**Practical File**

**Of**

**Computer Graphics**

Submitted By:-

Aman Chauhan

D2 CSE A1

1805158

**Practical 1**

Write a program for creating a simple two-dimensional shape of any object using lines, circle, etc.

**Program:**

#include<graphics.h>

int main()

{

int i;

int gd = DETECT,gm,color;

initgraph(&gd,&gm, NULL);

for(i=0;i<100;i++)

{

putpixel(50+i,60, WHITE);

putpixel(50,60+i, WHITE);

putpixel(150,60+i, WHITE);

putpixel(50+i,160, WHITE);

}

delay(100000);

closegraph();

return 0;

}

**OUTPUT**

****

**Program 2**

Write a program to Draw a color cube and spin it using transformation matrices.

**Program:**

#include<GL/glut.h>

GLfloat vertices[]={-0.5f,-0.5f,-0.5f, -0.5f,0.5f,-0.5f, 0.5f,0.5f,-0.5f, 0.5f,-0.5f,-0.5f, -0.5f,-0.5f,0.5f, -0.5f,0.5f,0.5f, 0.5f,0.5f,0.5f, 0.5f,-0.5f,0.5f};

GLfloat colors[] = {0,0,0, 0,0,1, 0,1,0, 0,1,1, 1,0,0, 1,0,1,1,1,0, 1,1,1};

GLbyte faces[] = {0,1,2,3,2,3,7,6, 4,5,6,7,4,5,1,0, 5,6,2,1, 0,3,7,4};

GLint currentBtn = GLUT\_MIDDLE\_BUTTON;

void mouse(int btn, int state, int x, int y) {

currentBtn = btn;

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glRotated(0.06,

currentBtn == GLUT\_LEFT\_BUTTON,

currentBtn == GLUT\_MIDDLE\_BUTTON,

currentBtn == GLUT\_RIGHT\_BUTTON);

glDrawElements(GL\_QUADS, 24, GL\_UNSIGNED\_BYTE, faces);

glFlush();

}

void glInit(int w, int h) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glEnableClientState(GL\_COLOR\_ARRAY);

glEnableClientState(GL\_VERTEX\_ARRAY);

glVertexPointer(3, GL\_FLOAT, 0, vertices);

glColorPointer(3, GL\_FLOAT, 0, colors);

glEnable(GL\_DEPTH\_TEST);

glViewport(0, 0, w, h);

if (h > w)

glOrtho(-1.0, 1.0, (GLfloat) -h / w, (GLfloat) h / w, -1.0, 1.0);

else

glOrtho((GLfloat) -w / h, (GLfloat) w / h, -1.0, 1.0, -1.0, 1.0);

}

int main(int argc, char \*argv[]) {

glutInit(&argc, argv);

glutInitWindowSize(720, 720);

glutCreateWindow("Spin a cube");

glutDisplayFunc(display);

glutIdleFunc(display);

glutReshapeFunc(glInit);

glutMouseFunc(mouse);

glutMainLoop();

}

**Program 3**

Implement the DDA algorithm for drawing line (programmer is expected to shift the origin to the center of the screen and divide the screen into required quadrants).

**Program:**

#include <graphics.h>

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

float x,y,x1,y1,x2,y2,dx,dy,step;

int i,gd=DETECT,gm;

initgraph(&gd,&gm,NULL);

cout<<"Enter The Value Of x1 And y1 : ";

cin>>x1>>y1;

cout<<"Enter The Value Of x2 And y2: ";

cin>>x2>>y2;

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx>=dy)

step=dx;

else

step=dy;

dx=dx/step;

dy=dy/step;

x=x1;

y=y1;

i=1;

while(i<=step)

{

putpixel(x,y,5);

x=x+dx;

y=y+dy;

i=i+1;

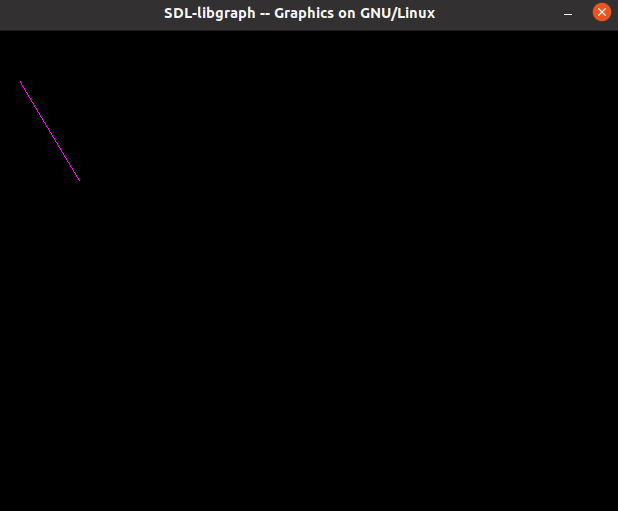
}

delay(100000);

closegraph();

}

**OUTPUT**

****

**Program 4**

Write a program to input the line coordinates from the user to generate a line using Bresenham’s Algorithm.

**Program:**

#include<iostream>

#include<graphics.h>

using namespace std;

void drawline(int x1, int y1, int x2, int y2)

{

int dx, dy, p, x, y;

dx=x2-x1;

dy=y2-y1;

x=x1;

y=y1;

p=2\*dy-dx;

while(x<x2)

{

if(p>=0)

{

putpixel(x,y,7);

y=y+1;

p=p+2\*dy-2\*dx;

}

else

{

putpixel(x,y,7);

p=p+2\*dy;

}

x=x+1;

}

}

int main()

{

int gd = DETECT, gm, error, x1, y1, x2, y2;

initgraph(&gd, &gm,NULL);

cout<<"Enter The Value Of x1 And y1 : ";

cin>>x1>>y1;

cout<<"Enter The Value Of x2 And y2: ";

cin>>x2>>y2;

drawline(x1, y1, x2, y2);

delay(100000);

closegraph();

return 0;

}

**OUTPUT**

****

**Program 5**

Write a program to generate a complete moving wheel using Midpoint circle drawing algorithm and DDA line drawing algorithm.

**Program:**

#include<iostream>

#include<graphics.h>

using namespace std;

void drawcircle(int x1, int y1, int radius)

{

int x = radius;

int y = 0;

int err = 0;

while (x >= y)

{

putpixel(x1 + x, y1 + y, 7);

putpixel(x1 + y, y1 + x, 7);

putpixel(x1 - y, y1 + x, 7);

putpixel(x1 - x, y1 + y, 7);

putpixel(x1 - x, y1 - y, 7);

putpixel(x1 - y, y1 - x, 7);

putpixel(x1 + y, y1 - x, 7);

putpixel(x1 + x, y1 - y, 7);

if (err <= 0)

{

y += 1;

err += 2\*y + 1;

}

if (err > 0)

{

x -= 1;

err -= 2\*x + 1;

}

}

}

int main()

{

int gd = DETECT, gm, x, y, r;

initgraph(&gd, &gm, NULL);

cout<<"Enter Radius Of Circle: ";

cin>>r;

cout<<"Enter Co-ordinates Of Center: ";

cin>>x>>y;

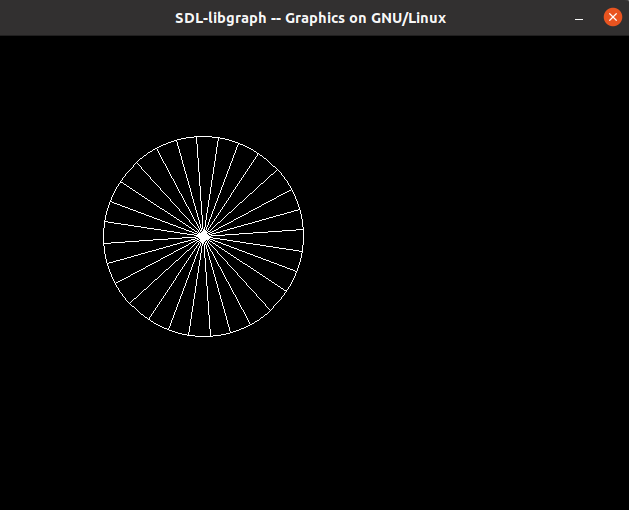
drawcircle(x, y, r);

delay(100000);

return 0;

}

**OUTPUT**

****

**Program 6**

Write a program to draw an ellipse using the Midpoint ellipse generation algorithm for both the regions.

**Program:**

#include<iostream>

#include<graphics.h>

using namespace std;

int main()

{

long x,y,x\_center,y\_center;

long a\_sqr,b\_sqr, fx,fy, d,a,b,tmp1,tmp2;

int gd = DETECT,gm;

initgraph(&gd,&gm, NULL);

cout<<"Enter The Coordinates x and y: ";

cin >> x\_center >> y\_center;

cout<<"Enter The Constants a and b: ";

cin >> a >> b;

x=0;

y=b;

a\_sqr=a\*a;

b\_sqr=b\*b;

fx=2\*b\_sqr\*x;

fy=2\*a\_sqr\*y;

d=b\_sqr-(a\_sqr\*b)+(a\_sqr\*0.25);

while(fx<fy)

{

putpixel(x\_center+x,y\_center+y,1);

putpixel(x\_center-x,y\_center-y,1);

putpixel(x\_center+x,y\_center-y,1);

putpixel(x\_center-x,y\_center+y,1);

if(d<0)

{

d=d+fx+b\_sqr;

}

else

{

y=y-1;

d=d+fx+-fy+b\_sqr;

fy=fy-(2\*a\_sqr);

}

x=x+1;

fx=fx+(2\*b\_sqr);

}

tmp1=(x+0.5)\*(x+0.5);

tmp2=(y-1)\*(y-1);

d=b\_sqr\*tmp1+a\_sqr\*tmp2-(a\_sqr\*b\_sqr);

while(y>0)

{

putpixel(x\_center+x,y\_center+y,1);

putpixel(x\_center-x,y\_center-y,1);

putpixel(x\_center+x,y\_center-y,1);

putpixel(x\_center-x,y\_center+y,1);

if(d>=0)

d=d-fy+a\_sqr;

else

{

x=x+1;

d=d+fx-fy+a\_sqr;

fx=fx+(2\*b\_sqr);

}

y=y-1;

fy=fy-(2\*a\_sqr);

}

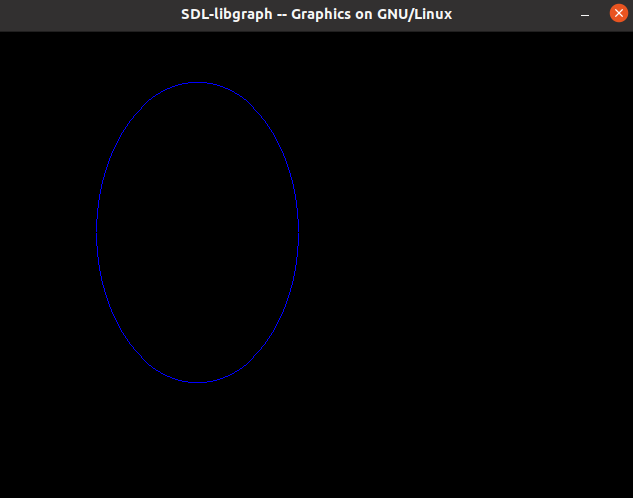
delay(100000);

closegraph();

return 0;

}

**OUTPUT**

****

**Program 7**

Write a program to draw any 2-D object and perform the transformations on it according to the input parameters from the user.

**Program:**

#include<graphics.h>

void findNewCoordinate(int s[][2], int p[][1])

{

int temp[2][1] = { 0 };

for (int i = 0; i < 2; i++)

for (int j = 0; j < 1; j++)

for (int k = 0; k < 2; k++)

temp[i][j] += (s[i][k] \* p[k][j]);

p[0][0] = temp[0][0];

p[1][0] = temp[1][0];

}

void scale(int x[], int y[], int sx, int sy)

{

line(x[0], y[0], x[1], y[1]);

line(x[1], y[1], x[2], y[2]);

line(x[2], y[2], x[0], y[0]);

int s[2][2] = { sx, 0, 0, sy };

int p[2][1];

for (int i = 0; i < 3; i++)

{

p[0][0] = x[i];

p[1][0] = y[i];

findNewCoordinate(s, p);

x[i] = p[0][0];

y[i] = p[1][0];

}

line(x[0], y[0], x[1], y[1]);

line(x[1], y[1], x[2], y[2]);

line(x[2], y[2], x[0], y[0]);

}

int main()

{

int x[] = { 100, 200, 300 };

int y[] = { 200, 100, 200 };

int sx = 2, sy = 2;

int gd = DETECT,gm;

initgraph(&gd,&gm, NULL);

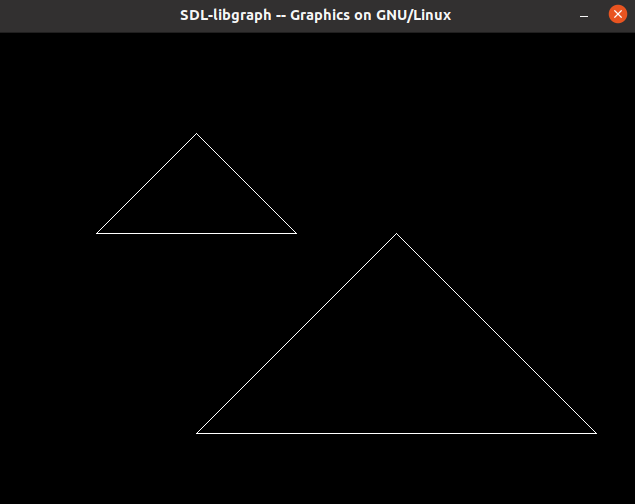
scale(x, y, sx,sy);

delay(100000);

return 0;

}

**OUTPUT**

****

**Program 8**

Write a program to rotate a triangle about any one of its end coordinates.

**Program:**

#include<iostream>

#include<graphics.h>

#include<math.h>

using namespace std;

int main()

{

double s,c, angle;

int gd = DETECT,gm,x1,y1,x2,y2,x3,y3;

initgraph(&gd, &gm,NULL);

cout << "Enter Coordinates of Triangle: ";

cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;

line(x1,y1,x2,y2);

line(x2,y2, x3,y3);

line(x3, y3, x1, y1);

cout << "Enter Rotation Angle: ";

cin >> angle;

c = cos(angle \*M\_PI/180);

s = sin(angle \*M\_PI/180);

x1 = floor(x1 \* c + y1 \* s);

y1 = floor(-x1 \* s + y1 \* c);

x2 = floor(x2 \* c + y2 \* s);

y2 = floor(-x2 \* s + y2 \* c);

x3 = floor(x3 \* c + y3 \* s);

y3 = floor(-x3 \* s + y3 \* c);

line(x1, y1 ,x2, y2);

line(x2,y2, x3,y3);

line(x3, y3, x1, y1);

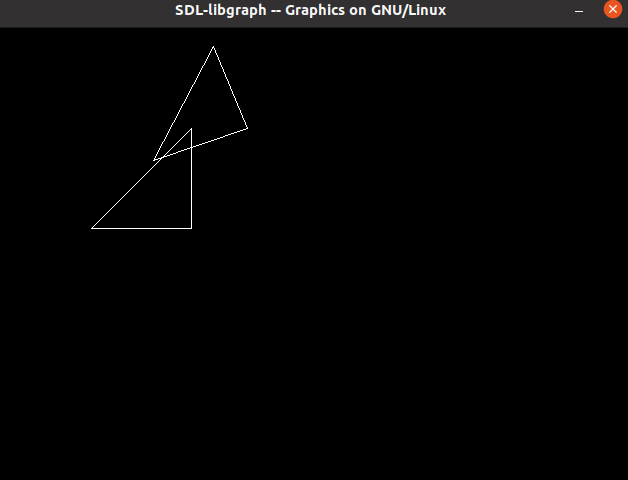
delay(100000);

closegraph();

return 0;

}

**OUTPUT**

****

**Program 9**

Write program to draw a house like figure and perform the following operations.

a)Scaling about the origin followed by translation.

b)Scaling with reference to an arbitrary point