**Lab Taks-1**

|  |
| --- |
| **Question-**  Draw the object- |
| **Graph Plot (Picture)-** |
| **Code-**  **#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 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);**  **// Draw a Red 1x1 Square centered at origin**  **glBegin(GL\_LINES); // Each set of 4 vertices form a quad**  **glColor3f(0.0f, 0.0f, 0.0f); // Red**  **glVertex2f(0.8f, -0.4f); // x, y**  **glVertex2f(0.8f, 0.4f); // x, y**  **glVertex2f(0.8f, 0.4f); // x, y**  **glVertex2f(-0.8f, 0.4f); // x, y**  **glVertex2f(-0.8f, 0.4f); // x, y**  **glVertex2f(-0.8f, -0.4f); // x, y**  **glVertex2f(-0.8f, -0.4f); // x, y**  **glVertex2f(0.8f, -0.4f); // x, y**  **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**  **glutInitWindowSize(520, 520); // Set the window's initial width & height**  **glutCreateWindow("OpenGL Setup"); // Create a window with the given title**  **//glutInitWindowSize(320, 320); // Set the window's initial width & height**  **glutDisplayFunc(display); // Register display callback handler for window re-paint**  **glutMainLoop(); // Enter the event-processing loop**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Draw the object- |
| **Graph Plot (Picture)-** |
| **Code-**  **#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 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);**  **// Draw a Red 1x1 Square centered at origin**  **glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad**  **glColor3f(1.0f, 0.0f, 0.0f); // Red**  **glVertex2f(0.8f, -0.4f); // x, y**  **glVertex2f(0.5f, 0.4f); // x, y**  **glVertex2f(0.5f, 0.4f); // x, y**  **glVertex2f(-0.5f, 0.4f); // x, y**  **glVertex2f(-0.5f, 0.4f); // x, y**  **glVertex2f(-0.8f, -0.4f); // x, y**  **glVertex2f(-0.8f, -0.4f); // x, y**  **glVertex2f(0.8f, -0.4f); // x, y**  **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**  **glutInitWindowSize(520, 520); // Set the window's initial width & height**  **glutCreateWindow("OpenGL Setup"); // Create a window with the given title**  **//glutInitWindowSize(320, 320); // Set the window's initial width & height**  **glutDisplayFunc(display); // Register display callback handler for window re-paint**  **glutMainLoop(); // Enter the event-processing loop**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Draw the object-  Octagon Shape | Area & Angles - Video & Lesson Transcript | Study.com |
| **Graph Plot (Picture)-** |
| **Code-**  **#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 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);**  **//##################**  **//## OCTAGON ##**  **//##################**  **glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad**  **glColor3f(1.0f, 0.0f, 0.0f); // Red**  **glVertex2f(-0.4f, -0.8f); // x, y**  **glVertex2f(0.4f, -0.8f); // x, y**  **glVertex2f(0.4f, -0.8f); // x, y**  **glVertex2f(0.8f, -0.4f); // x, y**  **glVertex2f(0.8f, -0.4f); // x, y**  **glVertex2f(0.8f, 0.4f); // x, y**  **glVertex2f(0.8f, 0.4f); // x, y**  **glVertex2f(0.4f, 0.8f); // x, y**  **glVertex2f(0.4f, 0.8f); // x, y**  **glVertex2f(-0.4f, 0.8f); // x, y**  **glVertex2f(-0.4f, 0.8f); // x, y**  **glVertex2f(-0.8f, 0.4f); // x, y**  **glVertex2f(-0.8f, 0.4f); // x, y**  **glVertex2f(-0.8f, -0.4f); // x, y**  **glVertex2f(-0.8f, -0.4f); // x, y**  **glVertex2f(-0.4f, -0.8f); // x, y**  **glEnd();**  **//##################**  **//## OUTLINE ##**  **//##################**  **glBegin(GL\_LINES); // Each set of 4 vertices form a quad**  **glColor3f(0.0f, 0.0f, 0.0f); // Black**  **glVertex2f(-0.4f, -0.8f); // x, y**  **glVertex2f(0.4f, -0.8f); // x, y**  **glVertex2f(0.4f, -0.8f); // x, y**  **glVertex2f(0.8f, -0.4f); // x, y**  **glVertex2f(0.8f, -0.4f); // x, y**  **glVertex2f(0.8f, 0.4f); // x, y**  **glVertex2f(0.8f, 0.4f); // x, y**  **glVertex2f(0.4f, 0.8f); // x, y**  **glVertex2f(0.4f, 0.8f); // x, y**  **glVertex2f(-0.4f, 0.8f); // x, y**  **glVertex2f(-0.4f, 0.8f); // x, y**  **glVertex2f(-0.8f, 0.4f); // x, y**  **glVertex2f(-0.8f, 0.4f); // x, y**  **glVertex2f(-0.8f, -0.4f); // x, y**  **glVertex2f(-0.8f, -0.4f); // x, y**  **glVertex2f(-0.4f, -0.8f); // x, y**  **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**  **glutInitWindowSize(520, 520); // Set the window's initial width & height**  **glutCreateWindow("OpenGL Setup"); // Create a window with the given title**  **//glutInitWindowSize(320, 320); // Set the window's initial width & height**  **glutDisplayFunc(display); // Register display callback handler for window re-paint**  **glutMainLoop(); // Enter the event-processing loop**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Draw the object- |
| **Graph Plot (Picture)-** |
| **Code-**  **#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 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);**  **// Draw a Red 1x1 Square centered at origin**  **// ########################**  **// ## ##**  **// ## AXIS ##**  **// ## ##**  **// ########################**  **glBegin(GL\_LINES); // Each set of 4 vertices form a quad**  **glColor3f(0.0f, 0.0f, 0.0f); // Black**  **glVertex2f(-0.95f, 0.0f); // x, y**  **glVertex2f(0.95f, 0.0f); // x, y**  **glVertex2f(0.0f, 0.95f); // x, y**  **glVertex2f(0.0f, -0.95f); // x, y**  **glEnd();**  **// ########################**  **// ## ##**  **// ## SQUARE ##**  **// ## ##**  **// ########################**  **glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad**  **glColor3f(1.0f, 0.0f, 0.0f); // Red**  **glVertex2f(-0.5f, 0.1f); // x, y**  **glVertex2f(-0.1f, 0.1f); // x, y**  **glVertex2f(-0.1f, 0.1f); // x, y**  **glVertex2f(-0.1f, 0.5f); // x, y**  **glVertex2f(-0.1f, 0.5f); // x, y**  **glVertex2f(-0.5f, 0.5f); // x, y**  **glVertex2f(-0.5f, 0.1f); // x, y**  **glVertex2f(-0.1f, 0.1f);**  **glEnd();**  **// #########################**  **// ## PURPLE ##**  **// ## TRIANGLES ##**  **// ## ##**  **// #########################**  **glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad**  **glColor3f(0.5f, 0.0f, 0.5f); // Purple**  **glVertex2f(-0.1f, -0.1f); // x, y**  **glVertex2f(-0.1f, -0.5f); // x, y**  **glVertex2f(-0.1f, -0.5f); // x, y**  **glVertex2f(-0.5f, -0.3f); // x, y**  **glVertex2f(-0.5f, -0.3f); // x, y**  **glVertex2f(-0.1f, -0.1f); // x, y**  **glEnd();**  **// #########################**  **// ## YELLOW ##**  **// ## TRIANGLES ##**  **// ## ##**  **// #########################**  **glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad**  **glColor3f(1.0f, 1.0f, 0.0f); // Yellow**  **glVertex2f(0.1f, -0.45f); // x, y**  **glVertex2f(0.5f, -0.45f); // x, y**  **glVertex2f(0.5f, -0.45f); // x, y**  **glVertex2f(0.3f, -0.15f); // x, y**  **glVertex2f(0.3f, -0.15f); // x, y**  **glVertex2f(0.1f, -0.45f); // x, y**  **glEnd();**    **// ########################**  **// ## ##**  **// ## ARROW ##**  **// ## ##**  **// ########################**  **// SQUARE**  **glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad**  **glColor3f(0.0f, 1.0f, 0.0f); // Green**  **glVertex2f(0.15f, 0.2f); // x, y**  **glVertex2f(0.55f, 0.2f); // x, y**  **glVertex2f(0.55f, 0.2f); // x, y**  **glVertex2f(0.55f, 0.4f); // x, y**  **glVertex2f(0.55f, 0.4f); // x, y**  **glVertex2f(0.15f, 0.4f); // x, y**  **glVertex2f(0.15f, 0.4f); // x, y**  **glVertex2f(0.15f, 0.2f);**  **glEnd();**  **// TRIANGLE**  **glBegin(GL\_POLYGON); // Each set of 4 vertices form a quad**  **glColor3f(0.0f, 1.0f, 0.0f); // Green**  **glVertex2f(0.55f, 0.1f); // x, y**  **glVertex2f(0.75f, 0.3f); // x, y**  **glVertex2f(0.75f, 0.3f); // x, y**  **glVertex2f(0.55f, 0.5f); // x, y**  **glVertex2f(0.55f, 0.5f); // x, y**  **glVertex2f(0.55f, 0.1f); // x, y**  **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**  **glutInitWindowSize(520, 520); // Set the window's initial width & height**  **glutCreateWindow("OpenGL Setup"); // Create a window with the given title**  **//glutInitWindowSize(320, 320); // Set the window's initial width & height**  **glutDisplayFunc(display); // Register display callback handler for window re-paint**  **glutMainLoop(); // Enter the event-processing loop**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |