**Lab Report. 03**

**Subject: Computer Graphics Lab**



Submitted to: Dr. Junaid

Submitted by: Nayab Ashraf (RC-299)

**Date of Submission: 25 September, 2024**

National University of Modern Languages, Department of Computer Science, Rawalpindi Campus.

**Task 01:**

a)

#include<stdio.h>

#include<iostream>

#include<GL\glut.h>

#include<math.h>

using namespace std;

void myInit()

{

glClearColor(1.0, 1.0, 1.0, 1.0);//Defines the color of the background

glMatrixMode(GL\_MODELVIEW);//takes either GL\_MODELVIEW or GL\_PROJECTION as input

glLoadIdentity();//Loads the identity matrix into the chosen matrix i.e. resets the MODEL\_VIEW matrix to identity

gluOrtho2D(0.0, 500, 0.0, 500);//Describes a unit system Right, Left, Bottom, Top

}

void displayCircle()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

GLfloat centerx, centery, radius, n;

GLfloat const PI = 3.14;

centerx = 250; centery = 250; radius = 100;

glColor3f(0.0f, 1.0f, 0.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POLYGON);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

//left eye

centerx = 200; centery = 280; radius = 20;

glColor3f(0.0f, 0.0f, 0.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POLYGON);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

//right eye

centerx = 290; centery = 280; radius = 20;

glColor3f(0.0f, 0.0f, 0.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POLYGON);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

//smile

centerx = 250; centery = 180; radius = 50;

glColor3f(0.0f, 0.0f, 0.0f);

n = 180;

glPointSize(2);

glLineWidth(4);

glBegin(GL\_POINTS);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

glFlush();

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(50, 50);

glutCreateWindow("To draw a circle");

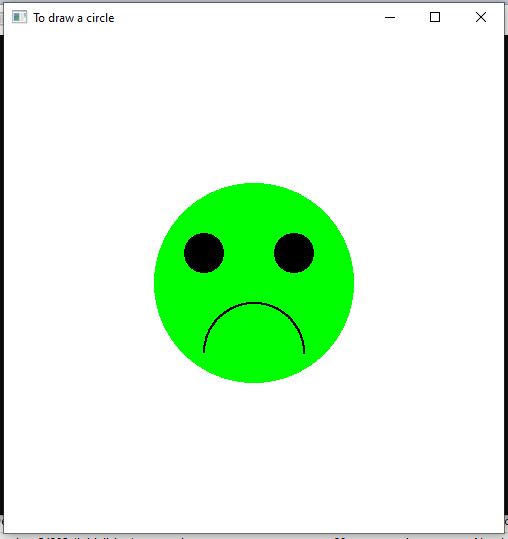
myInit();

glutDisplayFunc(displayCircle);

glutMainLoop();

return 0;

}



b)

#include<stdio.h>

#include<iostream>

#include<GL\glut.h>

#include<math.h>

using namespace std;

void myInit()

{

glClearColor(1.0, 1.0, 1.0, 1.0);//Defines the color of the background

glMatrixMode(GL\_MODELVIEW);//takes either GL\_MODELVIEW or GL\_PROJECTION as input

glLoadIdentity();//Loads the identity matrix into the chosen matrix i.e. resets the MODEL\_VIEW matrix to identity

gluOrtho2D(0.0, 500, 0.0, 500);//Describes a unit system Right, Left, Bottom, Top

}

void displayCircle()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

GLfloat centerx, centery, radius, n;

GLfloat const PI = 3.14;

centerx = 250; centery = 250; radius = 100;

glColor3f(1.0f, 1.0f, 0.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POLYGON);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

//left eye

centerx = 200; centery = 280; radius = 20;

glColor3f(0.0f, 0.0f, 0.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POLYGON);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

//right eye

centerx = 290; centery = 280; radius = 20;

glColor3f(0.0f, 0.0f, 0.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POLYGON);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

//smile

centerx = 250; centery = 230; radius = 50;

glColor3f(0.0f, 0.0f, 0.0f);

n = 360;

glPointSize(2);

glLineWidth(4);

glBegin(GL\_POINTS);

for (GLfloat i = 180; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

glFlush();

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(50, 50);

glutCreateWindow("To draw a circle");

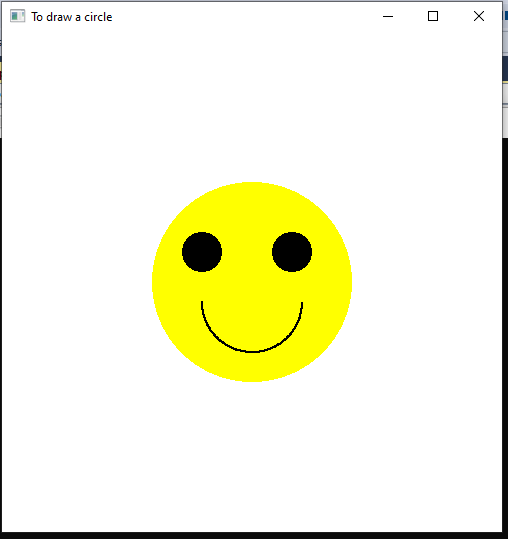
myInit();

glutDisplayFunc(displayCircle);

glutMainLoop();

return 0;

}



c)

#include<stdio.h>

#include<iostream>

#include<GL\glut.h>

#include<math.h>

using namespace std;

void myInit()

{

glClearColor(1.0, 1.0, 1.0, 1.0);//Defines the color of the background

glMatrixMode(GL\_MODELVIEW);//takes either GL\_MODELVIEW or GL\_PROJECTION as input

glLoadIdentity();//Loads the identity matrix into the chosen matrix i.e. resets the MODEL\_VIEW matrix to identity

gluOrtho2D(0.0, 500, 0.0, 500);//Describes a unit system Right, Left, Bottom, Top

}

void displayCircle()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

GLfloat centerx, centery, radius, n;

GLfloat const PI = 3.14;

centerx = 250; centery = 250; radius = 100;

glColor3f(1.0f, 0.0f, 1.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POLYGON);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

//left eye

centerx = 200; centery = 280; radius = 20;

glColor3f(0.0f, 0.0f, 0.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POLYGON);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

//right eye

centerx = 290; centery = 280; radius = 20;

glColor3f(0.0f, 0.0f, 0.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POLYGON);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(radius\*cosf(theta) + centerx, radius\*sinf(theta) + centery);

}

glEnd();

//smile

centerx = 200; centery = 180; radius = 50;

glColor3f(0.0f, 0.0f, 0.0f);

n = 360;

glPointSize(2);

glLineWidth(10);

glBegin(GL\_LINES);

glVertex2d(200, 200);

glVertex2d(300, 200);

glEnd();

glFlush();

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(50, 50);

glutCreateWindow("To draw a circle");

myInit();

glutDisplayFunc(displayCircle);

glutMainLoop();

return 0;

}



**Task 02:**

#include<stdio.h>

#include<iostream>

#include<GL\glut.h>

#include<math.h>

using namespace std;

void myInit()

{

glClearColor(1.0, 1.0, 1.0, 1.0);//Defines the color of the background

glMatrixMode(GL\_MODELVIEW);//takes either GL\_MODELVIEW or GL\_PROJECTION as input

glLoadIdentity();//Loads the identity matrix into the chosen matrix i.e. resets the MODEL\_VIEW matrix to identity

gluOrtho2D(0.0, 500, 0.0, 500);//Describes a unit system Right, Left, Bottom, Top

}

void dynamicCircle(GLfloat x, GLfloat y, GLfloat R)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

GLfloat centerx, centery, radius, n;

GLfloat const PI = 3.14;

glColor3f(1.0f, 0.0f, 0.0f);

n = 360;

glPointSize(2);

glBegin(GL\_POINTS);

for (GLfloat i = 0; i <n; i++)

{

float theta = i\*PI / 180.0;

glVertex2f(R\*cosf(theta) + x, R\*sinf(theta) + y);

}

glEnd();

}

void displayCircle()

{

dynamicCircle(250, 250, 150);

glFlush();

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(50, 50);

glutCreateWindow("To draw a circle");

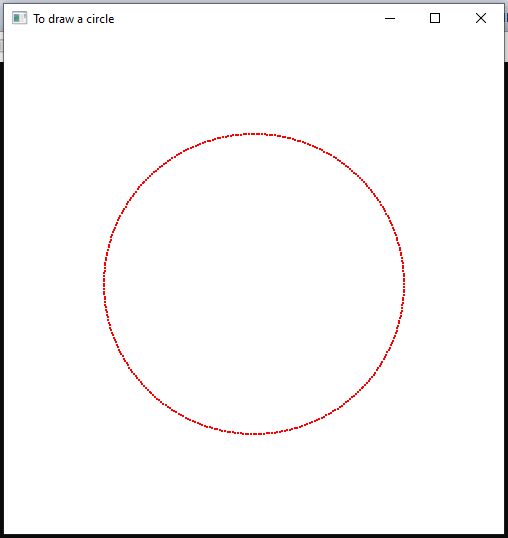
myInit();

glutDisplayFunc(displayCircle);

glutMainLoop();

return 0;

}

****