SSN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Name : Rahul Ram M
Reg .No : 185001121
Date : 25/07/2021

EX NO: 1

Study of Basic output primitives in OpenGL

1)

Aim:

To create a window using OPENGL and to draw the following basic output primitives – POINTS, LINES, LINE_STRIP, LINE_LOOP, TRIANGLES, TRIANGLE STRIP, TRIANGLE FAN, QUADS, QUAD_STRIP, POLYGON.

Algorithm:

1. Using glutInit() to initialize glut.
2. glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB) - *initial display mode* is used when creating top-level windows, subwindows, and overlays to determine the OpenGL display mode for the to-be-created window or overlay.
3. glutInitWindowSize(800,600) - setting the initial size of the window as width and height.
4. glutInitWindowPosition(50, 50) - positioning the window (50, 50) from the top left corner.
5. glutCreateWindow("OpenGL Setup Test") - create a window with the given title.
6. glClearColor(0.0f, 0.0f, 0.0f, 0.0f) - setting the background color to black and opaque.
7. glPointSize(3); - making the point size as 3.
8. glMatrixMode(GL_PROJECTION) - glMatrixMode **sets the current matrix mode**. mode can assume one of four values: ... Applies subsequent matrix operations to the modelview matrix stack. GL_PROJECTION. Applies

subsequent matrix operations to the projection matrix stack.

9. glutDisplayFunc(display) - Register display callback handler for window re-paint.
10. glutMainLoop() - Enter the infinitely event-processing loop

Code:

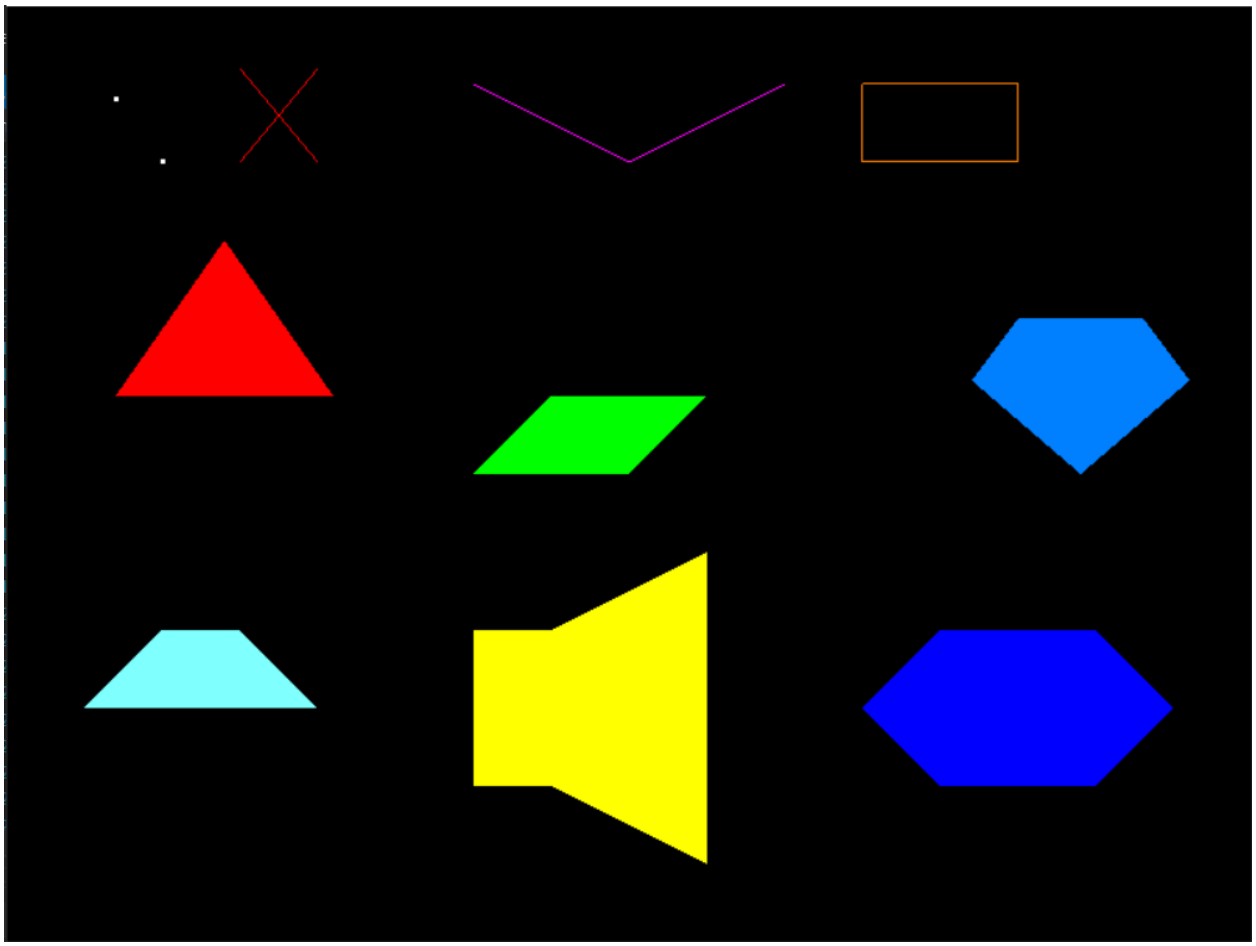
```
#include<windows.h>
#include<GL/glut.h>

void myInit()
{
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f); // Set background color to black
and opaque
    //glColor3f(0.0f, 0.0f, 0.0f);
    glPointSize(3);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 800.0, 0.0, 600.0);
}

void display() {

    glClear(GL_COLOR_BUFFER_BIT);         // Clear the color buffer

    // Points - 2
    glBegin(GL_POINTS);
    glColor3f(1.0f, 1.0f, 1.0f);
    // point 1
    glVertex2d(100, 500);
    // point 2
    glVertex2d(70, 540);
    glEnd();

    // lines - 2
    glBegin(GL_LINES);
    glColor3f(1.0f, 0.0f, 0.0f);
    // line 1
    glVertex3f(150.0f, 500.0f, 0.0f); // origin of the line
    glVertex3f(200.0f, 560.0f, 0.0f); // ending point of the line
    // line 2
    glVertex2d(150, 560);
    glVertex2d(200, 500);
    glEnd();
```

```
// line strip
glBegin(GL_LINE_STRIP);
glColor3d(127, 0, 255);
glVertex2d(300, 550);
glVertex2d(400, 500);
glVertex2d(500, 550);
glEnd();

// line loop
glBegin(GL_LINE_LOOP);
glColor3f(1.0f, 0.5f, 0.0f);
glVertex2d(550, 500);
glVertex2d(650, 500);
glVertex2d(650, 550);
glVertex2d(550, 550);
glEnd();

// triangle
glBegin(GL_TRIANGLES);
glColor3f(1.0f, 0.0f, 0.0f);
glVertex2d(70, 350);
glVertex2d(140, 450);
glVertex2d(210, 350);
glEnd();

// triangle strip
glBegin(GL_TRIANGLE_STRIP);
glColor3f(0.0f, 1.0f, 0.0f);
glVertex2d(300, 300);
glVertex2d(350, 300);
glVertex2d(350, 350);
glVertex2d(400, 300);
glVertex2d(400, 350);
glVertex2d(450, 350);
glEnd();

// triangle fan
glBegin(GL_TRIANGLE_FAN);
glColor3f(0.0f, 0.5f, 1.0f);
glVertex2d(690, 300);
glVertex2d(620, 360);
glVertex2d(650, 400);
```

```
    glVertex2d(730, 400);
    glVertex2d(760, 360);
    glEnd();

    // quads
    glBegin(GL_QUADS);
    glColor3f(0.5f, 1.0f, 1.0f);
    glVertex2d(50, 150);
    glVertex2d(100, 200);
    glVertex2d(150, 200);
    glVertex2d(200, 150);
    glEnd();

    // quad strip
    glBegin(GL_QUAD_STRIP);
    glColor4f(1.0f, 1.0f, 0.0f, 0.0f);
    glVertex2d(300, 200);
    glVertex2d(300, 100);
    glVertex2d(350, 200);
    glVertex2d(350, 100);
    glVertex2d(450, 250);
    glVertex2d(450, 50);
    glEnd();

    // polygon
    glBegin(GL_POLYGON);
    glColor3f(0.0f, 0.0f, 1.0f);
    glVertex2d(550, 150);
    glVertex2d(600, 200);
    glVertex2d(700, 200);
    glVertex2d(750, 150);
    glVertex2d(700, 100);
    glVertex2d(600, 100);
    glEnd();

    glFlush();  // Render now
}

/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
    glutInit(&argc, argv);                 // Initialize GLUT
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800,600);   // Set the window's initial width &
```

```
height
    glutInitWindowPosition(50, 50); // Position the window's initial
top-left corner
    glutCreateWindow("OpenGL Setup Test"); // Create a window with the
given title
    myInit();
    glutDisplayFunc(display); // Register display callback handler for
window re-paint
    glutMainLoop();           // Enter the infinitely event-processing loop
    return 0;
}
```

Output Screenshot:



Result:

The following basic output primitives – POINTS, LINES, LINE_STRIP, LINE_LOOP, TRIANGLES, TRIANGLE STRIP, TRIANGLE FAN, QUADS, QUAD_STRIP, POLYGON is drawn using OPENGL in a window with cpp language.

2)

Aim:

To Create a window and draw a simple House using OpenGL.

Algorithm:

1. Creating the basic structure of the house by using GL_QUADS.
2. Creating door, setp, window outline, chimney using GL_QUADS.
3. Using GL_POINTS with the point size as 3 to create a doorknob.
4. Creating the roof of the house by using GL_TRIANGLE.
5. Using GL_LINES to create the inner lines for the windows.

Code:

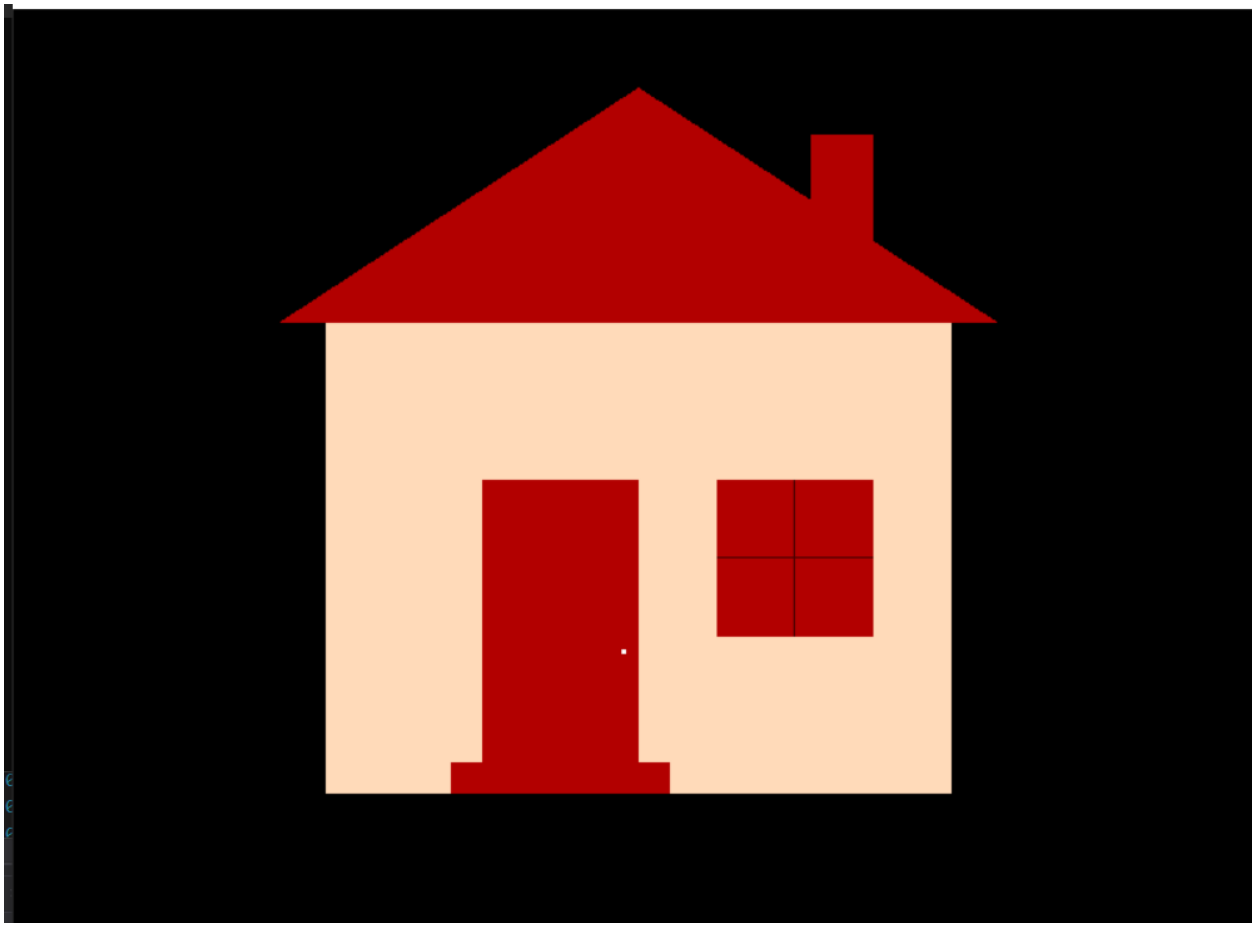
```
#include <windows.h>  // For MS Windows
#include <GL/glut.h>

void myInit()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    //glColor3f(0.0f, 0.0f, 0.0f);
    glPointSize(3);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 800.0, 0.0, 600.0);
}

// glVertex2d();

void display() {
    glClear(GL_COLOR_BUFFER_BIT);         // Clear the color buffer

    // house basic outline block
    glBegin(GL_QUADS);
    glColor3f(1.0f, 0.855f, 0.725f);
```

```
    glVertex2d(200, 400);
    glVertex2d(200, 100);
    glVertex2d(600, 100);
    glVertex2d(600, 400);
    glEnd();

    // door
    glBegin(GL_QUADS);
    glColor3f(0.7f, 0.0f, 0.0f);
    glVertex2d(300, 300);
    glVertex2d(300, 100);
    glVertex2d(400, 100);
    glVertex2d(400, 300);
    glEnd();

    // window outline
    glBegin(GL_QUADS);
    glColor3f(0.7f, 0.0f, 0.0f);
    glVertex2d(450, 300);
    glVertex2d(450, 200);
    glVertex2d(550, 200);
    glVertex2d(550, 300);
    glEnd();

    // chimney
    glBegin(GL_QUADS);
    glColor3f(0.7f, 0.0f, 0.0f);
    glVertex2d(510, 520);
    glVertex2d(510, 400);
    glVertex2d(550, 400);
    glVertex2d(550, 520);
    glEnd();

    // step
    glBegin(GL_QUADS);
    glColor3f(0.7f, 0.0f, 0.0f);
    glVertex2d(280, 120);
    glVertex2d(280, 100);
    glVertex2d(420, 100);
    glVertex2d(420, 120);
    glEnd();

    // window vertical inside line
```

```
    glBegin(GL_LINES);
    glColor3f(0.3f, 0.0f, 0.0f);
    glVertex2d(500, 200);
    glVertex2d(500, 300);
    glEnd();

    // window horizontal inside line
    glBegin(GL_LINES);
    glColor3f(0.3f, 0.0f, 0.0f);
    glVertex2d(450, 250);
    glVertex2d(550, 250);
    glEnd();

    // doorknob
    glBegin(GL_POINTS);
    glColor3f(1.0f, 1.0f, 1.0f);
    glVertex2d(390, 190);
    glEnd();

    // roof
    glBegin(GL_TRIANGLES);
    glColor3f(0.7f, 0.0f, 0.0f);
    glVertex2d(170, 400);
    glVertex2d(630, 400);
    glVertex2d(400, 550);
    glEnd();

    glFlush();  // Render now
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);                 // Initialize GLUT
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);   // Set the window's initial width &
height
    glutInitWindowPosition(50, 50); // Position the window's initial
top-left corner
    glutCreateWindow("OpenGL House build"); // Create a window with the
given title
    myInit();
    glutDisplayFunc(display); // Register display callback handler for
window re-paint
    glutMainLoop();           // Enter the infinitely event-processing loop
```

```
    return 0;
}
```

Output Screenshots:



Result:

Window is created and a simple house is drawn using OpenGL.