**Lab Taks-1**

Submission Guidelines-

* Rename the file with your serial number only.
* Must submit within the given deadline in VUES to the section named Lab Tak-1
* Must include resources for all the section in the table

|  |
| --- |
| **Question-**  Draw the object- |
| **Graph Plot (Picture)-** |
| **Code-**  /\*  \* GL02Primitive.cpp: Vertex, Primitive and Color  \* Draw Simple 2D colored Shapes: quad, triangle and polygon.  \*/  #include <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  /\* Initialize OpenGL Graphics \*/  void initGL() {  // Set "clearing" or background color  glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // white background  glColor3f(1,0,0);  }  /\* Handler for window-repaint event. Call back when the window first appears and  whenever the window needs to be re-painted. \*/  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer with current clearing color  glBegin(GL\_QUADS); // Each set of 4 vertices form a quad  glColor3f(1.0f, 1.0f, 1.0f); // white  glVertex2f(-0.2f, 0.2f);  glVertex2f(-0.2f, -0.2f);  glVertex2f(0.4f, -0.2f);  glVertex2f(0.4f, 0.2f);  glEnd();  glLineWidth(3.0f);  glBegin(GL\_LINE\_LOOP); // These vertices form a closed polygon  glColor3f(0.0f, 0.0f, 0.0f); // black  glVertex2f(-0.2f, 0.2f);  glVertex2f(-0.2f, -0.2f);  glVertex2f(0.4f, -0.2f);  glVertex2f(0.4f, 0.2f);  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  glutCreateWindow("Rectangle"); // Create window with the given title  glutInitWindowSize(500, 500); // Set the window's initial width & height  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner  glutDisplayFunc(display); // Register callback handler for window re-paint event  initGL(); // Our own OpenGL initialization  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Draw the object- |
| **Graph Plot (Picture)-** |
| **Code**:  /\*  \* GLUT Shapes Demo  \*  \* Written by Nigel Stewart November 2003  \*  \* This program is test harness for the sphere, cone  \* and torus shapes in GLUT.  \*  \* Spinning wireframe and smooth shaded shapes are  \* displayed until the ESC or q key is pressed. The  \* number of geometry stacks and slices can be adjusted  \* using the + and - keys.  \*/  #include <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  /\* Handler for window-repaint event. Call back when the window first appears and  whenever the window needs to be re-painted. \*/  void initGL(){  glClearColor(1.0f,1.0f,1.0f,1.0f);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluOrtho2D(-1.0, 1.0, -1.0, 1.0);  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3f(1.0f, 0.0f, 0.0f); // Red fill  glBegin(GL\_QUADS);  glVertex2f(-0.2f, 0.2f);  glVertex2f(-0.4f, -0.2f);  glVertex2f(0.4f, -0.2f);  glVertex2f(0.2f, 0.2f);  glEnd();  glFlush();  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInit(&argc, argv); // Initialize GLUT  glutInitWindowSize(500, 500); // Set the window's initial width & height  glutInitWindowPosition(50,50);  glutCreateWindow("Trapezoid"); // Create a window with the given title  initGL();  glutDisplayFunc(display); // Register display callback handler for window re-paint  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-**  A computer screen shot of a red logo  AI-generated content may be incorrect. |

|  |
| --- |
| **Question-**  Draw the object- |
| **Graph Plot (Picture)-** |
| **Code-**  /\*  \* GL02Primitive.cpp: Vertex, Primitive and Color  \* Draw Simple 2D colored Shapes: quad, triangle and polygon.  \*/  #include <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  /\* Initialize OpenGL Graphics \*/  void initGL() {  // Set "clearing" or background color  glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Black and opaque  }  /\* Handler for window-repaint event. Call back when the window first appears and  whenever the window needs to be re-painted. \*/  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer with current clearing color  glLineWidth(2.0f);  glBegin(GL\_LINES);  glColor3f(0.0f,0.0f,1.0f);  glVertex2f(-1.0, 0.0f);  glVertex2f(1.0f, 0.0f) ;  glVertex2f(0.0f,-1.0f);  glVertex2f(0.0f, 1.0f);  glEnd();  glBegin(GL\_QUADS); // Each set of 4 vertices form a quad  glColor3f(1.0f, 0.0f, 0.0f);  glVertex2f(-0.8f, 0.6f); // x, y  glVertex2f(-0.8f, 0.2f);  glVertex2f(-0.2f, 0.2f); // x, y  glVertex2f(-0.2f, 0.6f);  glEnd();  glBegin(GL\_QUADS);  glColor3f(0.0f, 1.0f, 0.0f);  glVertex2f(0.2f, 0.6f); // x, y  glVertex2f(0.2f, 0.4f);  glVertex2f(0.6f, 0.4f); // x, y  glVertex2f(0.6f, 0.6f);  glEnd();  glBegin(GL\_TRIANGLES);  glColor3f(0.0f, 1.0f, 0.0f);  glVertex2f(0.6f, 0.8f); // x, y  glVertex2f(0.6f, 0.2f);  glVertex2f(0.92f, 0.51f);  glEnd();  glBegin(GL\_TRIANGLES);  glColor3f(0.5f, 0.0f, 0.5f); //rgb color picker  glVertex2f(-0.2f, -0.2f); // x, y  glVertex2f(-0.8f,-0.5f);  glVertex2f(-0.2f, -0.8f);  glEnd();  glBegin(GL\_TRIANGLES);//  glColor3f(1.0f, 1.0f, 0.0f);//rgb color picker  glVertex2f(0.2f, -0.6f); // x, y  glVertex2f(0.8f,-0.6f);  glVertex2f(0.5f, -0.2f);  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  glutCreateWindow("Vertex, Primitive & Color"); // Create window with the given title  glutInitWindowSize(500, 500); // Set the window's initial width & height  glutInitWindowPosition(50,50); glutDisplayFunc(display); // Register callback handler for window re-paint event  initGL(); // Our own OpenGL initialization  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-**  **A computer screen with a black screen  AI-generated content may be incorrect.** |