**Experiment No: 01**

**Experiment Name:** Write a program to draw a line using direct equation

**Objective:** To write a program to draw a line using direct equation

**Algorithm:**

1. Input the two line end-points, storing the left end-point in (x1, y1)

2.Set x= x1 and y=y1

3.Draw a pixel at (x, y) location in the output device.

4.Increase x by 1.

5.Compute y from line equation y=mx+b

6.Draw a pixel at (x, y) location in the output device.

7.Repeat step 4 to step 6 till x>x2.

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*int CLR=10;*

*void LineCreate(int x1, int y1, int x2, int y2){*

*if(x1==x2){*

*for(int y=y1; y<=y2; y++)*

*putpixel((int)x1,y,CLR);*

*return;*

*}*

*double m=(y1-y2)\*1.0/(x1-x2);*

*double c=y1-m\*x1;*

*if(abs(x1-x2)>=abs(y1-y2)){*

*for(int x=x1; x<=x2; x++){*

*int y=round(m\*x+c);*

*putpixel(x,y,CLR);*

*}*

*}*

*else{*

*for(int y=y1; y<=y2; y++){*

*int x=round((y-c)/m);*

*putpixel(x,y,CLR);*

*}*

*}*

*}*

*int main(){*

*int gmode=DETECT,gdriver;*

*initgraph(&gmode,&gdriver,"");*

*int x1=100,y1=100,x2=300,y2=300;*

*LineCreate(x1,y1,x2,y2);*

*getch();*

*closegraph();*

*return 0;*

*}*

**Experiment No: 02**

**Experiment Name:** Write a program to draw a line using Brasenham's Algorithm

**Objective:** To write a program to draw a line using Brasenham's Algorithm

**Algorithm:**

1. Input the two line end-points, storing the left end-point in (x0, y0)

2. Plot the point (x0, y0)

3. Calculate the constants Δx, Δy, 2Δy, and (2Δy - 2Δx) and get the first value for the decisionparameter as:

d0 = 2 \* dY - dX

4. At each xk along the line, starting at i = 0, perform the following test. If di< 0, the next point to plot is (xi+1, yi ) and:

di+1=di +2 Δy

Otherwise, the next point to plot is (xi+1, yi+1) and:

di+1= di +2Δy - 2Δx

5. Repeat step 4 (Δx – 1) times

**Source code:**

*#include<graphics.h>*

*using namespace std;*

*int CLR=15;*

*void BresenhamsLine(int x1, int y1, int x2, int y2){*

*putpixel(x1,y1,CLR);*

*int dx=x2-x1;*

*int dy=y2-y1;*

*int d=2\*dy-dx;*

*int dT=2\*(dy-dx);*

*int dS=2\*dy;*

*while(x1<x2){*

*x1++;*

*if(d<0)*

*d+=dS;*

*else{*

*d+=dT;*

*y1++;*

*}*

*putpixel(x1,y1,CLR);*

*}*

*}*

*int main(){*

*int gmode=DETECT, gdriver;*

*initgraph(&gmode, &gdriver,"");*

*int x1=100,y1=100,x2=200,y2=200;*

*BresenhamsLine(x1,y1,x2,y2);*

*getch();*

*closegraph();*

*return 0;*

*}*

**Experiment No: 03**

**Experiment Name:** Write a program to draw a circle using direct equation

**Objective:** To write a program to draw a circle using direct equation

**Algorithm:**

Step 1. Set x to 0 and y = r;

Step 2. Increase x by 1, compute y, draw 8 points using symmetry as

P5= (-x, -y)

P6= (-y, -x)

P7= (y, -x)

P8= (x, -y)

P1= (x, y)

P2= (y, x)

P3= (-y, x)

P4= (-x, y)

Step 3. Repeat step 2 until x<r/√2

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*int CLR=10;*

*void CircleDirect(int r){*

*int x=0,y=r;*

*int tr=r+10;*

*putpixel(x,y,CLR);*

*while(x<=y){*

*x++;*

*y=round(sqrt(r\*r-x\*x));*

*putpixel(x+tr,y+tr,CLR);*

*putpixel(-x+tr,y+tr,CLR);*

*putpixel(x+tr,-y+tr,CLR);*

*putpixel(-x+tr,-y+tr,CLR);*

*putpixel(y+tr,x+tr,CLR);*

*putpixel(-y+tr,x+tr,CLR);*

*putpixel(y+tr,-x+tr,CLR);*

*putpixel(-y+tr,-x+tr,CLR);*

*}*

*}*

*int main(){*

*int gmode=DETECT, gdriver;*

*initgraph(&gmode, &gdriver, "");*

*int r=100;*

*CircleDirect(r);*

*getch();*

*closegraph();*

*return 0;*

*}*

**Experiment No: 04**

**Experiment Name:** Write a program to draw a circle using Brasenham's Algorithm.

**Objective:** To write a program to draw a circle using Brasenham's Algorithm.

**Algorithm:**

1. Read radius r.

2.Set x=0 ,y=r and decision variable d= 3-2\*r

3. If d<0 Then

x= x+ 1

d= d+ 4\*x +6

otherwise

y= y-1

x=x+1

p=p+4\*(x-y) +10

4. putpixel(x, y, color)

5. Repeat steps 3 and 4 until x<= y

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*int CLR = 15;*

*void CircleBresenham(int r){*

*int x = 0, y = r;*

*int tr = r + 10;*

*int d = 3 - 2 \* r;*

*while(x <= y){*

*putpixel(x + tr, y + tr, CLR);*

*putpixel(-x + tr, y + tr, CLR);*

*putpixel(x + tr, -y + tr, CLR);*

*putpixel(-x + tr, -y + tr, CLR);*

*putpixel(y + tr, x + tr, CLR);*

*putpixel(-y + tr, x + tr, CLR);*

*putpixel(y + tr, -x + tr, CLR);*

*putpixel(-y + tr, -x + tr, CLR);*

*x++;*

*if(d<0) d += 4\*x + 6;*

*else {*

*d += 4\*(x-y) + 10;*

*y--;*

*}*

*}*

*}*

*int main(){*

*int gmode = DETECT, gdriver;*

*initgraph ( &gmode, &gdriver, "" );*

*int r = 100;*

*CircleBresenham(r);*

*getch();*

*closegraph();*

*cout<<"finished"<<endl;*

*}*

**Experiment No: 05**

**Experiment Name:** Write a program to draw a circle using mid point Algorithm.

**Objective:** To write a program to draw a circle using mid point Algorithm.

**Algorithm:**

Step 1: Start.

Step 2: Declare x, y, r, xc , yc , P as variables, where (xc , yc) are coordinates of the center.

Step 3: Put x = 0 and y = r

Step 4: Repeat the steps while x ≤ y;

Step 5: Plot (x, y).

Step 6: if (P < 0):

Set P = P + 2x + 3

else if (P >= 0):

Set P = P + 2(x-y) + 5

y = y - 1

Step 7: Do x = x + 1

Step 8: End

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*int CLR = 15;*

*void CircleMidPoint(int r){*

*int x = 0, y = r;*

*int tr = r + 10;*

*int p = 1 - r;*

*while(x <= y){*

*putpixel(x + tr, y + tr, CLR);*

*putpixel(-x + tr, y + tr, CLR);*

*putpixel(x + tr, -y + tr, CLR);*

*putpixel(-x + tr, -y + tr, CLR);*

*putpixel(y + tr, x + tr, CLR);*

*putpixel(-y + tr, x + tr, CLR);*

*putpixel(y + tr, -x + tr, CLR);*

*putpixel(-y + tr, -x + tr, CLR);*

*x++;*

*if(p<0) p += 2\*x + 3;*

*else {*

*p += 2\*(x-y) + 5;*

*y--;*

*}*

*}*

*}*

*int main(){*

*int gmode = DETECT, gdriver;*

*initgraph ( &gmode, &gdriver, "" );*

*int r = 100;*

*CircleMidPoint(r);*

*getch();*

*closegraph();*

*cout<<"finished"<<endl;*

*}*

**Experiment No: 06**

**Experiment Name:** Write a program to implement Flood Fill Algorithm

**Objective:** To write a program to implement Flood Fill Algorithm

**Algorithm:**

Flood-fill (node, target-color, replacement-color):

1. Set Q to the empty queue.

2. If the color of node is not equal to target-color, return.

3. Add node to the end of Q.

4. For each element n of Q:

5. Set the color of n to replacement-color.

6. If the color of the node to the west of n is target-color, add that node to the end of Q.

If the color of the node to the east of n is target-color, add that node to the end of Q.

If the color of the node to the north of n is target-color, add that node to the end of Q.

If the color of the node to the south of n is target-color, add that node to the end of Q.

7. Continue looping until Q is exhausted.

8. Return.

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*vector<pair<int,int>>points;*

*int borderColor=15;*

*void Flood\_Fill(int x,int y, int clr){*

*int ColorGot=getpixel(x,y);*

*if(ColorGot==clr||ColorGot==borderColor)return;*

*putpixel(x,y,clr);*

*Flood\_Fill(x+1,y,clr);*

*Flood\_Fill(x-1,y,clr);*

*Flood\_Fill(x,y+1,clr);*

*Flood\_Fill(x,y-1,clr);*

*}*

*void Make\_Polygon(){*

*for(int i=1; i<points.size();i++)*

*line(points[i-1].first,points[i-1].second,points[i].first,points[i].second);*

*line(points[0].first,points[0].second,points[points.size()-1].first, points[points.size()-1].second);*

*}*

*int main(){*

*int gmode=DETECT, gdriver;*

*initgraph(&gmode, &gdriver,"");*

*int n,x,y;*

*cout<<"points number ";*

*cin>>n;*

*while(n<3){*

*cout<<"Invalid again input";*

*cin>>n;*

*}*

*for(int i=0; i<n; i++){*

*cin>>x>>y;*

*points.push\_back({x,y});*

*}*

*Make\_Polygon();*

*cout<<"Input a point ";*

*cin>>x>>y;*

*/// floodfill(x,y,15); ///build in*

*Flood\_Fill(x,y,5);*

*getch();*

*closegraph();*

*return 0;*

*}*

**Experiment No: 07**

**Experiment Name:** Write a program to implement Object Transformation

**Objective:** To write a program to implement Object Transformation

**Algorithm:**

Step 1: Start the program.

Step 2: Input the object coordinates depending on objects.

Step 3:Enter the translation factors tx and ty

Step 3:ADD translation factors tx to x coordinates and ty to y coordinate and Move the original coordinates position to new position. .

Step 4: Generate the output.

Step 5: Stop the program.

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*void WorkWithLine(int x1,int y1, int x2, int y2, int tx, int ty){*

*setcolor(YELLOW);*

*line(x1,y1,x2,y2);*

*setcolor(WHITE);*

*line(x1+tx,y1+ty,x2+tx,y2+ty);*

*}*

*void WorkWithCircle(int r, int tx, int ty){*

*int origin=r+10;*

*setcolor(YELLOW);*

*circle(origin,origin,r);*

*setcolor(WHITE);*

*circle(origin+tx, origin+ty, r);*

*}*

*void WorkWithTriangle(int x1, int y1, int x2, int y2, int x3, int y3,int tx, int ty){*

*WorkWithLine(x1,y1,x2,y2,tx,ty);*

*WorkWithLine(x2,y2,x3,y3,tx,ty);*

*WorkWithLine(x1,y1,x3,y3,tx,ty);*

*}*

*void WorkWithRectangle(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4,int tx, int ty){*

*WorkWithLine(x1,y1,x2,y2,tx,ty);*

*WorkWithLine(x2,y2,x3,y3,tx,ty);*

*WorkWithLine(x4,y4,x3,y3,tx,ty);*

*WorkWithLine(x1,y1,x4,y4,tx,ty);*

*}*

*int main(){*

*int gmode=DETECT, gdrive;*

*initgraph(&gmode,&gdrive,"");*

*while(1){*

*system("cls");*

*cout<<"1. Line"<<endl<<"2.Circle"<<endl<<"3. triangle"<<endl<<"4.Rectangle"<<endl<<"5.Exit"<<endl;*

*cout<<"enter your choice "<<endl;*

*int n;*

*cin>>n;*

*cleardevice();*

*if(n==1){*

*int x1,x2,y1,y2,tx,ty;*

*cout<<"Input two points";*

*cin>>x1>>y1>>x2>>y2;*

*cout<<"Input transformation value ";*

*cin>>tx>>ty;*

*WorkWithLine(x1,y1,x2,y2,tx,ty);*

*}*

*if(n==2){*

*int r,tx,ty;*

*cout<<"Input the radius";*

*cin>>r;*

*cout<<"Input transformation value ";*

*cin>>tx>>ty;*

*WorkWithCircle(r,tx,ty);*

*}*

*if(n==3){*

*int x1,x2,y1,y2,x3,y3,tx,ty;*

*cout<<"Input three points";*

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

*cout<<"Input transformation value ";*

*cin>>tx>>ty;*

*WorkWithTriangle(x1,y1,x2,y2,x3,y3,tx,ty);*

*}*

*if(n==4){*

*int x1,x2,y1,y2,x3,y3,x4,y4,tx,ty;*

*cout<<"Input four points";*

*cin>>x1>>y1>>x2>>y2>>x3>>y3>>x4>>y4;*

*cout<<"Input transformation value ";*

*cin>>tx>>ty;*

*WorkWithRectangle(x1,y1,x2,y2,x3,y3,x4,y4,tx,ty);*

*}*

*}*

*closegraph();*

*getch();*

*return 0;*

*}*

**Experiment No: 08**

**Experiment Name:** Write a program to implement Object rotation (2D)

**Objective:** To write a program to implement Object rotation (2D)

**Algorithm:**

Step 1: Start the program.

Step 2: Input the object coordinates depending on objects.

Step 3: Enter the radian value ra

Step 4: Using the ra calculate theta value (float)(3.14xra) 180.

Step 5:From the theta value calculate the x..p.q values.

Step 6:Rotate the original coordinate position

Step 7: Generate the output.

Step 8: Stop the program.

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*int midx=319, midy=239;*

*int maxx=638, maxy=479;*

*void AxisMaker(){*

*line(0,midy,maxx,midy);*

*line(midx,0,midx,maxy);*

*}*

*void WorkWithLine(int x1, int y1, int x2, int y2, int degree){*

*setcolor(YELLOW);*

*line(x1+midx,y1+midy,x2+midx,y2+midy);*

*setcolor(WHITE);*

*double radian=degree\*3.1416/180.0;*

*int x11=round(x1\*cos(radian)-y1\*sin(radian))+midx;*

*int y11=round(y1\*cos(radian)+x1\*sin(radian))+midy;*

*int x22=round(x2\*cos(radian)-y2\*sin(radian))+midx;*

*int y22=round(y2\*cos(radian)+x2\*sin(radian))+midy;*

*line(x11,y11,x22,y22);*

*}*

*void WorkWithTriangle(int x1, int y1, int x2, int y2, int x3, int y3,int degree){*

*WorkWithLine(x1,y1,x2,y2,degree);*

*WorkWithLine(x2,y2,x3,y3,degree);*

*WorkWithLine(x1,y1,x3,y3,degree);*

*}*

*void WorkWithRectangle(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4,int degree){*

*WorkWithLine(x1,y1,x2,y2,degree);*

*WorkWithLine(x2,y2,x3,y3,degree);*

*WorkWithLine(x4,y4,x3,y3,degree);*

*WorkWithLine(x1,y1,x4,y4,degree);*

*}*

*int main(){*

*int gmode=DETECT, gdrive;*

*initgraph(&gmode,&gdrive,"");*

*while(1){*

*AxisMaker();*

*system("cls");*

*cout<<"1. Line"<<endl<<"3. triangle"<<endl<<"4.Rectangle"<<endl<<"5.Exit"<<endl;*

*cout<<"enter your choice "<<endl;*

*int n;*

*cin>>n;*

*cleardevice();*

*AxisMaker();*

*if(n==1){*

*int x1,x2,y1,y2,theta;*

*cout<<"Input two points";*

*cin>>x1>>y1>>x2>>y2;*

*cout<<"Input rotation value ";*

*cin>>theta;*

*WorkWithLine(x1,y1,x2,y2,theta);*

*}*

*if(n==3){*

*int x1,x2,y1,y2,x3,y3,theta;*

*cout<<"Input three points";*

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

*cout<<"Input rotation value ";*

*cin>>theta;*

*WorkWithTriangle(x1,y1,x2,y2,x3,y3,theta);*

*}*

*if(n==4){*

*int x1,x2,y1,y2,x3,y3,x4,y4,theta;*

*cout<<"Input four points";*

*cin>>x1>>y1>>x2>>y2>>x3>>y3>>x4>>y4;*

*cout<<"Input rotation value ";*

*cin>>theta;*

*WorkWithRectangle(x1,y1,x2,y2,x3,y3,x4,y4,theta);*

*}*

*}*

*closegraph();*

*getch();*

*return 0;*

*}*

**Experiment No: 09**

**Experiment Name:** Write a program to implement Object scaling(2D)

**Objective:** To write a program to implement Object scaling(2D)

**Algorithm:**

Step 1: Start the program.

Step 2: Input the object coordinates depending on objects.

Step 3:Enter the scaling factors Sx and Sy

Step 3:ADD scaling factors Sx to x coordinate and Sy to y coordinate and Move the original coordinates position to new position.

Step 4: Generate the output.

Step 5: Stop the program.

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*int midx=319, midy=239;*

*int maxx=638, maxy=479;*

*void AxisMaker(){*

*line(0,midy,maxx,midy);*

*line(midx,0,midx,maxy);*

*}*

*void WorkWithLine(int x1, int y1, int x2, int y2, double sx, double sy){*

*setcolor(YELLOW);*

*line(x1+midx,y1+midy,x2+midx,y2+midy);*

*setcolor(WHITE);*

*int x11=round(x1\*sx)+midx;*

*int y11=round(y1\*sy)+midy;*

*int x22=round(x2\*sx)+midx;*

*int y22=round(y2\*sy)+midy;*

*line(x11,y11,x22,y22);*

*}*

*void WorkWithTriangle(int x1, int y1, int x2, int y2, int x3, int y3,double sx, double sy){*

*WorkWithLine(x1,y1,x2,y2,sx,sy);*

*WorkWithLine(x2,y2,x3,y3,sx,sy);*

*WorkWithLine(x3,y3,x1,y1,sx,sy);*

*}*

*void WorkWithRectangle(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4,double sx, double sy){*

*WorkWithLine(x1,y1,x2,y2,sx,sy);*

*WorkWithLine(x2,y2,x3,y3,sx,sy);*

*WorkWithLine(x3,y3,x4,y4,sx,sy);*

*WorkWithLine(x4,y4,x1,y1,sx,sy);*

*}*

*int main(){*

*int gmode=DETECT, gdrive;*

*initgraph(&gmode,&gdrive,"");*

*while(1){*

*AxisMaker();*

*system("cls");*

*cout<<"1. Line"<<endl<<"3. triangle"<<endl<<"4.Rectangle"<<endl<<"5.Exit"<<endl;*

*cout<<"enter your choice "<<endl;*

*int n;*

*cin>>n;*

*cleardevice();*

*AxisMaker();*

*if(n==1){*

*int x1,x2,y1,y2;*

*double sx,sy;*

*cout<<"Input two points";*

*cin>>x1>>y1>>x2>>y2;*

*cout<<"Input scaling value ";*

*cin>>sx>>sy;*

*WorkWithLine(x1,y1,x2,y2,sx,sy);*

*}*

*if(n==3){*

*int x1,x2,y1,y2,x3,y3;*

*double sx,sy;*

*cout<<"Input three points";*

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

*cout<<"Input scaling value ";*

*cin>>sx>>sy;*

*WorkWithTriangle(x1,y1,x2,y2,x3,y3,sx,sy);*

*}*

*if(n==4){*

*int x1,x2,y1,y2,x3,y3,x4,y4;*

*double sx,sy;*

*cout<<"Input four points";*

*cin>>x1>>y1>>x2>>y2>>x3>>y3>>x4>>y4;*

*cout<<"Input scaling value ";*

*cin>>sx>>sy;*

*WorkWithRectangle(x1,y1,x2,y2,x3,y3,x4,y4,sx,sy);*

*}*

*}*

*closegraph();*

*getch();*

*return 0;*

*}*

**Experiment No: 10**

**Experiment Name:** Write a program to implement Object mirroring (2D)

**Objective:** To write a program to implement Object mirroring (2D).

**Algorithm:**

Step 1: Start the program.

Step 2: Input the object coordinates depending on objects

Step 3: Input the axis of reflection.

Step 4: If axis is x then

x1 = -x1;

x2 = -x2;

Step 5: If axis is y then

y1 = -y1;

y2 = -y2;

Step 4: Generate the output.

Step 5: Stop the program.

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*int midx=319, midy=239;*

*int maxx=638, maxy=479;*

*void AxisMaker(){*

*line(0,midy,maxx,midy);*

*line(midx,0,midx,maxy);*

*}*

*void WorkWithLine(int x1, int y1, int x2, int y2, int axis){*

*setcolor(YELLOW);*

*line(x1+midx,y1+midy,x2+midx,y2+midy);*

*setcolor(WHITE);*

*if(axis==1){*

*x1=-x1;*

*x2=-x2;*

*}*

*else{*

*y1=-y1;*

*y2=-y2;*

*}*

*line(x1+midx,y1+midy,x2+midx,y2+midy);*

*}*

*void WorkWithTriangle(int x1, int y1, int x2, int y2, int x3, int y3,int axis){*

*WorkWithLine(x1,y1,x2,y2,axis);*

*WorkWithLine(x2,y2,x3,y3,axis);*

*WorkWithLine(x3,y3,x1,y1,axis);*

*}*

*void WorkWithRectangle(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4,int axis){*

*WorkWithLine(x1,y1,x2,y2,axis);*

*WorkWithLine(x2,y2,x3,y3,axis);*

*WorkWithLine(x3,y3,x4,y4,axis);*

*WorkWithLine(x4,y4,x1,y1,axis);*

*}*

*int main(){*

*int gmode=DETECT, gdrive;*

*initgraph(&gmode,&gdrive,"");*

*while(1){*

*AxisMaker();*

*system("cls");*

*cout<<"1. Line"<<endl<<"3. triangle"<<endl<<"4.Rectangle"<<endl<<"5.Exit"<<endl;*

*cout<<"enter your choice "<<endl;*

*int n;*

*cin>>n;*

*cleardevice();*

*AxisMaker();*

*if(n==1){*

*int x1,x2,y1,y2;*

*int axis;*

*cout<<"Input two points";*

*cin>>x1>>y1>>x2>>y2;*

*cout<<"Input Mirorring value ";*

*cin>>axis;*

*WorkWithLine(x1,y1,x2,y2,axis);*

*}*

*if(n==3){*

*int x1,x2,y1,y2,x3,y3;*

*int axis;*

*cout<<"Input three points";*

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

*cout<<"Input Mirorring value ";*

*cin>>axis;*

*WorkWithTriangle(x1,y1,x2,y2,x3,y3,axis);*

*}*

*if(n==4){*

*int x1,x2,y1,y2,x3,y3,x4,y4;*

*int axis;*

*cout<<"Input four points";*

*cin>>x1>>y1>>x2>>y2>>x3>>y3>>x4>>y4;*

*cout<<"Input Mirorring value ";*

*cin>>axis;*

*WorkWithRectangle(x1,y1,x2,y2,x3,y3,x4,y4,axis);*

*}*

*}*

*closegraph();*

*getch();*

*return 0;*

*}*

**Experiment No: 11**

**Experiment Name:** Write a program to implement Object Shearing(2D)

**Objective:** To write a program to implement Object Shearing(2D).

**Algorithm:**

Step 1: Start the program.

Step 2: Input the object coordinates depending on objects.

Step 3: To shear the object using the formula

1. Along x-direction

y=y, x=x+(xls\*y),

1. Along y-direction

x=x, y=y+(yls\*x),

Step 4: Generate the output.

Step 5: Stop the program.

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*int midx=319, midy=239;*

*int maxx=638, maxy=479;*

*void AxisMaker(){*

*line(0,midy,maxx,midy);*

*line(midx,0,midx,maxy);*

*}*

*void WorkWithLine(int x1, int y1, int x2, int y2, int axis, double s){*

*setcolor(YELLOW);*

*line(x1+midx,y1+midy,x2+midx,y2+midy);*

*setcolor(WHITE);*

*if(axis==1){*

*x1=x1+s\*y1;*

*x2=x2+s\*y2;*

*}*

*else{*

*y1=y1+s\*x1;*

*y2=y2+s\*x2;*

*}*

*line(x1+midx,y1+midy,x2+midx,y2+midy);*

*}*

*void WorkWithTriangle(int x1, int y1, int x2, int y2, int x3, int y3,int axis,double s){*

*WorkWithLine(x1,y1,x2,y2,axis,s);*

*WorkWithLine(x2,y2,x3,y3,axis,s);*

*WorkWithLine(x3,y3,x1,y1,axis,s);*

*}*

*void WorkWithRectangle(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4,int axis,double s){*

*WorkWithLine(x1,y1,x2,y2,axis,s);*

*WorkWithLine(x2,y2,x3,y3,axis,s);*

*WorkWithLine(x3,y3,x4,y4,axis,s);*

*WorkWithLine(x4,y4,x1,y1,axis,s);*

*}*

*int main(){*

*int gmode=DETECT, gdrive;*

*initgraph(&gmode,&gdrive,"");*

*while(1){*

*AxisMaker();*

*system("cls");*

*cout<<"1. Line"<<endl<<"3. triangle"<<endl<<"4.Rectangle"<<endl<<"5.Exit"<<endl;*

*cout<<"enter your choice "<<endl;*

*int n;*

*cin>>n;*

*cleardevice();*

*AxisMaker();*

*if(n==1){*

*int x1,x2,y1,y2;*

*int axis;*

*double s;*

*cout<<"Input two points";*

*cin>>x1>>y1>>x2>>y2;*

*cout<<"Input 1 for x axis and 2 for y axis ";*

*cin>>axis;*

*cout<<"input shearing value: ";*

*cin>>s;*

*WorkWithLine(x1,y1,x2,y2,axis,s);*

*}*

*if(n==3){*

*int x1,x2,y1,y2,x3,y3;*

*int axis;*

*double s;*

*cout<<"Input three points";*

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

*cout<<"Input 1 for x axis and 2 for y axis ";*

*cin>>axis;*

*cout<<"input shearing value: ";*

*cin>>s;*

*WorkWithTriangle(x1,y1,x2,y2,x3,y3,axis,s);*

*}*

*if(n==4){*

*int x1,x2,y1,y2,x3,y3,x4,y4;*

*int axis;*

*double s;*

*cout<<"Input four points";*

*cin>>x1>>y1>>x2>>y2>>x3>>y3>>x4>>y4;*

*cout<<"Input 1 for x axis and 2 for y axis ";*

*cin>>axis;*

*cout<<"input shearing value: ";*

*cin>>s;*

*WorkWithRectangle(x1,y1,x2,y2,x3,y3,x4,y4,axis,s);*

*}*

*}*

*closegraph();*

*getch();*

*return 0;*

*}*

**Experiment No: 12**

**Experiment Name:** Write a program to implement Object rotation (3D)

**Objective:** To write a program to implement Object rotation (3D)

**Algorithm:**

Step 1: Start the program.

Step 2: Input the object coordinates depending on objects.

Step 3: Enter the radian value ra

Step 4: Using the ra calculate theta value (float)(3.14xra) 180.

Step 5:From the theta value calculate the x1,y1,x2,y2 values.

Step 6:Rotate the original coordinate position

Step 7: Generate the output.

Step 8: Stop the program.

**Source code:**

*#include<bits/stdc++.h>*

*#include<graphics.h>*

*using namespace std;*

*int midx = 319, midy = 239;*

*int maxx = 638, maxy = 479;*

*void AxisMaker(){*

*line(0,midy,maxx,midy);*

*line(midx,0,midx,maxy);*

*}*

*void WorkWithLine(int x1, int y1, int z1, int x2, int y2, int z2, int Axis, int Degree){ ///may be problem exist in y and z axis.*

*setcolor(YELLOW);*

*line(x1 + midx, y1 + midy, x2 + midx, y2 + midy);*

*setcolor(WHITE);*

*double radian = Degree \* 3.1416 / 180.0;*

*int x11, y11, z11, x22, y22, z22;*

*if(Axis == 3){*

*x11 = round(x1 \* cos(radian) - y1 \* sin(radian)) + midx;*

*y11 = round(x1 \* sin(radian) + y1 \* cos(radian)) + midy;*

*z11 = z1;*

*x22 = round(x2 \* cos(radian) - y2 \* sin(radian)) + midx;*

*y22 = round(x2 \* sin(radian) + y2 \* cos(radian)) + midy;*

*z22 = z2;*

*}*

*else if(Axis == 2){*

*x11 = round(x1 \* cos(radian) + z1 \* sin(radian)) + midx;*

*y11 = y1 + midy;*

*z11 = round(-x1 \* sin(radian) + z1 \* cos(radian));*

*x22 = round(x2 \* cos(radian) + z2 \* sin(radian)) + midx;*

*y22 = y2 + midy;*

*z22 = round(-x2 \* sin(radian) + z2 \* cos(radian));*

*}*

*else if(Axis == 1){*

*x11 = x1 + midx;*

*y11 = round(y1 \* cos(radian) - z1 \* sin(radian)) + midy;*

*z11 = round(y1 \* sin(radian) + z1 \* cos(radian));*

*x22 = x2 + midx;*

*y22 = round(y2 \* cos(radian) - z2 \* sin(radian)) + midy;*

*z22 = round(y2 \* sin(radian) + z2 \* cos(radian));*

*}*

*// cout<<x11 - midx<<" "<<y11 - midy<<" "<<z11<<" "<<x22 - midx<<" "<<y22 - midy<<" "<<z22<<endl;*

*line(x11, y11, x22, y22);*

*}*

*void WorkWithTriangle(int x1, int y1, int z1, int x2, int y2, int z2, int x3, int y3, int z3, int Axis, int Degree){*

*WorkWithLine(x1, y1, z1, x2, y2, z2, Axis, Degree);*

*WorkWithLine(x1, y1, z1, x3, y3, z3, Axis, Degree);*

*WorkWithLine(x3, y3, z3, x2, y2, z2, Axis, Degree);*

*}*

*void WorkWithRectangle(int x1, int y1, int z1, int x2, int y2, int z2, int x3, int y3, int z3, int x4, int y4, int z4, int Axis, int Degree){*

*WorkWithLine(x1, y1, z1, x2, y2, z2, Axis, Degree);*

*WorkWithLine(x2, y2, z2, x3, y3, z3, Axis, Degree);*

*WorkWithLine(x3, y3, z3, x4, y4, z4, Axis, Degree);*

*WorkWithLine(x1, y1, z1, x4, y4, z4, Axis, Degree);*

*}*

*void WorkWithCube(int x1, int y1, int z1, int x2, int y2, int z2, int x3, int y3, int z3, int x4, int y4, int z4, int x5, int y5, int z5, int x6, int y6, int z6, int x7, int y7, int z7, int x8, int y8, int z8, int Axis, int Degree){*

*WorkWithRectangle(x1, y1 , z1, x2, y2, z2, x3, y3, z3, x4, y4, z4, Axis, Degree);*

*WorkWithRectangle(x5, y5 , z5, x6, y6, z6, x7, y7, z7, x8, y8, z8, Axis, Degree);*

*WorkWithRectangle(x1, y1 , z1, x5, y5, z5, x8, y8, z8, x4, y4, z4, Axis, Degree);*

*WorkWithRectangle(x2, y2 , z2, x6, y6, z6, x7, y7, z7, x3, y3, z3, Axis, Degree);*

*}*

*int main(){*

*int gmode = DETECT, gdriver;*

*int Axis;*

*initgraph ( &gmode, &gdriver, "" );*

*while(1){*

*AxisMaker();*

*system("cls");*

*cout<<"1. Line"<<endl;*

*cout<<"2. Triangle"<<endl;*

*cout<<"3. Rectangle"<<endl;*

*cout<<"4. Cube"<<endl;*

*cout<<"5. Exit"<<endl;*

*int n;*

*cout<<endl<<"Select Your Choich: ";*

*cin>>n;*

*cleardevice();*

*AxisMaker();*

*if(n == 1){*

*int x1, x2, y1, y2, z1, z2, Degree;*

*cout<<"Input your two points (x1, y1, z1) (x2, y2, z2): ";*

*cin>>x1>>y1>>z1>>x2>>y2>>z2;*

*cout<<"Input your Rotation value (Degree): ";*

*cin>>Degree;*

*cout<<"In which axis you want to rotate (1=>x, 2=>y, 3=>z): ";*

*cin>>Axis;*

*WorkWithLine(x1, y1, z1, x2, y2, z2, Axis, Degree);*

*}*

*else if(n == 2){*

*int x1, y1, x2, y2, x3, y3, z1, z2, z3, Degree;*

*cout<<"Input three points of your triangle (x1, y1, z1), (x2, y2, z2), (x3, y3, z3): ";*

*cin>>x1>>y1>>z1>>x2>>y2>>z2>>x3>>y3>>z3;*

*cout<<"Input your Rotation value (Degree): ";*

*cin>>Degree;*

*cout<<"In which axis you want to rotate (1=>x, 2=>y, 3=>z): ";*

*cin>>Axis;*

*WorkWithTriangle(x1, y1, z1, x2, y2, z2, x3, y3, z3, Axis, Degree);*

*}*

*else if(n == 3){*

*int x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4, Degree;*

*cout<<"Input Four points of your rectangle (x1, y1, z1), (x2, y2, z2), (x3, y3, z3), (x4, y4, z4): ";*

*cin>>x1>>y1>>z1>>x2>>y2>>z2>>x3>>y3>>z3>>x4>>y4>>z4;*

*cout<<"Input your Rotation value (Degree): ";*

*cin>>Degree;*

*cout<<"In which axis you want to rotate (1=>x, 2=>y, 3=>z): ";*

*cin>>Axis;*

*WorkWithRectangle(x1, y1 , z1, x2, y2, z2, x3, y3, z3, x4, y4, z4, Axis, Degree);*

*}*

*else if(n == 4){*

*int x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4, Degree;*

*cout<<"Input Four points of your rectangle (x1, y1, z1), (x2, y2, z2), (x3, y3, z3), (x4, y4, z4): ";*

*cin>>x1>>y1>>z1>>x2>>y2>>z2>>x3>>y3>>z3>>x4>>y4>>z4;*

*cout<<"Input your Rotation value (Degree): ";*

*cin>>Degree;*

*cout<<"In which axis you want to rotate (1=>x, 2=>y, 3=>z): ";*

*cin>>Axis;*

*int t = 20;*

*WorkWithCube(x1, y1 , z1, x2, y2, z2, x3, y3, z3, x4, y4, z4, x1 + t, y1 + t, z1 + t, x2 + t, y2 + t, z2 + t, x3 + t, y3 + t, z3 + t, x4 + t, y4 + t, z4 + t, Axis, Degree);*

*}*

*else if(n == 5){*

*exit(0);*

*}*

*}*

*closegraph();*

*cout<<"finished"<<endl;*

*}*

**Experiment No: 13**

**Experiment Name:** Write a program to implement Waller Algorithm

**Objective:** To write a program to implement Waller Algorithm

**Source code:**

*#include <iostream>*

*#include<dos.h>*

*#include<graphics.h>*

*using namespace std;*

*int main()*

*{*

*int i,gd=DETECT,gm;*

*int x, y, c=4;*

*initgraph(&gd,&gm,"");*

*rectangle(100,100, 300,300);*

*line(120,250,120,50);*

*line(120,50,180,200);*

*line(180,200,280,50);*

*line(280,50,280,250);*

*line(280,250,120,250);*

*x=120;*

*y=200;*

*putpixel(x,y,4);*

*y=y-1;*

*outtextxy(x,y,"Start");*

*while(1)*

*{*

*putpixel(x,y,c);*

*if(getpixel(x-1,y+1)==15)*

*{*

*x=x-1;*

*y=y+1;*

*}*

*else if(getpixel(x,y+1)==15)*

*{*

*y=y+1;*

*}*

*else if(getpixel(x+1,y+1)==15)*

*{*

*x=x+1;*

*y=y+1;*

*}*

*else if(getpixel(x+1,y)==15)*

*{*

*x=x+1;*

*}*

*else if(getpixel(x+1,y-1)==15) {*

*x=x+1;*

*y=y-1;*

*}*

*else if(getpixel(x,y-1)==15){ ;*

*y=y-1;*

*}*

*else if(getpixel(x,y-2)==15)*

*{*

*;*

*y=y-2;*

*}*

*else if(getpixel(x-1,y-1)==15)*

*{*

*x=x-1;*

*y=y-1;*

*}*

*else if(getpixel(x-1,y)==15)*

*{*

*x=x-1;*

*y=y;*

*}*

*else if(getpixel(x-2,y)==15)*

*{*

*x=x-2;*

*y=y;*

*}*

*else break;*

*delay(20);*

*}*

*outtextxy(x,y+2,"End");*

*cin>>i;*

*return 0;*

*}*