Date: 29/07/2025
Experiment No: 01
Experiment Name: Introduction to OpenGL with GLUT, Setup and Basic Shape Rendering

Introduction

OpenGL (Open Graphics Library) is a widely used API that allows programmers to create both 2D and 3D graphics. It is cross-platform and provides many functions to handle rendering and graphical objects. GLUT (OpenGL Utility Toolkit) works as a helper library to make window creation, input handling, and shape rendering easier. In this experiment, we focused on setting up the OpenGL environment with GLUT and learning the basics of rendering.

The main goal of this lab was to prepare our system for OpenGL programming, create a working project, and draw simple shapes like a triangle, a star, and points with different colors and sizes. Through this, we gained a practical understanding of the OpenGL coordinate system and rendering process.

This Lab Covers

- How to set up GLUT for OpenGL applications.
- How to create and configure a new OpenGL project in VS Code.
- Understanding the OpenGL coordinate system.
- Drawing shapes (triangle, star, and points) with colors and sizes.

GLUT Setup in VS Code

1. Install MinGW or any GCC compiler on your system.
2. Download and extract freeglut library files.
3. Place the files:
   - freeglut.h - MinGW include/GL folder
   - freeglut.lib (or .a) - MinGW lib folder
   - freeglut.dll - project working directory
4. Configure VS Code:
   - Install the C/C++ extension by Microsoft.
   - Create a tasks.json file in .vscode folder for build commands.
   - Example build command: `g++ main.cpp -o main.exe -lfreeglut -lopengl32 -lglu32`
5. Run the program from the terminal: `./main.exe`
   A window should appear showing the OpenGL output.

Creating a Project

To start a new project:

- In VS Code → manually create a new folder, add a main.cpp file, and configure build tasks.
- Ensure that linker settings include OpenGL and GLUT libraries.

• Build and run the program. If successful, a window with a simple shape (like a triangle) will open.

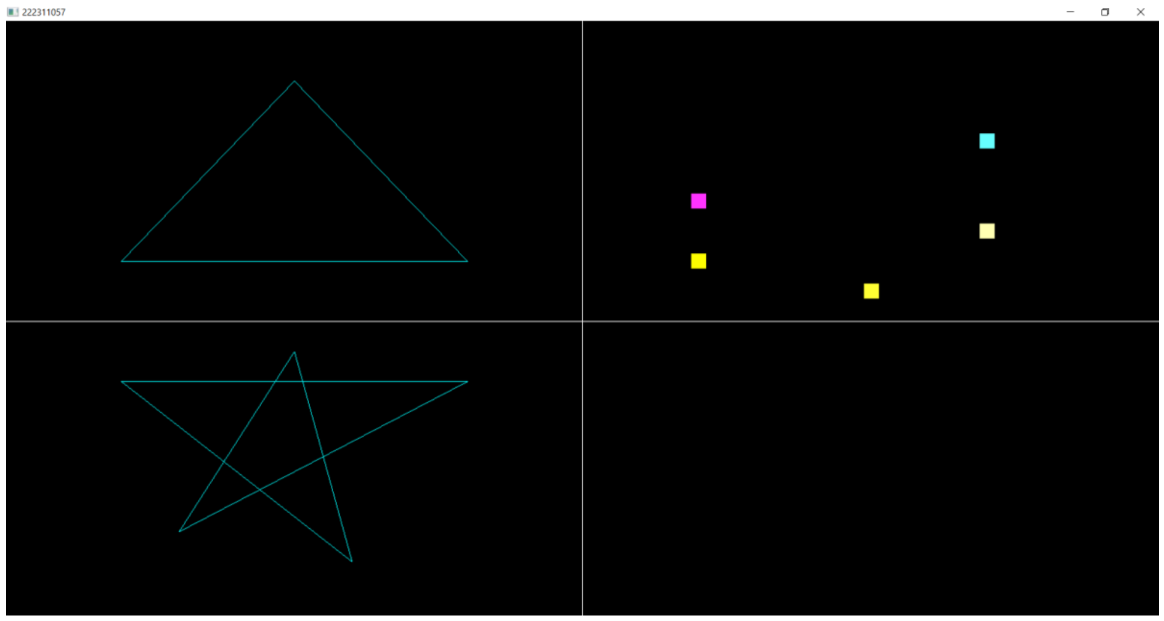
Drawing co-ordinate system, triangle, star, and 5 different points with different color and size:

Code:

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

void display() {
   glClear(GL_COLOR_BUFFER_BIT);

   // coordinate
   glColor3f(1.0, 1.0, 1.0);
   glBegin(GL_LINES);

   glVertex2f(1, 0);
   glVertex2f(-1, 0);

   glVertex2f(0, 1);
   glVertex2f(0, -1);
   glEnd();

   // triangle
   glColor3f(0.0, 1.0, 1.0);
   glBegin(GL_LINES);

   glVertex2f(-0.2, 0.2);
   glVertex2f(-0.5, 0.8);
   glVertex2f(-0.5, 0.8);
   glVertex2f(-0.8, 0.2);
   glVertex2f(-0.8, 0.2);
   glVertex2f(-0.2, 0.2);
   glEnd();

   // star
   glColor3f(0.0, 1.0, 1.0);
   glBegin(GL_LINES);

   glVertex2f(-0.8, -0.2);
   glVertex2f(-0.2, -0.2);
   glVertex2f(-0.2, -0.2);
   glVertex2f(-0.7, -0.7);
```

```
    glVertex2f(-0.7, -0.7);
    glVertex2f(-0.5, -0.1);
    glVertex2f(-0.5, -0.1);
    glVertex2f(-0.4, -0.8);
    glVertex2f(-0.4, -0.8);
    glVertex2f(-0.8, -0.2);
    glEnd();

    // 5 points with different color
    glPointSize(20.0);
    glBegin(GL_POINTS);
    glColor3f(1.0, 1.0, 0.0);
    glVertex2f(0.2, 0.2);
    glColor3f(0.4, 1.0, 1.0);
    glVertex2f(0.7, 0.6);
    glColor3f(1.0, 0.2, 1.0);
    glVertex2f(0.2, 0.4);
    glColor3f(1.0, 1.0, 0.7);
    glVertex2f(0.7, 0.3);
    glColor3f(1.0, 1.0, 0.2);
    glVertex2f(0.5, 0.1);
    glEnd();

    glFlush();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutCreateWindow("222311057");
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE | GLUT_DEPTH);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(800, 600);

    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

Output:



Conclusion

In this lab, we learned how to set up the OpenGL environment using GLUT in VS Code. We successfully created and configured projects, linked the required libraries, and ran test programs. We also practiced drawing basic shapes like triangles, stars, and colored points, which improved our understanding of the OpenGL coordinate system and rendering process. This lab built a strong foundation for future graphics programming experiments.