**直线段的扫描转换算法（3）**

**1.DDA画直线算法**

void CLine\_ScanView::DDALine(int x1, int y1, int x2, int y2, int color)

{

int dx,dy,epsl,k;

float x=10,y=10,xIncre,yIncre;

dx = x2-x1;

dy = y2-y1;

k = dy/dx;

y = y1;

if(abs(dx)>abs(dy))

epsl = abs(dx);

else

epsl = abs(dy);

xIncre = (float)dx/(float)epsl;

yIncre = (float)dy/(float)epsl;

CClientDC dc(this);

for(k=0;k<=epsl;k++){

dc.SetPixel((int) x,(int) y,color);

x += xIncre;

y += yIncre;

} }

**2.MidBresenham画直线算法**

void CLine\_ScanView::MidBresenhamLine(int x1, int y1, int x2, int y2, int color)

{

int dx,dy,d,UpIncre,DownIncre,x,y;

if(x1>x2){

x=x2;x2=x1;x1=x;

y=y2;y2=y1;y1=y;

}

x=x1;

y=y1;

dx=x2-x1;

dy=y2-y1;

d=dx-2\*dy;

CClientDC dc(this);

UpIncre=2\*dx-2\*dy;

DownIncre=-2\*dy;

while(x<=x2){

dc.SetPixel(x,y,color);

x++;

if(d<0){

y++;

d += UpIncre;

}

else

d += DownIncre;

} }

**3.Bresenham画直线算法**

void CLine\_ScanView::BresenhamLine(int x1, int y1, int x2, int y2, int color)

{

int x,y,dx,dy,e;

dx = x2-x1;

dy = y2-y1;

e = -dx;

x = x1;

y = y1;

CClientDC dc(this);

while(x<=x2){

dc.SetPixel(x,y,color);

x++;

e = e+2\*dy;

if(e>0){

y++;

e=e-2\*dx;

}

} }

**圆的扫描转换算法（2/椭圆1）**

**1. 中点画圆算法**

void CLine\_ScanView::MidBresenhamCircle(int r, int color)

{

int x = 0;

int y = r;

int d = 1-r;

int x0 = 100;

int y0 = 100;

CDC \*pDC = GetDC();

while(x<=y){

pDC->SetPixel(x+x0,y+y0,color);

pDC->SetPixel(-x+x0,y+y0,color);

pDC->SetPixel(-x+x0,-y+y0,color);

pDC->SetPixel(x+x0,-y+y0,color);

pDC->SetPixel(y+x0,x+y0,color);

pDC->SetPixel(-y+x0,x+y0,color);

pDC->SetPixel(-y+x0,-x+y0,color);

pDC->SetPixel(y+x0,-x+y0,color);

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

else{ d+=2\*(x-y)+5; y--; }

x++;

} }

**2.Breshnham画圆算法**

circle (xc, yc, radius, c)

int xc, yc, radius, c;

{

int x, y, p;

x=0;

y=radius;

p=3-2\*radius;

while (x<y)

{

  plot\_circle\_points(xc, yc, x, y, c);

  if (p<0) p=p+4\*x+6;

  else{ p=p+4\*(x-y)+10; y-=1; }

  x+=1;

}

if (x= =y)

plot\_circle\_points(xc, yc, x, y, c);

}

plot\_circle\_points(xc, yc, x, y, c)

int xc, yc, x, y, c;

{

set\_pixel(xc+x, yc+y, c);

set\_pixel(xc+x, yc+y, c);

set\_pixel(xc+x, yc-y, c);

set\_pixel(xc-x, yc-y, c);

set\_pixel(xc+y, yc+x, c);

set\_pixel(xc-y, yc+x, c);

set\_pixel(xc+y, yc-x, c);

set\_pixel(xc-y, yc-x, c);

}

**3.中点画椭圆算法**

MidpointEllipe(a,b, color)

int a,b,color;

{ int x,y; float d1,d2;

x = 0; y = b;

d1 = b\*b +a\*a\*(-b+0.25);

ellipesputpixel(x,y,color);

while( b\*b\*(x+1) < a\*a\*(y-0.5)) 生成上半部分椭圆

{ { if (d1<0)

d1 +=b\*b\*(2\*x+3); x++; }

else { d1 +=(b\*b\*(2\*x+3)+a\*a\*(-2\*y+2))

x++; y--; }

ellipesputpixel(x,y,color);

}//上部分

d2 = b\*b\*(x+0.5)\*(x+0.5)+a\*a\*(y-1)\*(y-1)-a\*a\*b\*b;

while(y >0) 生成下半部分椭圆

{ if (d2 <0) { d2 +=b\*b\*(2\*x+2)+a\*a\*(-2\*y+3);

x++; y--；}

else {d2 += a\*a\*(-2\*y+3); y--; }

ellipesputpixel(x,y,color); }

Void ellipespixel(int x,int y,int color)

{

putpixel(x,y,color)

putpixel(-x,y,color)

putpixel(x,-y,color)

putpixel(-x,-y,color)

}

**填充算法（3）**

**1.种子填充算法——递归算法可实现如下：**

void FloodFill4(int x,int y,int oldColor,int newColor)

{ if(GetPixel(x,y) == oldColor)

{ PutPixel(x,y,newColor);

FloodFill4(x,y+1,oldColor,newColor);

FloodFill4(x,y-1,oldColor,newColor);

FloodFill4(x-1,y,oldColor,newColor);

FloodFill4(x+1,y,oldColor,newColor);

}

}

**2.多边形的4种子填充算法——边界表示的4连通区域**

void CLine\_ScanView::BoundaryFill4(int x, int y, int boundarycolor, int newcolor)

{

CDC\* pDC = GetDC();

CPen pen;

pen.CreatePen(PS\_SOLID,2,RGB(0,255,0));

pDC->SelectObject(&pen);

int a[4][2] = {200,100,180,120,220,120,200,100};

pDC->MoveTo(a[0][0],a[0][1]);

for(int i=0;i<4;i++) {

pDC->LineTo(a[i][0],a[i][1]);

}

int color = pDC->GetPixel(x,y);

if(color!=newcolor&&color!=boundarycolor){

pDC->SetPixel(x,y,newcolor);

BoundaryFill4(x,y+1,boundarycolor,newcolor);

BoundaryFill4(x,y-1,boundarycolor,newcolor);

BoundaryFill4(x+1,y,boundarycolor,newcolor);

BoundaryFill4(x-1,y,boundarycolor,newcolor);

}

}

**3.多边形的8种子填充算法——边界表示的4连通区域**

void CLine\_ScanView::BoundaryFill8(int x, int y, int boundarycolor, int newcolor)

{

CDC\* pDC = GetDC();

CPen pen;

pen.CreatePen(PS\_SOLID,2,RGB(0,255,0));

pDC->SelectObject(&pen);

int a[4][2] = {200,100,180,120,220,120,200,100};

pDC->MoveTo(a[0][0],a[0][1]);

for(int i=0;i<4;i++) {

pDC->LineTo(a[i][0],a[i][1]);

}

int color = pDC->GetPixel(x,y);

if(color!=newcolor&&color!=boundarycolor){

pDC->SetPixel(x,y,newcolor);

BoundaryFill8(x,y+1,boundarycolor,newcolor);

BoundaryFill8(x+1,y+1,boundarycolor,newcolor);

BoundaryFill8(x,y-1,boundarycolor,newcolor);

BoundaryFill8(x-1,y-1,boundarycolor,newcolor);

BoundaryFill8(x+1,y,boundarycolor,newcolor);

BoundaryFill8(x+1,y-1,boundarycolor,newcolor);

BoundaryFill8(x-1,y,boundarycolor,newcolor);

BoundaryFill8(x-1,y+1,boundarycolor,newcolor);

}

}

**二维变换（5）**

**1.二维变换的比例变换响应函数**

void CLine\_ScanView::OnTwoProportion()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1;

pen1.CreatePen(PS\_SOLID,2,RGB(255,10,255));

int a[11][2]={100,105,90,135,60,135,85,155,75,185,100,165,125,185,115,155,140,135,110,135,100,105};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0],a[i][1]);

}

double sx=0.5,sy=0.5;

dc.MoveTo (a[0][0]\*sx,a[0][1]\*sy);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0]\*sx,a[i][1]\*sy);

Sleep(5);

}

pen1.DeleteObject();

}

**2.二维变换的旋转变换响应函数**

void CLine\_ScanView::OnTwoRotating()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1;

pen1.CreatePen(PS\_SOLID,2,RGB(255,10,255));

int a[11][2]={100,105,90,135,60,135,85,155,75,185,100,165,125,185,115,155,140,135,110,135,100,105};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0],a[i][1]);

}

int t = 600;

dc.MoveTo (a[0][0]\*cos((t/6)\*1.0)-a[0][1]\*sin((t/6)\*1.0),//一个点

a[0][0]\*sin((t/6)\*1.0)+a[0][1]\*cos((t/6)\*1.0));

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0]\*cos((t/6)\*1.0)-a[i][1]\*sin((t/6)\*1.0),//一个点

a[i][0]\*sin((t/6)\*1.0)+a[i][1]\*cos((t/6)\*1.0));

}

pen1.DeleteObject();

}

**3.二维变换的错切变换响应函数**

void CLine\_ScanView::OnTwoShear()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1;

pen1.CreatePen(PS\_SOLID,2,RGB(255,10,255));

int a[11][2]={100,105,90,135,60,135,85,155,75,185,100,165,125,185,115,155,140,135,110,135,100,105};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0],a[i][1]);

}

double c=2,d=1.5,u=50;

dc.MoveTo (a[0][0]+c\*a[0][1],a[0][1]-u);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0]+c\*a[i][1],a[i][1]-u);

}

int s=170,h=200;

dc.MoveTo (a[0][0]+s,d\*a[0][0]+a[0][1]-h);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0]+s,d\*a[i][0]+a[i][1]-h);

}

pen1.DeleteObject();

}

**4.二维变换的对称变换响应函数**

void CLine\_ScanView::OnTwoSymmetry()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1;

pen1.CreatePen(PS\_SOLID,2,RGB(255,10,255));

int a[11][2]={100,105,90,135,60,135,85,155,75,185,100,165,125,185,115,155,140,135,110,135,100,105};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0],a[i][1]);

}

int m=0,b=200;

dc.MoveTo (a[0][0]\*(1-m\*m)/(1+m\*m)+2\*(a[0][1]-b)\*m/(1+m\*m),//一个点

a[0][0]\*(2\*m)/(1+m\*m)+(a[0][1]-b)\*(m\*m-1)/(1+m\*m)+b);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0]\*(1-m\*m)/(1+m\*m)+2\*(a[i][1]-b)\*m/(1+m\*m), a[i][0]\*(2\*m)/(1+m\*m)+(a[i][1]-b)\*(m\*m-1)/(1+m\*m)+b);

}

pen1.DeleteObject();

}

**5.二维变换的平移变换响应函数**

void CLine\_ScanView::OnTwoTranslation()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1;

pen1.CreatePen(PS\_SOLID,2,RGB(255,10,255));

int a[11][2]={100,105,90,135,60,135,85,155,75,185,100,165,125,185,115,155,140,135,110,135,100,105};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0],a[i][1]);

}

int tx=120,ty=100;

dc.MoveTo (a[0][0]+tx,a[0][1]+ty);

for(int i=0;i<11;i++) {

dc.LineTo (a[i][0]+tx,a[i][1]+ty);

}

pen1.DeleteObject();

}

**三维变换（5）**

**1.三维变换的比例变换响应函数**

void CLine\_ScanView::OnThreeProportion()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(0,255,0));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(a[i][0],a[i][1]);

}

int b[4][4] = {2,0,0,0,0,2,0,0,0,0,2,0,0,0,0,1};

int c[4][4];

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

c[i][j] = 0;

}

}

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

for(int k=0;k<4;k++){

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

}

}

}

dc.SelectObject(&pen2);

dc.MoveTo(c[0][0],c[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(c[i][0],c[i][1]);

}

pen1.DeleteObject();

pen2.DeleteObject();

}

**2.三维变换的旋转变换响应函数**

void CLine\_ScanView::OnThreeRotating()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(255,0,0));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++) {

dc.LineTo (a[i][0],a[i][1]);

}

int t = 600;

dc.SelectObject(&pen2);

dc.MoveTo (a[0][0]\*cos((t/6)\*1.0)-a[0][1]\*sin((t/6)\*1.0),//一个点

a[0][0]\*sin((t/6)\*1.0)+a[0][1]\*cos((t/6)\*1.0));

for(int i=0;i<4;i++) {

dc.LineTo (a[i][0]\*cos((t/6)\*1.0)-a[i][1]\*sin((t/6)\*1.0),//一个点

a[i][0]\*sin((t/6)\*1.0)+a[i][1]\*cos((t/6)\*1.0));

}

pen1.DeleteObject();

pen2.DeleteObject();

}

**3.三维变换的错切变换响应函数**

void CLine\_ScanView::OnThreeShear()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(255,255,0));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(a[i][0],a[i][1]);

}

int b[4][4] = {1,2,0,0,1.5,1,0,0,0,0,1,0,0,0,0,1};

int c[4][4];

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

c[i][j] = 0;

}

}

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

for(int k=0;k<4;k++){

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

}

}

}

dc.SelectObject(&pen2);

dc.MoveTo(c[0][0],c[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(c[i][0],c[i][1]);

}

pen1.DeleteObject();

pen2.DeleteObject();

}

**4.三维变换的对称变换响应函数**

void CLine\_ScanView::OnThreeSymmetry()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(0,255,255));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(a[i][0],a[i][1]);

}

int b[4][4] = {0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1}; //沿y=x对称

int c[4][4];

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

c[i][j] = 0;

}

}

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

for(int k=0;k<4;k++){

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

}

}

}

dc.SelectObject(&pen2);

dc.MoveTo(c[0][0],c[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(c[i][0],c[i][1]);

}

pen1.DeleteObject();

pen2.DeleteObject();

}

**5.三维变换的平移变换响应函数**

void CLine\_ScanView::OnThreeTranslation()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1,pen2;

pen1.CreatePen(PS\_SOLID,2,RGB(0,0,0));

pen2.CreatePen(PS\_SOLID,2,RGB(0,0,255));

int a[4][4] = {100,150,0,1,50,200,0,1,150,200,0,1,100,150,0,1};

dc.SelectObject(&pen1);

dc.MoveTo (a[0][0],a[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(a[i][0],a[i][1]);

}

int b[4][4] = {1,0,0,0,0,1,0,0,0,0,1,0,100,100,0,1};

int c[4][4];

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

c[i][j] = 0;

}

}

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

for(int k=0;k<4;k++){

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

}

}

}

dc.SelectObject(&pen2);

dc.MoveTo(c[0][0],c[0][1]);

for(int i=0;i<4;i++){

dc.LineTo(c[i][0],c[i][1]);

}

pen1.DeleteObject();

pen2.DeleteObject();

}