**Lab Taks-2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Question- 1**  Draw a Rainbow Flag   |  | | --- | |  | |  | |  | |  | |  | |  | |  | |
| **Graph Plot (Picture)-** |
| **Code-**  //Draw Points  #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 outer\_rectangle(){  glBegin(GL\_LINES); // Each set of 4 vertices form a quad  glColor3f(0.0f, 0.0f, 0.0f);  glVertex2f(-4.0f, -3.0f); // x, y  glVertex2f(4.0f, -3.0f); // x, y  glVertex2f(4.0f, -3.0f); // x, y  glVertex2f(4.0f, 4.0f); // x, y  glVertex2f(4.0f, 4.0f); // x, y  glVertex2f(-4.0f, 4.0f); // x, y  glVertex2f(-4.0f, 4.0f); // x, y  glVertex2f(-4.0f, -3.0f); // x, y  glEnd();  }  void rectangle\_purple1(){  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glColor3f(1.0f, 0.0f, 1.0f);  glVertex2f(-4.0f, 3.0f); // x, y  glVertex2f(4.0f, 3.0f); // x, y  //glVertex2f(4.0f, 3.0f); // x, y  glVertex2f(4.0f, 4.0f); // x, y  //glVertex2f(4.0f, 4.0f); // x, y  glVertex2f(-4.0f, 4.0f); // x, y  glEnd();  }  void rectangle\_blue2(){  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glColor3f(0.0f, 0.0f, 0.9f);  glVertex2f(-4.0f, 2.0f); // x, y  glVertex2f(4.0f, 2.0f); // x, y  glVertex2f(4.0f, 2.0f); // x, y  glVertex2f(4.0f, 3.0f); // x, y  glVertex2f(4.0f, 3.0f); // x, y  glVertex2f(-4.0f, 3.0f); // x, y  glEnd();  }  void rectangle\_lightblue3(){  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glColor3f(0.0f, 0.5f, 1.0f); // Red  glVertex2f(-4.0f, 1.0f); // x, y  glVertex2f(4.0f, 1.0f); // x, y  glVertex2f(4.0f, 1.0f); // x, y  glVertex2f(4.0f, 2.0f); // x, y  glVertex2f(4.0f, 2.0f); // x, y  glVertex2f(-4.0f, 2.0f); // x, y  glEnd();  }  void rectangle\_green4(){  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glColor3f(0.0f, 1.0f, 0.0f);  glVertex2f(-4.0f, 0.0f); // x, y  glVertex2f(4.0f, 0.0f); // x, y  glVertex2f(4.0f, 0.0f); // x, y  glVertex2f(4.0f, 1.0f); // x, y  glVertex2f(4.0f, 1.0f); // x, y  glVertex2f(-4.0f, 1.0f); // x, y  glEnd();  }  void rectangle\_orange5(){  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glColor3f(1.0f, 0.5f, 0.0f);  glVertex2f(-4.0f, -1.0f); // x, y  glVertex2f(4.0f, -1.0f); // x, y  glVertex2f(4.0f, 0.0f); // x, y  glVertex2f(-4.0f, 0.0f); // x, y  glEnd();  }  void rectangle\_yellow6(){  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glColor3f(1.0f, 1.0f, 0.0f); // Red  glVertex2f(-4.0f, -2.0f); // x, y  glVertex2f(4.0f, -2.0f); // x, y  glVertex2f(4.0f, -1.0f); // x, y  glVertex2f(-4.0f, -1.0f); // x, y  glEnd();  }  void rectangle\_red7(){  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glColor3f(1.0f, 0.0f, 0.0f); // Red  glVertex2f(-4.0f, -3.0f); // x, y  glVertex2f(4.0f, -3.0f); // x, y  glVertex2f(4.0f, -2.0f); // x, y  glVertex2f(-4.0f, -2.0f); // x, y  glEnd();  }  void inner\_outlines(){  glBegin(GL\_LINES); // Each set of 4 vertices form a quad  glColor3f(0.0f, 0.0f, 0.0f); // Red  glVertex2f(-4.0f, 3.0f); // x, y  glVertex2f(4.0f, 3.0f); // x, y  glVertex2f(-4.0f, 2.0f); // x, y  glVertex2f(4.0f, 2.0f); // x, y  glVertex2f(-4.0f, 1.0f); // x, y  glVertex2f(4.0f, 1.0f); // x, y  glVertex2f(-4.0f, 0.0f); // x, y  glVertex2f(4.0f, 0.0f); // x, y  glVertex2f(-4.0f, -1.0f); // x, y  glVertex2f(4.0f, -1.0f);  glVertex2f(-4.0f, -2.0f); // x, y  glVertex2f(4.0f, -2.0f); // x, y  glEnd();  }  void display() {  glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)  glLineWidth(1.5);  outer\_rectangle();  rectangle\_purple1();  rectangle\_blue2();  rectangle\_lightblue3();  rectangle\_green4();  rectangle\_orange5();  rectangle\_yellow6();  rectangle\_red7();  inner\_outlines();  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  glutInitWindowSize(720, 520); // Set the window's initial width & height  glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title  gluOrtho2D(-5, +5,-5, +5);  glutDisplayFunc(display); // Register display callback handler for window re-paint  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question- 2**  Draw 4X4 Chess Board |
| **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.  \*///Draw Points  #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 outer\_rectangle(){  glBegin(GL\_LINES); // Each set of 4 vertices form a quad  glColor3f(0.0f, 0.0f, 0.0f);  glVertex2f(-4.0f, -4.0f); // x, y  glVertex2f(4.0f, -4.0f); // x, y  glVertex2f(4.0f, -4.0f); // x, y  glVertex2f(4.0f, 4.0f); // x, y  glVertex2f(4.0f, 4.0f); // x, y  glVertex2f(-4.0f, 4.0f); // x, y  glVertex2f(-4.0f, 4.0f); // x, y  glVertex2f(-4.0f, -4.0f); // x, y  glEnd();  }  void black\_squares(){  glColor3f(0.0f, 0.0f, 0.0f);  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glVertex2f(-4.0f, 2.0f); // x, y  glVertex2f(-2.0f, 2.0f); // x, y  glVertex2f(-2.0f, 4.0f); // x, y  glVertex2f(-4.0f, 4.0f); // x, y  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(0.0f, 2.0f); // x, y  glVertex2f(2.0f, 2.0f); // x, y  glVertex2f(2.0f, 4.0f); // x, y  glVertex2f(0.0f, 4.0f); // x, y  glEnd();  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glVertex2f(-2.0f, -0.0f); // x, y  glVertex2f(0.0f, 0.0f); // x, y  glVertex2f(0.0f, 2.0f); // x, y  glVertex2f(-2.0f, 2.0f); // x, y  glEnd();  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glVertex2f(2.0f, -0.0f); // x, y  glVertex2f(4.0f, 0.0f); // x, y  glVertex2f(4.0f, 2.0f); // x, y  glVertex2f(2.0f, 2.0f); // x, y  glEnd();  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glVertex2f(-4.0f, -2.0f); // x, y  glVertex2f(-2.0f, -2.0f); // x, y  glVertex2f(-2.0f, 0.0f); // x, y  glVertex2f(-4.0f, 0.0f); // x, y  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(0.0f, -2.0f); // x, y  glVertex2f(2.0f, -2.0f); // x, y  glVertex2f(2.0f, 0.0f); // x, y  glVertex2f(0.0f, 0.0f); // x, y  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-2.0f, -4.0f); // x, y  glVertex2f(0.0f, -4.0f); // x, y  glVertex2f(0.0f, -2.0f); // x, y  glVertex2f(-2.0f, -2.0f); // x, y  glEnd();  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glVertex2f(2.0f, -4.0f); // x, y  glVertex2f(4.0f, -4.0f); // x, y  glVertex2f(4.0f, -2.0f); // x, y  glVertex2f(2.0f, -2.0f); // x, y  glEnd();  }  void display() {  glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)  glLineWidth(3 );  outer\_rectangle();  black\_squares();  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  glutInitWindowSize(620, 620); // Set the window's initial width & height  glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title  gluOrtho2D(-5, +5,-5, +5);  glutDisplayFunc(display); // Register display callback handler for window re-paint  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question- 3**  Create the batman logo given below- |
| **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.  \*///Draw Points  #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 batman\_logo\_1(){  glColor3f(0.0f, 0.0f, 0.0f);  glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad  glVertex2f(-1.5, 4.8);  glVertex2f(-6.2, 4.8);  glVertex2f(-5.6, 4.6);  glVertex2f(-5.2, 4.4);  glVertex2f(-4.8, 4.2);  glVertex2f(-4.6, 4);  glVertex2f(-4.4, 3.6);  glVertex2f(-4.4, 2.9);  glVertex2f(-3.4, 2.7);  glVertex2f(-2.6, 2.5);  glVertex2f(-2.1, 2.2);  glVertex2f(-1.7, 1.9);  glVertex2f(-1.4, 1.6);  glVertex2f(-1.1, 1.2);  glVertex2f(-0.9, 0.9);  glVertex2f(-0.75, 0.6);  glVertex2f(-0.6, 0.35);  glVertex2f(-0.4, 0);  glVertex2f(0, -1);  glVertex2f(0.4, 0);  glVertex2f(0.4, 0);  glVertex2f(0.6, 0.35);  glVertex2f(0.75, 0.6);  glVertex2f(0.9, 0.9);  glVertex2f(1.1, 1.2);  glVertex2f(1.4, 1.6);  glVertex2f(1.7, 1.9);  glVertex2f(2.1, 2.2);  glVertex2f(2.6, 2.5);  glVertex2f(3.4, 2.7);  glVertex2f(4.4, 2.9);  glVertex2f(4.4, 3.6);  glVertex2f(4.6, 4);  glVertex2f(4.8, 4.2);  glVertex2f(5.2, 4.4);  glVertex2f(5.6, 4.6);  glVertex2f(6.2, 4.8);  glVertex2f(1.5, 4.8);  glEnd();  }  void logo\_extension(){  glColor3f(1.0f, 1.0f, 1.0f);  glBegin(GL\_POLYGON);  glVertex2f(0, 4.8);  glVertex2f(0, 3.9);  glVertex2f(-0.4, 3.8);  glVertex2f(-0.6, 4.6);  glVertex2f(-0.78, 3.8);  glVertex2f(-1, 3.5);  glVertex2f(-1.25, 3.6);  glVertex2f(-1.4, 3.8);  glVertex2f(-1.5, 4.2);  glVertex2f(-1.5, 4.8);  glEnd();  glColor3f(1.0f, 1.0f, 1.0f);  glBegin(GL\_POLYGON);  glVertex2f(0, 4.8);  glVertex2f(0, 3.9);  glVertex2f(0.4, 3.8);  glVertex2f(0.6, 4.6);  glVertex2f(0.78, 3.8);  glVertex2f(1, 3.5);  glVertex2f(1.25, 3.6);  glVertex2f(1.4, 3.8);  glVertex2f(1.5, 4.2);  glVertex2f(1.5, 4.8);  glEnd();  }  void display() {  glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT); // Clear the color buffer (background)  glLineWidth(2.5);  batman\_logo\_1();  logo\_extension();  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  glutInitWindowSize(820, 420); // Set the window's initial width & height  glutCreateWindow("OpenGL Setup Test"); // Create a window with the given title  gluOrtho2D(-7, +6.5,-1.5, +5.5);  glutDisplayFunc(display); // Register display callback handler for window re-paint  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| **Output Screenshot (Full Screen)-** |