//clipping

#include<iostream>

#include<cstdlib>

#include<graphics.h>

using namespace std;

static int LEFT=1,RIGHT=2,BOTTOM=4,TOP=8,xmin,ymin,xmax,ymax;

int getcode(int x,int y){

int code=0;

if(y>ymax) code|=TOP;

if(y<ymin) code|=BOTTOM;

if(x<xmin) code|=LEFT;

if(x>xmax) code|=RIGHT;

return code;

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,NULL);

cout<<"\nEnter coordinates of window: ";

cin>>xmin>>ymin>>xmax>>ymax;

int x1,y1,x2,y2;

cout<<"Enter line coordinates: ";

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

rectangle(xmin,ymin,xmax,ymax);

line(x1,y1,x2,y2);

delay(5000);

int outcode1=getcode(x1,y1);

int outcode2=getcode(x2,y2);

int acc=0;

while(1){

float m=(float)(y2-y1)/(x2-x1);

if((outcode1==0)&&(outcode2==0)){

acc=1;

break;

}

else if((outcode1 & outcode2) !=0){

break;

}

else{

int x,y;

int temp;

if(outcode1==0){

temp=outcode2;

}

else{

temp=outcode1;

}

if(temp&TOP){

x=x1+(ymax-y1)/m;

y=ymax;

}

else if(temp & BOTTOM){

x=x1+(ymin-y1)/m;

y=ymin;

}

else if(temp & LEFT){

x=xmin;

y=y1+m\*(xmin-x);

}

else if(temp & RIGHT){

x=xmax;

y=y1+m\*(xmax-x1);

}

if(temp==outcode1){

x1=x;

y1=y;

outcode1=getcode(x1,y1);

}

else{

x2=x;

y2=y;

outcode2=getcode(x2,y2);

}

}

}

setcolor(YELLOW);

cleardevice();

if(acc){

rectangle(xmin,ymin,xmax,ymax);

line(x1,y1,x2,y2);

}

else{

rectangle(xmin,ymin,xmax,ymax);

}

getch();

closegraph();

return 0;

}

//pattern

#include<iostream>

#include<graphics.h>

#include<math.h>

using namespace std;

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

{

float x,y,xi,yi,dx,dy,length;

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx>dy)

{

length=dx;

}

else{length=dy;}

xi=(x2-x1)/(float)length;

yi=(y2-y1)/(float)length;

x=x1+0.5;

y=y1+0.5;

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

{

putpixel(x,y,2);

x=x+xi;

y=y+yi;

}

delay(500);

}

void bres(int centx,int centy,int r,int c)

{

int x,y,p;

x=0;

y=r;

p=3-(2\*r);

do

{

putpixel(centx+x,centy+y,c);

putpixel(centx+x,centy-y,c);

putpixel(centx-x,centy+y,c);

putpixel(centx-x,centy-y,c);

putpixel(centx+y,centy+x,c);

putpixel(centx+y,centy-x,c);

putpixel(centx-y,centy+x,c);

putpixel(centx-y,centy-x,c);

if(p<0)

{

p=p+(4\*x)+6;

}

else{

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

y=y-1;

}

x=x+1;

}while(x<y);

delay(500);

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,NULL);

/\* int x1,y1,x2,y2,d,D,v,r;

int c=12;

cout<<"Enter x1,y1: ";

cin>>x1>>y1;

cout<<"\nENTER X2 Y2: ";

cin>>x2>>y2;

d=abs(x2-x1);

D=abs(y2-y1);

v=sqrt((d\*d)+(D\*D));

r=abs((d\*D)/(2\*v));

r=int(r);

bres((x1+x2)/2,(y1+y2)/2,r,c);

dda(x1,y1,x2,y1);

dda(x2,y1,x2,y2);

dda(x2,y2,x1,y2); //pattern1

dda(x1,y2,x1,y1);

dda((x1+x2)/2,y1,x2,(y1+y2)/2);

dda(x2,(y1+y2)/2,(x1+x2)/2,y2);

dda((x1+x2)/2,y2,x1,(y1+y2)/2);

dda(x1,(y1+y2)/2,(x1+x2)/2,y1);\*/

int centx,centy,r;

cout<<"Enter centx,centy,r: ";

cin>>centx>>centy>>r;

int c=12;

bres(centx,centy,r,c); //pattern2

bres(centx,centy,r/2,c);

dda(centx,centy-r,centx-0.865\*r,centy+r/2);

dda(centx-0.865\*r,centy+r/2,centx+0.865\*r,centy+r/2);

dda(centx+0.865\*r,centy+r/2,centx,centy-r);

delay(1000);

return 0;

}

//transformation

#include<iostream>

#include<graphics.h>

#include<cmath>

using namespace std;

int main()

{

int gd = DETECT,gm;

initgraph(&gd,&gm,NULL);

int b[3][3];

int d;

cout<<"Enter value of x1: ";

cin>>b[0][0];

cout<<"Enter value of y1: ";

cin>>b[1][0];

cout<<"Enter value of x2: ";

cin>>b[0][1];

cout<<"Enter value of y2: ";

cin>>b[1][1];

cout<<"Enter value of x3: ";

cin>>b[0][2];

cout<<"Enter value of y3: ";

cin>>b[1][2];

b[2][0] = b[2][1] = b[2][2];

setcolor(RED);

line(b[0][0],b[1][0],b[0][1],b[1][1]); //x1,y1,x2,y2

line(b[0][1],b[1][1],b[0][2],b[1][2]); //x2,y2,x3,y3

line(b[0][2],b[1][2],b[0][0],b[1][0]); //x3,y3,x1,y1

delay(5000);

cout<<"\nYour Choices Are!! \n";

cout<<"1. Translation: \n2. Scaling: \n3. Rotation: ";

cout<<"\nEnter your choice: ";

cin>>d;

switch(d)

{

case 1:

{

float a[3][3],c[3][3];

cout<<"\nEnter value value of tx and ty: ";

cin>>a[0][2]>>a[1][2];

a[0][0] = a[1][1] = a[2][2] = 1;

a[1][0] = a[2][0] = a[0][1] = a[2][1] = 0;

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

{

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

{

c[i][j] = 0;

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

{

c[i][j]+=a[i][k]\*b[k][j];

}

}

}

setcolor(BLUE);

line(c[0][0],c[1][0],c[0][1],c[1][1]);

line(c[0][1],c[1][1],c[0][2],c[1][2]);

line(c[0][2],c[1][2],c[0][0],c[1][0]);

delay(5000);

break;

}

case 2:

{

float sx,sy;

float a[3][3],c[3][3];

cout<<"\nEnter value value of sx and sy: ";

cin>>a[0][0]>>a[1][1];

a[2][2] = 1;

a[0][1] = a[0][2] = a[1][0] = a[1][2] = a[2][0] = a[2][1] = 0;

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

{

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

{

c[i][j] = 0;

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

{

c[i][j]+=a[i][k]\*b[k][j];

}

}

}

setcolor(BLUE);

line(c[0][0],c[1][0],c[0][1],c[1][1]);

line(c[0][1],c[1][1],c[0][2],c[1][2]);

line(c[0][2],c[1][2],c[0][0],c[1][0]);

delay(5000);

break;

}

case 3:

{

float z, A;

float a[3][3];

float c[3][3];

cout<<"\nEnter the angle: ";

cin>>z;

A = (z\*3.14)/180;

a[2][2] = 1;

a[0][2] = a[1][2] = a[2][0] = a[2][1] = 0;

a[0][0] = a[1][1] = cos(A);

a[0][1] = sin(A);

a[1][0] = (-sin(A));

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

{

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

{

c[i][j] = 0;

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

{

c[i][j]+=a[i][k]\*b[k][j];

}

}

}

setcolor(BLUE);

line(c[0][0],c[1][0],c[0][1],c[1][1]);

line(c[0][1],c[1][1],c[0][2],c[1][2]);

line(c[0][2],c[1][2],c[0][0],c[1][0]);

delay(5000);

break;

}

default:

cout<<"\nEnter valid choice!!";

break;

}

return 0;

}

//Hilbert

#include<iostream>

#include<graphics.h>

using namespace std;

int move(int j,int h,int &x,int&y){

if (j==1)

y-=h;

else if (j==2)

x+=h;

else if (j==3)

y+=h;

else if(j==4)

x-=h;

lineto(x,y);

}

void hilbert(int r,int d,int l,int u,int i,int h,int &x,int &y){ //rdlu

if(i>0)

{

i--;

hilbert(d,r,u,l,i,h,x,y);//interchange dr and lu

move(r,h,x,y); //varcha r d l ek ek karun

hilbert(r,d,l,u,i,h,x,y);//same rdlu

move(d,h,x,y);

hilbert(r,d,l,u,i,h,x,y);//same rdlu

move(l,h,x,y);

hilbert(u,l,d,r,i,h,x,y);//double interchange

}

}

int main(){

int gd=DETECT,gm;

initgraph(&gd,&gm,NULL);

int x,y,x0=50,y0=150,n,h=10,u=1,r=2,d=3,l=4;

x=x0;

y=y0;

cout<<"Enter no of iterations: ";

cin>>n;

moveto(x,y);

hilbert(r,d,l,u,n,h,x,y);

delay(10000);

getch();

return 0;

}

//Koch &snowflake

#include<iostream>

#include<graphics.h>

#include<math.h>

using namespace std;

float angle=60\*3.14/180;

void koch(int x1,int y1,int x2,int y2,int i)

{

int x3,y3,x4,y4,x,y;

x3=(x1\*2+x2)/3;

y3=(y1\*2+y2)/3;

x4=(x1+2\*x2)/3;

y4=(y1+2\*y2)/3;

x=x3+(x4-x3)\*cos(angle)+(y4-y3)\*sin(angle);

y=y3-(x4-x3)\*sin(angle)+(y4-y3)\*cos(angle);

if(i>1){

koch(x1,y1,x3,y3,i-1);

koch(x3,y3,x,y,i-1);

koch(x,y,x4,y4,i-1);

koch(x4,y4,x2,y2,i-1);

}

else{

line(x1,y1,x3,y3);

line(x3,y3,x,y);

line(x,y,x4,y4);

line(x4,y4,x2,y2);

}

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,NULL);

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

cout<<"Enter line coordinates: ";

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

cout<<"Enter order: ";

cin>>n;

x3=x1+(x2-x1)\*cos(angle)+(y2-y1)\*sin(angle);

y3=y1-(x2-x1)\*sin(angle)+(y2-y1)\*cos(angle);

koch(x1,y1,x3,y3,n);

koch(x3,y3,x2,y2,n);

koch(x2,y2,x1,y1,n);

delay(10000);

return 0;

}

//umbrella

#include <graphics.h>

#include <conio.h>

void drawMan(int x, int y) {

setcolor(WHITE);

circle(x, y, 10); // Head

rectangle(x - 10, y + 10, x + 10, y + 50); // Body

line(x, y + 50, x + 10, y + 70); // Right leg

line(x, y + 50, x - 10, y + 70); // Left leg

}

void drawUmbrella(int x, int y) {

setcolor(LIGHTGRAY);

line(x, y, x, y - 30); // Handle

setcolor(RED);

pieslice(x, y - 30, 0, 180, 30); // Umbrella top

}

void drawRain() {

setcolor(BLUE);

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

int x = rand() % getmaxx();

int y = rand() % getmaxy();

line(x, y, x, y + 10); // Raindrop

}

}

void drawGround() {

setcolor(GREEN);

line(0, getmaxy() - 50, getmaxx(), getmaxy() - 50); // Ground line

}

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

int x = getmaxx()/2;

int y = getmaxy() - 120; // Set the initial y position above the ground

while (!kbhit()) {

cleardevice();

drawGround();

drawMan(x, y);

drawUmbrella(x, y);

drawRain();

x += 5; // Move the man to the right (you can modify this for a more complex walking animation)

delay(100);

}

closegraph();

return 0;

}