**Lab Taks-2**

Submission Guidelines-

* Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.
* Must submit within time that will be discussed in class VUES to the section named Lab Tak-2
* Must include resources for all the section in the table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Question- 1**  Draw a Rainbow Flag   |  | | --- | |  | |  | |  | |  | |  | |  | |  | |
| **Graph Plot (Picture)-** |
| **Code-**  **#include <windows.h> // for MS Windows**  **#include <GL/glut.h> // GLUT, include glu.h and gl.h**  **#include <math.h>**  **void rectangle()**  **{**  **glColor3ub(148,0,211);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-6,4);**  **glVertex2f(6,4);**  **glVertex2f(6,3);**  **glVertex2f(-6,3);**  **glEnd();**  **glColor3ub(0,125,180);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-6,3);**  **glVertex2f(6,3);**  **glVertex2f(6,2);**  **glVertex2f(-6,2);**  **glEnd();**  **glColor3ub(8, 180,177);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-6,2);**  **glVertex2f(6,2);**  **glVertex2f(6,1);**  **glVertex2f(-6,1);**  **glEnd();**  **glColor3ub(0, 255,0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-6,1);**  **glVertex2f(6,1);**  **glVertex2f(6,0);**  **glVertex2f(-6,0);**  **glEnd();**  **glColor3ub(255,127, 0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-6,0);**  **glVertex2f(6,0);**  **glVertex2f(6,-1);**  **glVertex2f(-6,-1);**  **glEnd();**  **glColor3ub(255, 255, 0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-6,-1);**  **glVertex2f(6,-1);**  **glVertex2f(6,-2);**  **glVertex2f(-6,-2);**  **glEnd();**  **glColor3ub(255, 0, 0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-6,-2);**  **glVertex2f(6,-2);**  **glVertex2f(6,-3);**  **glVertex2f(-6,-3);**  **glEnd();**  **}**  **void Line()**  **{**  **glBegin(GL\_LINES);**  **glColor3ub(0,0,0);**  **glVertex2f(-6,4);**  **glVertex2f(6,4);**  **glVertex2f(6,4);**  **glVertex2f(6,-3);**  **glVertex2f(6,-3);**  **glVertex2f(-6,-3);**  **glVertex2f(-6,-3);**  **glVertex2f(-6,4);**  **glVertex2f(-6,3);**  **glVertex2f(6,3);**  **glVertex2f(-6,2);**  **glVertex2f(6,2);**  **glVertex2f(-6,1);**  **glVertex2f(6,1);**  **glVertex2f(-6,0);**  **glVertex2f(6,0);**  **glVertex2f(-6,-1);**  **glVertex2f(6,-1);**  **glVertex2f(-6,-2);**  **glVertex2f(6,-2);**  **glVertex2f(-6,-3);**  **glVertex2f(6,-3);**  **glEnd();**  **}**  **void display()**  **{**  **glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // Set background color to black and opaque**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **rectangle();**  **Line();**  **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("OpenGL Setup Test");**  **//gluOrtho2D(-0.1,0.7,-0.1,0.3); // 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**  **gluOrtho2D(-8,8,-5,5);**  **glutMainLoop(); // Enter the event-processing loop**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question- 2**  Draw 4X4 Chess Board |
| **Graph Plot (Picture)-** |
| **Code-**  **#include <windows.h>**  **#include <GL/glut.h>**  **void ChessBoard()**  **{**  **glBegin(GL\_POLYGON);**  **glColor3f(1,1,1);**  **glVertex2f(1,7);**  **glVertex2f(9,7);**  **glVertex2f(9,-1);**  **glVertex2f(1,-1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(1,1,1);**  **glVertex2f(1,7);**  **glVertex2f(3,7);**  **glVertex2f(3,5);**  **glVertex2f(1,5);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(0,0,0);**  **glVertex2f(3,7);**  **glVertex2f(5,7);**  **glVertex2f(5,5);**  **glVertex2f(3,5);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(1,1,1);**  **glVertex2f(5,7);**  **glVertex2f(7,7);**  **glVertex2f(7,5);**  **glVertex2f(5,5);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(0,0,0);**  **glVertex2f(7,7);**  **glVertex2f(9,7);**  **glVertex2f(9,5);**  **glVertex2f(7,5);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(0,0,0);**  **glVertex2f(1,5);**  **glVertex2f(3,5);**  **glVertex2f(3,3);**  **glVertex2f(1,3);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(1,1,1);**  **glVertex2f(3,5);**  **glVertex2f(5,5);**  **glVertex2f(5,3);**  **glVertex2f(3,3);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(0,0,0);**  **glVertex2f(5,5);**  **glVertex2f(7,5);**  **glVertex2f(7,3);**  **glVertex2f(5,3);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(1,1,1);**  **glVertex2f(1,3);**  **glVertex2f(3,3);**  **glVertex2f(3,1);**  **glVertex2f(1,1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(0,0,0);**  **glVertex2f(3,3);**  **glVertex2f(5,3);**  **glVertex2f(5,1);**  **glVertex2f(3,1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(1,1,1);**  **glVertex2f(5,3);**  **glVertex2f(7,3);**  **glVertex2f(7,1);**  **glVertex2f(5,1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(0,0,0);**  **glVertex2f(7,3);**  **glVertex2f(9,3);**  **glVertex2f(9,1);**  **glVertex2f(7,1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(1,1,1);**  **glVertex2f(1,1);**  **glVertex2f(3,1);**  **glVertex2f(3,-1);**  **glVertex2f(1,-1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(0,0,0);**  **glVertex2f(7,3);**  **glVertex2f(9,3);**  **glVertex2f(9,1);**  **glVertex2f(7,1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(0,0,0);**  **glVertex2f(1,1);**  **glVertex2f(3,1);**  **glVertex2f(3,-1);**  **glVertex2f(1,-1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(1,1,1);**  **glVertex2f(3,1);**  **glVertex2f(5,1);**  **glVertex2f(5,-1);**  **glVertex2f(3,-1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(0,0,0);**  **glVertex2f(5,1);**  **glVertex2f(7,1);**  **glVertex2f(7,-1);**  **glVertex2f(5,-1);**  **glEnd();**  **glBegin(GL\_POLYGON);**  **glColor3f(1,1,1);**  **glVertex2f(7,1);**  **glVertex2f(9,1);**  **glVertex2f(9,-1);**  **glVertex2f(7,-1);**  **glEnd();**  **glBegin(GL\_LINES);**  **glColor3f(0,0,0);**  **glVertex2f(1,7);**  **glVertex2f(9,7);**  **glVertex2f(9,7);**  **glVertex2f(9,-1);**  **glVertex2f(9,-1);**  **glVertex2f(1,-1);**  **glVertex2f(1,-1);**  **glVertex2f(1,7);**  **glEnd();**  **}**  **void display()**  **{**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **glClearColor(1,1, 1, 1);**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **ChessBoard();**  **glFlush();**  **}**  **int main(int argc, char\*\* argv)**  **{**  **glutInit(&argc, argv);**  **glutCreateWindow("ChessBoard");**  **glutInitWindowSize(320, 320);**  **glutInitWindowPosition(50, 50);**  **glutDisplayFunc(display);**  **gluOrtho2D(-10,10,-10,10);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question- 3**  Create the batman logo given below- |
| **Graph Plot (Picture)-** |
| **Code-**  **#include <windows.h> // for MS Windows**  **#include <GL/glut.h> // GLUT, include glu.h and gl.h**  **#include <math.h>**  **void body()**  **{**  **glColor3ub(0,0,0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-8.1632482440969, 5.7155087572377);**  **glVertex2f(-2.86,5.7);**  **glVertex2f(8.9860610304192, 5.7155087572377);**  **glVertex2f(2.88, 5.7621101411359);**  **glVertex2f(7.27,3.33);**  **glVertex2f(0.07,-0.89);**  **glVertex2f(-6.69,3.22);**  **glEnd();**  **}**  **void head()**  **{**  **glColor3ub(0,0,0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(-0.8571492050897, 5.9799785520452);**  **glVertex2f(0,3.52);**  **glVertex2f(-1.5,5);**  **glEnd();**  **glColor3ub(0,0,0);**  **glBegin(GL\_POLYGON);**  **glVertex2f(1,7.5);**  **glVertex2f(0,4);**  **glVertex2f(1.5,5);**  **glEnd();**  **}**  **void circle()**  **{**  **glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin**  **for(int i=0;i<200;i++)**  **{**  **glColor3ub(255,255,255);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=8.854-4.180;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x-10.58,y+4.18 );**  **}**  **glEnd();**  **glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin**  **for(int i=0;i<200;i++)**  **{**  **glColor3ub(255,255,255);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=12.78669-4.2889;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x-7.3047,y-4.2892 );**  **}**  **glEnd();**  **glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin**  **for(int i=0;i<200;i++)**  **{**  **glColor3ub(255,255,255);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=8.854-4.180;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x+10.58,y+4.18 );**  **}**  **glEnd();**  **glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin**  **for(int i=0;i<200;i++)**  **{**  **glColor3ub(255,255,255);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=12.78669-4.2889;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x+7.3047,y-4.2892 );**  **}**  **glEnd();**  **glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin**  **for(int i=0;i<200;i++)**  **{**  **glColor3ub(255,255,255);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=8.3-5;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x,y+8.3 );**  **}**  **glEnd();**  **glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin**  **for(int i=0;i<200;i++)**  **{**  **glColor3ub(0,0,0);**  **float pi=3.1416;**  **float A=(i\*2\*pi)/200;**  **float r=1.5;**  **float x = r \* cos(A);**  **float y = r \* sin(A);**  **glVertex2f(x,y+5 );**  **}**  **glEnd();**  **}**  **void display()**  **{**  **glClearColor(1, 1, 1, 1); // Set background color to black and opaque**  **glClear(GL\_COLOR\_BUFFER\_BIT);**  **body();**  **circle();**  **head();**  **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("OpenGL Setup Test");**  **//gluOrtho2D(-0.1,0.7,-0.1,0.3); // 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**  **gluOrtho2D(-12,12,-8,12);**  **glutMainLoop(); // Enter the event-processing loop**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |