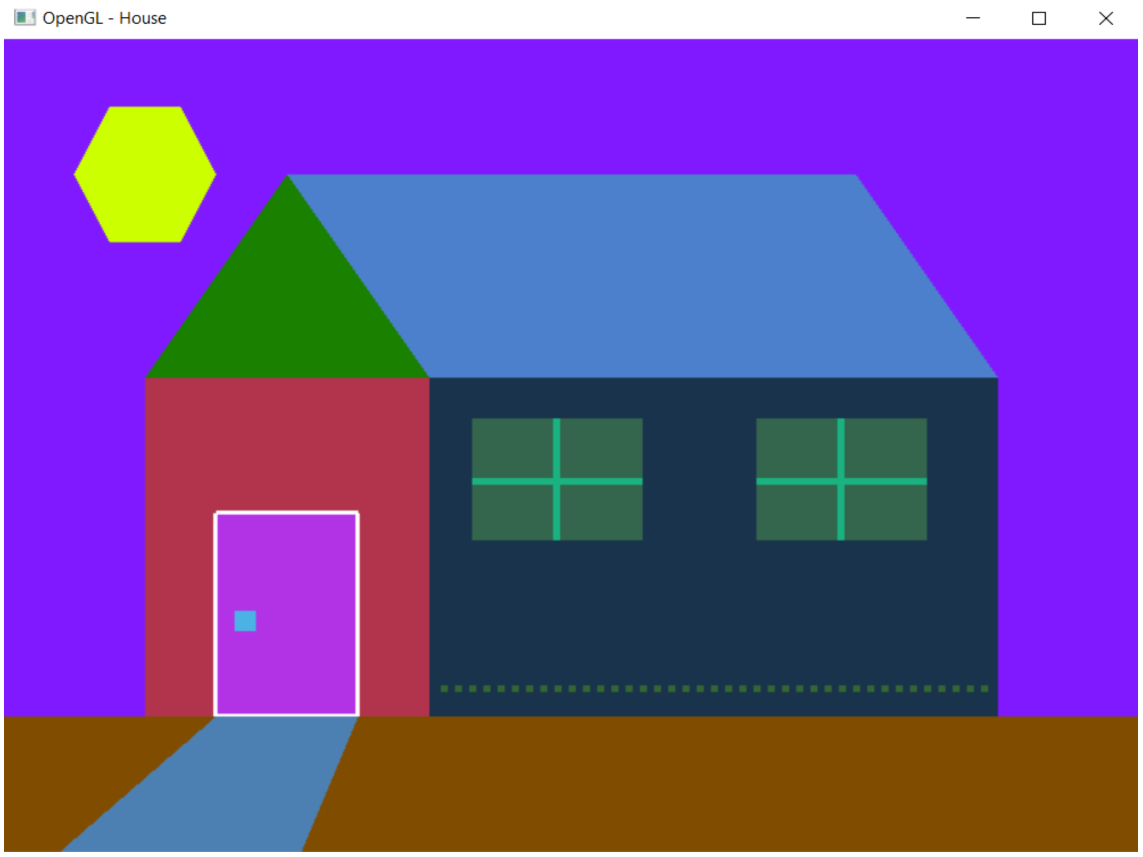
Output: Checker Board:

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.