# Bộ lệnh đồ họa devC

-intgetch();Lấy ký tự từ bàn phím-chờ ấn phím bất kỳ

-intkbhit();Kiểm tra trạng thái có phím bấm

-voiddelay(ms);Kéo trễ chương trình msmimligiây

-voidinitwindow(w, h);Mởcửasổđồhọavớichiềurộngw, chiềucaoh

-voidsetwindowtitle(s);Đặttiêuđềschocửasổđồhọa

-voidcleardevice( );Xóathiếtbịđồhọa(cửasổđồhọa)

-voidclosegraph( );Đóngcửasổđồhọa

-intgetbkcolor( );Lấymàunềnhiệnthời

-intgetcolor( );Lấymàuvẽhiệnthời

-voidsetbkcolor( c );Đặtmàunềnhiệnthờilàmàuc

-voidsetcolor( c );Đặtmàuvẽhiệnthờilàmàuc

-intgetmaxx( ); Lấy tọa độ xlớn nhất (chiều rộng cửa sổ)

-intgetmaxy();Lấy tọa độ ylớn nhất (chiều cao cửa sổ)

-intgetx();Lấy hoành độ hiện thời

-intgety();Lấy tung độ hiện thời

-voidmoveto(x, y);Di chuyển đến tọa độ (x, y)

-voidmoverel(dx, dy);Di chuyển tương đối một véctơ (dx, dy)-

voidouttext(s);Đưa xâu chữ sra màn hình-

voidouttextxy(x, y, s);Đưa xâu chữ sra màn hình tại vị trí (x, y)

-unsignedgetpixel(x , y);Lấy điểm ảnh tại vị trí (x , y)

-voidputpixel(x , y, c);Bật điểm ảnh tại vị trí (x , y) với màu c

-voidline(x1, y1, x2, y2);Kẻ đoạn thẳng với hai đầu mút cho trước

-voidlineto(x, y);Kẻ đoạn thẳng đến (x, y)

-voidlinerel(dx, dy);Kẻ tương đối đoạn thẳng (dx, dy)

-voidrectangle(x1, y1, x2, y2);Kẻ hình chữ nhật xác định bởi hai góc

-voiddrawpoly(n+1, P);Kẻ đa giác n đỉnh. Mảng P gồm 2(n+1) tọa độ của n+1 đỉnh với đỉnh cuối trùng đỉnh đầu

-voidcircle(x, y, r);Kẻ đường tròn tâm (x, y)bán kính r

-voidfloodfill(x, y, bc);Tô vùng khép kín có điểm bên trong (x, y) và màu biên bc

-voidfillpoly(n+1, P);Tô đa giác được cho như lệnh drawpoly( )

-voidsetviewport(x1, y1, x2, y2, clip);Đặt cổng quan sát hình ảnh là hình chữ nhật với tham số xén ảnh là clip.

-voidclearviewport(void);Xóa cổng quan sát hình ảnh

-voidsetwritemode(m);Đặt chế độ ghi ảnh là m. m=XOR\_PUTcó thể xóa hình ảnh bằng cách vẽ lạinó

# Kẻ đoạn thẳng

Bresenham

#include <winbgim.h>

#include<conio.h>

#define DELAY 10

int color = 2;

void Bresenham(int x1, int y1, int x2, int y2){

    int Dx = abs(x2 - x1);

    int Dy = abs(y2 - y1);

    int p = 2\*Dy - Dx;

    int c1 = 2\*Dy;

    int c2 = 2\*(Dy-Dx);

    int x = x1;

    int y = y1;

    int x\_unit = 1, y\_unit = 1;

    putpixel(x,y,color);

    while(x != x2){

        delay(DELAY);

        if (p<0) p += c1;

        else{

            p += c2;

            y += y\_unit;

        }

        x += x\_unit;

        putpixel(x, y, color);

    }

}

int main(){

    int x1,y1,x2,y2;

    int gd,gm=VGAMAX; gd=DETECT;

    initgraph(&gd,&gm,NULL);

//  setbkcolor(4);

    Bresenham(50,150, 300, 200);

    delay(9000);

    return 0;

}

Đại lượng

m>1 : ei = 2 delta X – delta Y

ekt = 2 delta X

et = 2 delta X – 2 delta Y

ei<0 : ei + 2 delta X

ei>0 : ei + 2delta X – 2 delta Y

1>m>0 : ei = 2 delta Y – delta X

ekt = 2 delta Y

et = 2 delta Y -2 dalta X

ei<0 ei + 2 delta X

ei>0 ei + 2 delta Y – 2 delta X

M < -1 delta X 🡪 - delta X

ei = 2delta X + delta Y

ekt = 2 delta X

et = 2 delta X + 2 delta Y

ei<0 ei+1 = ei + 2delta X

ei>0 ei+1 = ei +2(delta Y + delta X)

0>m>-1 delta Y 🡪 - delta Y

ei = -2delta Y – delta X

ekt = 2 delta X.

et = -2delta Y – 2 delta X

ei<0 ei+1 = ei - 2delta Y

ei>0 ei+1 = ei – 2 (delta Y + delta X)

Midpoint

#include<iostream>

#include<winbgim.h>

#include<math.h>

using namespace std;

void midpoint(int x1,int y1,int x2, int y2,int color){

    //Truong hop 0<m<1 && x1<x2 && y1<y2

    int a,b,pi,x,y,p;

    a=y2-y1;

    b=-(x2-x1);

    y=y1;

    x=x1;

    putpixel(x,y,color);    //Ve diem pixel dau tien

    p=2\*a+b;        //tinh vi tri tuong doi cua diem Midpoint so voi duong thang

    while(x < x2){

        if(p < 0){

            p+=2\*a; // ta chon chon diem yi

        }else{

            y++;

            p+=2\*(a+b);//ta chon diem yi +1

        }

        x++;

        cout<<"Pixel (x,y) midpoint = ("<<x<<", "<<y<<")\n";

        putpixel(x,y,color);

        delay(10);

    }

}

int main(){

    initwindow(800,600);

    //khoi tao window có chieu rong x=400 và chieu cao y = 500

    setwindowtitle("Thuat toan Line Midpoint");

      //thiet   lap tieu    de  cho windows

    midpoint(10,10,400,400,15);

    getch();

}

-------------------------------------------------------------------------

0<m<1

Di = deltaY -1/2 delta X

Dnoinc = delta Y

Dinc = delta Y – delta X

Di<0 : Di + delta Y

Di<0 :Di +deltaX\*deltaY.

## Midpoint vẽ đường tròn.

#include<iostream>

#include<winbgim.h>

#include<math.h>

using namespace std;

1. void put8pixel(int xc, int yc, int x, int y, int color)
2. {
3. putpixel(x + xc, y + yc, color);
4. putpixel(-x + xc, y + yc, color);
5. putpixel(x + xc, -y + yc, color);
6. putpixel(-x + xc, -y + yc, color);
7. putpixel( y + xc, x + yc, color);
8. putpixel(-y + xc, x + yc, color);
9. putpixel(y + xc, -x + yc, color);
10. putpixel(-y + xc, -x + yc, color);
11. }
12. void drawCircleMidpoint(int xc, int yc, int r, int color)
13. {
14. int x = 0; int y = r;
15. int f = 1 - r;
16. put8pixel(xc, yc, x, y, color);
17. while (x < y)
18. {
19. if (f < 0) f += (x << 1) + 3;
20. else
21. {
22. y--;
23. f += ((x - y) << 1) + 5;
24. }
25. x++;
26. put8pixel(xc, yc, x, y, color);
27. }
28. }
29. int main()
30. {
31. int gd = DETECT, gm;
32. initgraph(&gd, &gm, "c:\\tc\\bgi");
34. drawCircleMidpoint(200, 200, 100, colors::BLUE);
36. Sleep(3000);
37. closegraph();
38. return 0;
39. }

## Tô tràng

#include <conio.h>

#include <winbgim.h>

#include <iostream>

using namespace std;

struct ToaDo

{

    int x,y;

};

int MauNen;

void NhapDaGiac(int &n,int &x,int &y,ToaDo a[])

{

    cout<<"Nhap so dinh cua da giac n= "; cin>>n;

    for (int i=1;i<=n;i++)

    {

    cout<<"Toa do dinh P["<<i<<"].x= "; cin>>a[i].x;

    cout<<"Toa do dinh P["<<i<<"].y= "; cin>>a[i].y;

    }

    cout<<"Nhap diem (x,y) thuoc da giac:\n";

    cout<<"nhap x="; cin>>x;

    cout<<"nhap y="; cin>>y;

}

void VeDaGiac(int n,ToaDo a[],int color)

{

    setcolor(color);

    for (int i=1;i<=n;i++)

    {

    int j;

    if (i==n) j=1; else j=i+1;

    line(a[i].x,a[i].y,a[j].x,a[j].y);

    }

}

**void ToLoang(int x,int y,int color)**

**{**

**if (getpixel(x,y)==MauNen  && x<getmaxx() && y<getmaxy())**

**{**

**putpixel(x,y,color);**

**ToLoang(x-1,y,color);**

**ToLoang(x,y-1,color);**

**ToLoang(x+1,y,color);**

**ToLoang(x,y+1,color);**

**}**

**delay(1);**

**}**

int main()

{

    int x,y,n,Gd,Gm=VGAMAX;

    ToaDo a[10];

    NhapDaGiac(n,x,y,a);

    Gd=DETECT;

    initgraph(&Gd,&Gm,"");

    VeDaGiac(n,a,15);

    MauNen=getpixel(x,y);

    ToLoang(x,y,10);

    getch();

    closegraph();

}

Viết thủ tục tô màu Fill(x,y,bc,c) bằng thuật toán tô tràn, với x,y là tọa độ thuộc miền

tô màu. bc là màu viền còn c là màu vùng cần tô.

b. Nêu các hạn chế của thuật toán trên và cách giải quyết

giải :

void

a))

TFloodfill(int x,int y,int bc,int c){

If (getpixel(x,y)!=bc)

{

putpixel(x,y,c);

TFloodfill(x -1,y,bc,c);

TFloodfill(x+1,y,bc,c);

TFloodfill(x,y-1,bc,c);

TFloodfill(x,y+1,bc,c);

}

}

b))

Hạn chế:

+ Gọi đệ quy nhiều lần gây tràn stack

+ 1 điểm bị gọi lặp nhiều lần bởi các điểm kề nó->tô bị chậm

Cách giải quyết:

+ Hạn chế số lần gọi đệ quy (tô 3 điểm kề)

+ Không gọi đệ quy mà tô theo từng dòng

## Các phép AFFINE

Phép affine 2 chiều :

Một phép biến đổi hai chiều sẽ biến đổi điểm P trong mặt phẳng thành điểm có tọa độ

mới Q theo một quy luật nào đó. Về mặt bản chất, một phép biến đổi điểm là một ánh xạ

T được định nghĩa :

T : R2 -> R2

P(Px,Py) -> Q(Qx,Qy).

Có phương trình :

Qx = aPx+ cPy+ Trx ad-bc # 0

Qy = bPx+ dPy+ Try

Phép biến đổi Affine là phép biến đổi tọa độ điểm đặc trưng của đối tượng thành tập tương ứng các điểm mới để tạo ra các hiệu ứng cho toàn đối tượng.

– Ví dụ: phép biến đổi tọa độ với chỉ 2 điểm đầu cuối của đoạn thẳng tạo thành 2 điểm mới mà khi nối chúng với nhau tạo thành đoạn thẳng mới. Các điểm nằm trên đoạn thẳng sẽ có kết quả là điểm nằm trên đoạn thẳng mới với cùng phép biến đổi thông qua phép nội suy.

Bảo toàn tính song song

– Các đoạn thẳng song song sẽ biến đổi thành các đoạn thẳng song song

– Ví dụ: Hình bình hành sẽ biến đổi thành hình bình hành

Bảo toàn các khoảng cách tỉ lệ

– Các tỉ lệ sẽ được bảo toàn.

-Ví dụ: Trung điểm của đoạn thẳng sau khi biến đổi sẽ là trung điểm của đoạn thẳng mới.

Bảo toàn đoạn thẳng

– Các đoạn thẳng được bảo toàn, do đó ánh xạ của một đoạn thẳng vẫn là một đoạn thẳng

– Đơn giản hóa quá trình vẽ đoạn thẳng. Chúng ta chỉ cần xác định ánh xạ của hai điểm đầu cuối của đoạn thẳng và vẽ một đường thẳng nối hai điểm đó lại

– Bảo đảm sự thẳng hàng, do đó các đa giác sẽ biến đổi thành các đa giác