计算机图形学算法代码

学期：2021-2022-1

班级：计算机19-6

学号：2019214860

姓名：苗永正

山东农业大学信息学院

## 1 、DDA直线算法

算法代码：

void CzuoyeView::OnLbtDdl()

{

// TODO: 在此添加命令处理程序代码

int x0=100, y0=100, x1=500, y1=800, color=177;

int dx, dy, epsl, k;

CClientDC d(this);

float x, y, xIncre, yIncre;

dx = x1 - x0; dy = y1 - y0;

x = x0; y = y0;

if (abs(dx) < abs(dy)) epsl = abs(dx);

else epsl = abs(dy);

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

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

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

{

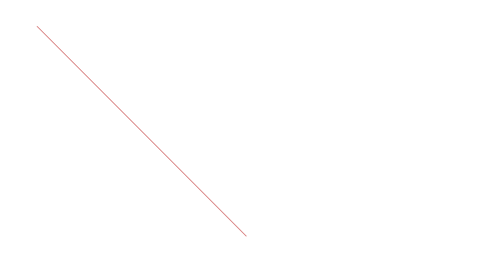
d.SetPixel(int(x + 0.5), (int)(y + 0.5), color);

x += xIncre;

y += yIncre;

}

## }运行截图



## 2、直线中点法

算法代码：

void CzuoyeView::OnLbtZhongdian()

{

// TODO: 在此添加命令处理程序代码

int x0 = 100; int y0 = 100; int x1 = 500; int y1 = 800; int color = 177;

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

CClientDC dc(this);

if (x0 > x1)

{

x = x1; x1 = x0; x0 = x;

y = y1; y1 = y0; y0 = y;

}

x = x0; y = x0;

dx = x1 - x0; dy = y1 - y0; d = dx - 2 \* dy;

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

while (x <= x1)

{

dc.SetPixel(x, y, color);

x++;

if (d < 0)

{

y++;

d += UpIncre;

}

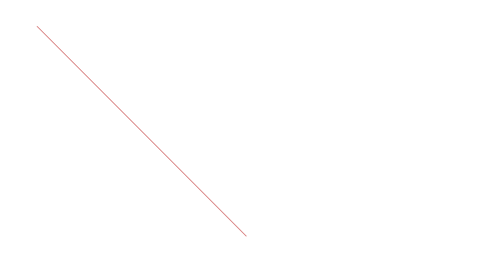
else

d += DownIncre;

}

}

## 运行截图



## 3、直线Bresenham法

算法代码：

void CzuoyeView::OnLbtBresenham()

{

// TODO: 在此添加命令处理程序代码

int x0 = 0 ;int y=100; int x1 = 0; int y1 = 800; int color = 177;

int x, y, dx, dy, e;

CClientDC dc(this);

dx = x1 - x0;

dy = y1 - y0;

e = -dx; x = x0; y = y0;

while (x <= x1)

{

dc.SetPixel(x, y, color);

x++;

e = e + 2 \* dy;

if (e > 0)

{

y++;

e = e - 2 \* dx;

}

}

## 运行截图



## 4、圆Bresenham法

算法代码：

void CzuoyeView::OnLbtYuanbre()

{

// TODO: 在此添加命令处理程序代码

int r = 200, color = 177;

int x = 0;

int y = r;

int d = 1 - r;

int x0 = 300;

int y0 = 400;

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--;

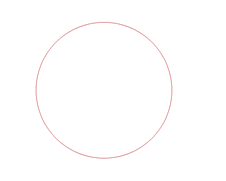
}

x++;

}

}

## 运行截图



## 5、椭圆Bresenham法

算法代码：

void CzuoyeView::OnLbtTuoyuan()

{

// TODO: 在此添加命令处理程序代码

CRect rect;

GetClientRect(&rect);

CDC\* pDC = GetDC();

//画坐标轴，必须在GetClientRect(&rect)一句后

pDC->MoveTo(0, rect.Height() / 2);

pDC->LineTo(rect.Width(), rect.Height() / 2);

pDC->MoveTo(rect.Width() / 2, 0);

pDC->LineTo(rect.Width() / 2, rect.Height());

//设置坐标原点在客户区

pDC->SetMapMode(MM\_ISOTROPIC);

pDC->SetViewportExt(rect.right, rect.bottom);

pDC->SetViewportOrg(rect.right / 2, rect.bottom / 2);

pDC->SetWindowOrg(0, 0);

pDC->SetWindowExt(1000, -1000);

int a = 100, b = 200, color = 255, x, y;//长半轴为100，短半轴为200,颜色为红色

float d1;

float d2 = 0;

x = 0; y = b;

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

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

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

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

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

while (b \* b \* (x + 1) < a \* a \* (y - 0.5))

{

if (d1 <= 0)

{

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

x++;

}

else

{

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

x++;

y--;

}

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

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

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

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

}

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

while (y > 0)

{

if (d2 <= 0)

{

d2 += b \* b \* (2 \* x + 2) + a \* a \* (-2 \* y + 3);

x++;

y--;

}

else

{

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

y--;

}

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

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

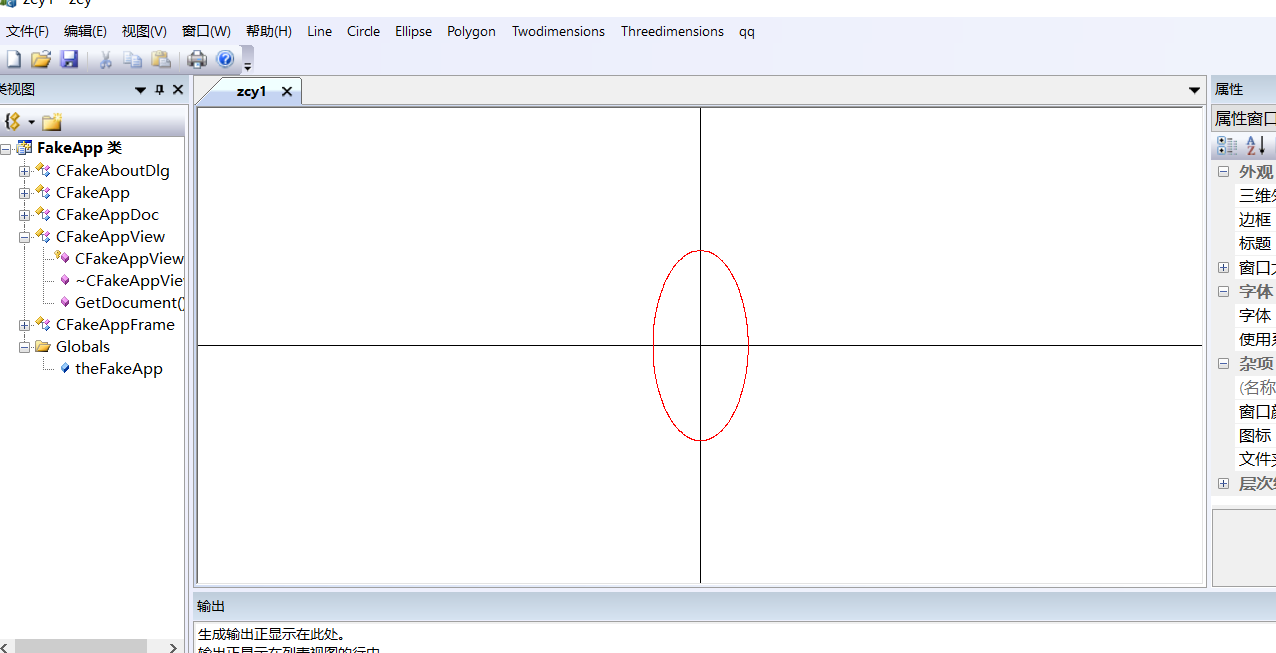
pDC->SetPixel(-x, y, color);

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

}

}

## 运行截图



## 6 、X-扫描算法

算法代码：

void CzuoyeView::OnLbtXsaomiao()

{

// TODO: 在此添加命令处理程序代码

CDC\* pDC = GetDC();

const int POINTNUM = 6;

typedef struct XET

{

float x;

float dx, ymax;

XET\* next;

}

AET, NET;

struct point

{

float x;

float y;

}

polypoint[POINTNUM] = { 245,47,578,77,457,126,878,123,545,789,445,231 };

int MaxY = 0;

int i;

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

if (polypoint[i].y > MaxY)

MaxY = polypoint[i].y;

AET\* pAET = new AET;

pAET->next = NULL;

NET\* pNET[1024];

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

{

pNET[i] = new NET;

pNET[i]->next = NULL;

}

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

{

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

if (polypoint[j].y == i)

{

if (polypoint[(j - 1 + POINTNUM) % POINTNUM].y > polypoint[j].y)

{

NET\* p = new NET;

p->x = polypoint[j].x;

p->ymax = polypoint[(j - 1 + POINTNUM) % POINTNUM].y;

p->dx = (polypoint[(j - 1 + POINTNUM) % POINTNUM].x - polypoint[j].x) / (polypoint[(j - 1 + POINTNUM) % POINTNUM].y - polypoint[j].y);

p->next = pNET[i]->next;

pNET[i]->next = p;

}

if (polypoint[(j + 1 + POINTNUM) % POINTNUM].y > polypoint[j].y)

{

NET\* p = new NET;

p->x = polypoint[j].x;

p->ymax = polypoint[(j + 1 + POINTNUM) % POINTNUM].y;

p->dx = (polypoint[(j + 1 + POINTNUM) % POINTNUM].x - polypoint[j].x) / (polypoint[(j + 1 + POINTNUM) % POINTNUM].y - polypoint[j].y);

p->next = pNET[i]->next;

pNET[i]->next = p;

}

}

}

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

{

NET\* p = pAET->next;

while (p)

{

p->x = p->x + p->dx;

p = p->next;

}

AET\* tq = pAET;

p = pAET->next;

tq->next = NULL;

while (p)

{

while (tq->next && p->x >= tq->next->x)

tq = tq->next;

NET\* s = p->next;

p->next = tq->next;

tq->next = p;

p = s;

tq = pAET;

}

AET\* q = pAET;

p = q->next;

while (p)

{

if (p->ymax == i)

{

q->next = p->next;

delete p;

p = q->next;

}

else

{

q = q->next;

p = q->next;

}

}

p = pNET[i]->next;

q = pAET;

while (p)

{

while (q->next && p->x >= q->next->x)

q = q->next;

NET\* s = p->next;

p->next = q->next;

q->next = p;

p = s;

q = pAET;

}

p = pAET->next;

while (p && p->next)

{

for (float j = p->x; j <= p->next->x; j++)

pDC->SetPixel(static\_cast<int>(j), i, RGB(152, 0, 0));

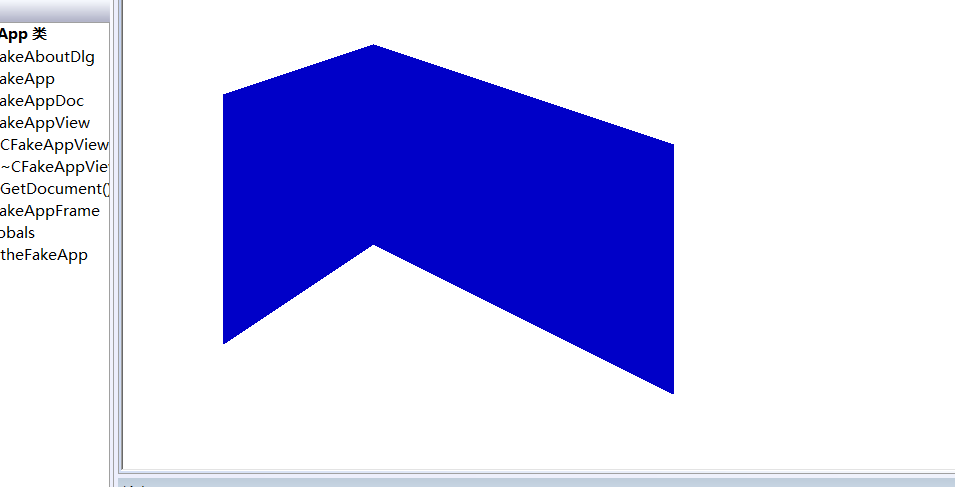
p = p->next->next;

}

}

}

## 运行截图



## 7、四邻域种子法

算法代码：

void CzuoyeView::OnLbtZzsi()

{

// TODO: 在此添加命令处理程序代码

CDC\* pDC = GetDC();

mpDC = pDC;

POINT ptpolygon1[6] = { {600,250},{600,80},{615,50},{615,230},{700,230},{700,250} };

mpDC->Polygon(ptpolygon1, 6);

BoundaryFill(605, 100, RGB(0, 0, 0), RGB(0, 0, 188));

}

struct Seed

{

int x; int y;

};

void CzuoyeView::BoundaryFill(int x, int y, COLORREF boundarycolor, COLORREF newcolor)

{

if (mpDC->GetPixel(x, y) != boundarycolor && mpDC->GetPixel(x, y) != newcolor)

{

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

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

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

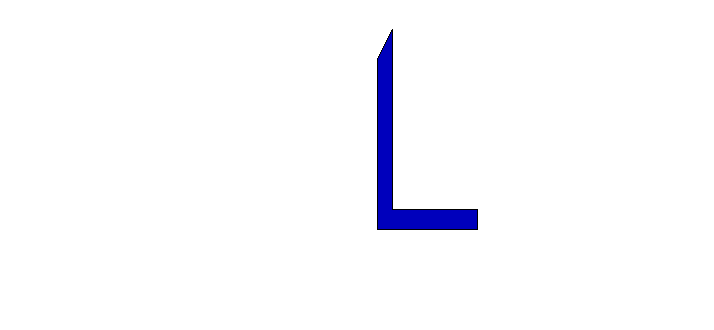
BoundaryFill(x - 1, y, boundarycolor, newcolor);

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

}

}

## 运行截图



## 8、八邻域种子法

算法代码：

void CzuoyeView::OnLbtZzba()

{

// TODO: 在此添加命令处理程序代码

CDC\* pDC = GetDC();

mpDC = pDC;

POINT ptpolygon1[6] = { {600,50},{600,250},{700,250},{700,230},{620,230},{620,50} };

mpDC->Polygon(ptpolygon1, 6);

BoundaryFill2(610, 100, RGB(0, 0, 0), RGB(0, 0, 188));

}

struct Seed1

{

int x; int y;

};

void CzuoyeView::BoundaryFill2(int x, int y, COLORREF boundarycolor, COLORREF newcolor)

{

if (mpDC->GetPixel(x, y) != boundarycolor && mpDC->GetPixel(x, y) != newcolor)

{

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

BoundaryFill2(x, y + 1, boundarycolor, newcolor);

BoundaryFill2(x, y - 1, boundarycolor, newcolor);

BoundaryFill2(x - 1, y, boundarycolor, newcolor);

BoundaryFill2(x + 1, y, boundarycolor, newcolor);

BoundaryFill2(x+1, y + 1, boundarycolor, newcolor);

BoundaryFill2(x-1, y - 1, boundarycolor, newcolor);

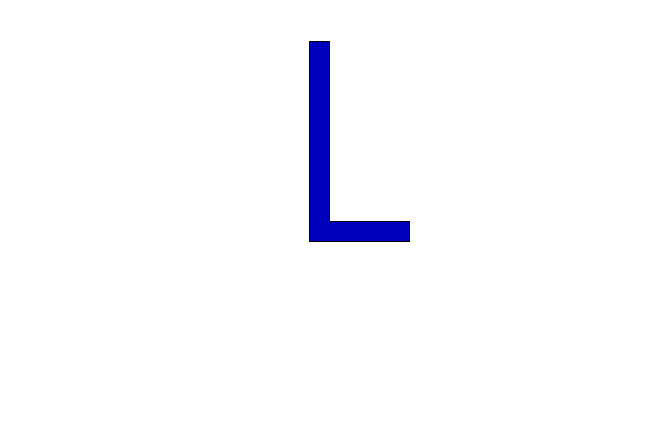
BoundaryFill2(x - 1, y+1, boundarycolor, newcolor);

BoundaryFill2(x + 1, y-1, boundarycolor, newcolor);

}

}

## 运行截图



## 9、二维几何变换——平移

算法代码：

void CzuoyeView::OnBianhuanErpingyi()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CRect rect;

GetClientRect(&rect);

dc.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);

int Tx = 100, Ty = 200;

int a[4][2] = { 0, 200, 100, 200, 50, 100, 0, 200 };

dc.TextOut(20, 80, (CString)"原图形");

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

{

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

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

}

CPen pen;

pen.CreatePen(PS\_SOLID, 1, RGB(0, 0, 100));

dc.SelectObject(&pen);

dc.TextOut(20 + Tx, 80+Ty, (CString)"平移后图形");

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

{

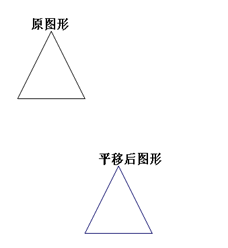
dc.MoveTo(a[i][0] + Tx, a[i][1] + Ty);

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

}

}

## 运行截图



## 10、二维几何变换——旋转

算法代码：

void CzuoyeView::OnBianhuanErxuanzhuan()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CRect rect;

GetClientRect(&rect);

dc.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);

float pi = 3.14 / 2;

float c = cos(pi);

float s = sin(pi);

int Tx = 150;

int a[4][2] = { 0, 200, 100, 200, 50, 100, 0, 200 };

dc.TextOut(100, 0, (CString)"原图形");

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

{

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

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

}

CPen pen;

pen.CreatePen(PS\_SOLID, 1, RGB(0, 0, 100));

dc.SelectObject(&pen);

dc.TextOut(100 \* c+Tx, 100, (CString)"旋转后图形");

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

{

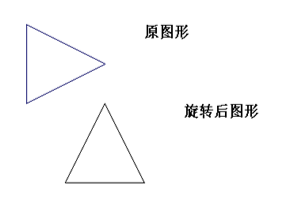
dc.MoveTo(a[i][0] \* c - a[i][1] \* s + Tx, a[i][0] \* s + a[i][0] \* c);

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

}

}

## 运行截图



## 11、二维几何变换——错切

算法代码：

void CzuoyeView::OnBianhuanErcuoqie()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CRect rect;

GetClientRect(&rect);

dc.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);

int b = 1, c = 2;

int Tx = 200;

int a[4][2] = { 0, 200, 100, 200, 50, 100, 0, 200 };

dc.TextOut(100, 100, (CString)"原图形");

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

{

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

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

}

CPen pen;

pen.CreatePen(PS\_SOLID, 1, RGB(0, 0, 100));

dc.SelectObject(&pen);

dc.TextOut(150 + Tx, 200, (CString)"错切后图形");

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

{

dc.MoveTo(a[i][0] + c \* a[i][1] + Tx, b \* a[i][0] + a[i][1]);

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

}

}

## 运行截图



## 12、二维几何变换——比例

算法代码：

void CzuoyeView::OnBianhuanErbili()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CRect rect;

GetClientRect(&rect);

dc.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);

int Tx = 300;

int Sx = 2, Sy = 2;

int a[4][2] = { 0, 150, 100, 100, 50, 100, 0, 150 };

dc.TextOut(100, 100, (CString)"原图形");

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

{

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

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

}

CPen pen;

pen.CreatePen(PS\_SOLID, 1, RGB(0, 0, 200));

dc.SelectObject(&pen);

dc.TextOut(100 + Tx, 150, (CString)"比例变换后图形");

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

{

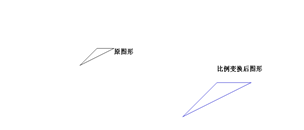
dc.MoveTo(a[i][0] \* Sx + Tx, a[i][1] \* Sy);

dc.LineTo(a[i + 1][0] \* Sx + Tx, a[i + 1][1] \* Sy);

}

}

## 运行截图



## 13、二维几何变换——对称

算法代码：

void CzuoyeView::OnBianhuanErduichen()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CRect rect;

GetClientRect(&rect);

dc.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);

int a[4][2] = { 0, 0, 100, 0, 100, 100, 0, 0 };

dc.TextOut(100, 0, (CString)"原图形");

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

{

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

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

}

CPen pen;

pen.CreatePen(PS\_SOLID, 1, RGB(0, 0, 200));

dc.SelectObject(&pen);

dc.TextOut(-100, 0, (CString)"对称后图形");

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

{

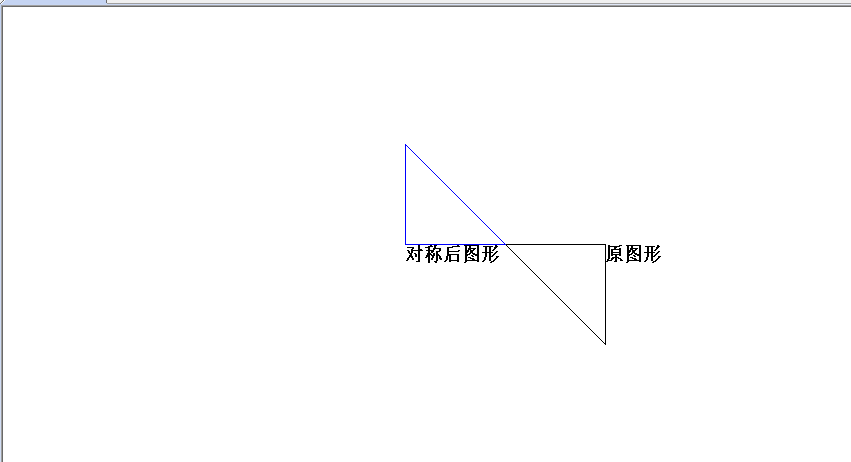
dc.MoveTo(-a[i][0], -a[i][1]);

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

}

}

## 运行截图



## 14 直线cohen-sutherland法

算法代码：

void CzuoyeView::OnSanCohen()

{

// TODO: 在此添加命令处理程序代码

CDC\* pDC = GetDC();

CzuoyeDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

wxl = 200, wxr = 700, wyb = 200, wyt = 600;

//CPoint X(200,200);//左下

// CPoint Y(700,600);//右上

pDC->Rectangle(wxl, wyb, wxr, wyt);

CPoint A(500, 400), B(800, 420);//全部保留

CohenSutherland(A, B, pDC);

CPoint C(1000, 500), D(900, 500);//完全去除

CohenSutherland(C, D, pDC);

CPoint E(350, 300), F(1200, 300);//直线需要剪切

CohenSutherland(E, F, pDC);

// pDC->Polyline()

}

void CzuoyeView::CohenSutherland(CPoint p1, CPoint p2, CDC\* pdc)

{

int code1, code2, code;

CPoint p;

code1 = EnCode(p1);

code2 = EnCode(p2);

while (code1 != 0 || code2 != 0)

{

if (code1 && code2 != 0)//全部位于外部

return;

if (code1 != 0)code = code1;//部分在外部

else

code = code2;

if ((D0 & code) != 0)//Left

{

p.x = wxl;

p.y = p1.y + (p.x - p1.x) \* (p2.y - p1.y) / (p2.x - p1.x);

}

else if ((D1 & code) != 0)//Right

{

p.x = wxr;

p.y = p1.y + (p.x - p1.x) \* (p2.y - p1.y) / (p2.x - p1.x);

}

else if ((D2 & code) != 0)//Bottom

{

p.y = wyb;

p.x = p1.x + (p.y - p1.y) \* (p2.x - p1.x) / (p2.y - p1.y);

}

else if ((D3 & code) != 0)//Top

{

p.y = wyt;

p.x = p1.x + (p.y - p1.y) \* (p2.x - p1.x) / (p2.y - p1.y);

}

if (code == code1)//p1在外部

{

p1.x = p.x; p1.y = p.y;

code1 = EnCode(p);

}

else

{

p2.x = p.x; p2.y = p.y;

code2 = EnCode(p);

}

}

CPen pen;

pen.CreatePen(PS\_SOLID, 2, RGB(0, 123, 100));

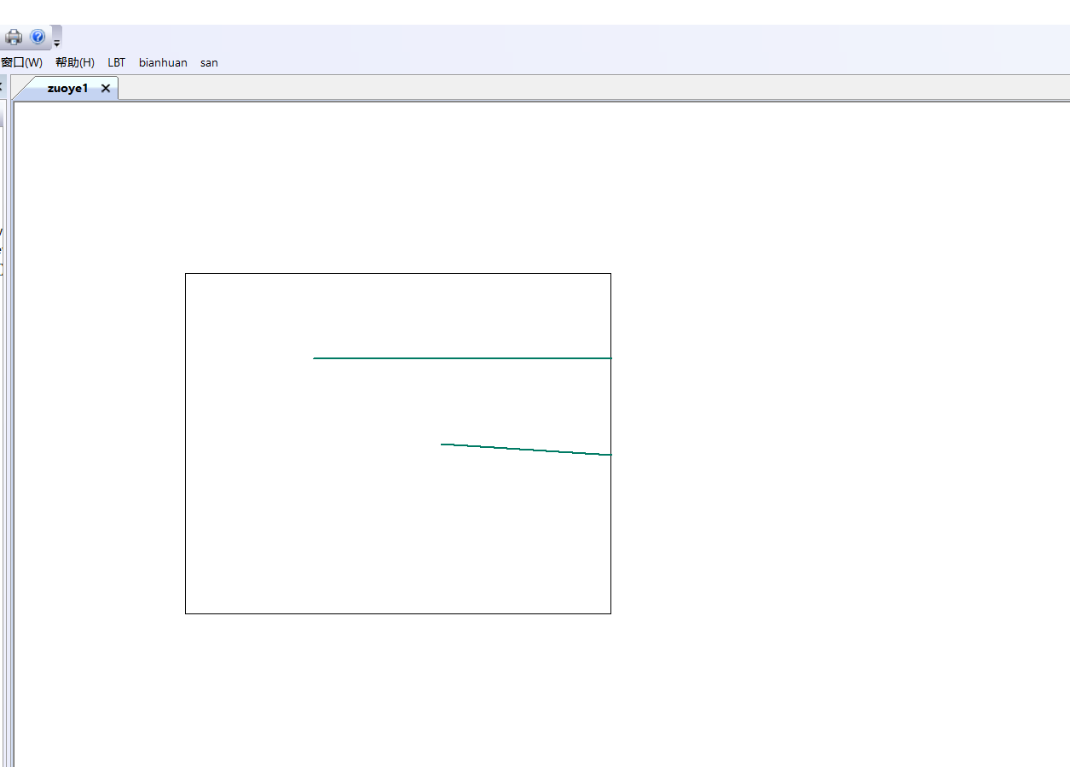
pdc->SelectObject(&pen);

pdc->MoveTo(p1);

pdc->LineTo(p2);

}

运行抓图：



## 15 多边形weiler-atherton法

算法代码：

void CzuoyeView::OnSanWeiler()

{

// TODO: 在此添加命令处理程序代码

int x0, y0, x1, y1;

CClientDC dc(this);

int min\_clip\_x = 100, min\_clip\_y = 100, max\_clip\_x = 300, max\_clip\_y = 300;

dc.MoveTo(min\_clip\_x, min\_clip\_y);

dc.LineTo(min\_clip\_x, max\_clip\_y);

dc.MoveTo(min\_clip\_x, max\_clip\_y);

dc.LineTo(max\_clip\_x, max\_clip\_y);

dc.MoveTo(max\_clip\_x, max\_clip\_y);

dc.LineTo(max\_clip\_x, min\_clip\_y);

dc.MoveTo(max\_clip\_x, min\_clip\_y);

dc.LineTo(min\_clip\_x, min\_clip\_y);

x0 = 120; y0 = 90; x1 = 310, y1 = 310;

cut\_line(x0, y0, x1, y1);

}

void CzuoyeView::cut\_line(int x0, int x1, int y0, int y1)

{

#define CLIP\_CODE\_C 0x0000

#define CLIP\_CODE\_N 0x0008

#define CLIP\_CODE\_S 0x0004

#define CLIP\_CODE\_E 0x0002

#define CLIP\_CODE\_W 0x0001

#define CLIP\_CODE\_NE 0x000a

#define CLIP\_CODE\_SE 0x0006

#define CLIP\_CODE\_NW 0x0009

#define CLIP\_CODE\_SW 0x0005

int min\_clip\_x = 100, min\_clip\_y = 100, max\_clip\_x = 300, max\_clip\_y = 300;

int p0\_code = 0, p1\_code = 0;

//确定各个顶点所在的位置代码

if (y0 < min\_clip\_y)

p0\_code |= CLIP\_CODE\_N;

else if (y0 > max\_clip\_y)

p0\_code |= CLIP\_CODE\_S;

if (x0 < min\_clip\_x)

p0\_code |= CLIP\_CODE\_W;

else if (x0 > max\_clip\_x)

p0\_code |= CLIP\_CODE\_E;

if (y1 < min\_clip\_y)

p1\_code |= CLIP\_CODE\_N;

else if (y1 > max\_clip\_y)

p1\_code |= CLIP\_CODE\_S;

if (x1 < min\_clip\_x)

p1\_code |= CLIP\_CODE\_W;

else if (x1 > max\_clip\_x)

p1\_code |= CLIP\_CODE\_E;

//先检测一些简单的情况

if (p0\_code & p1\_code) //有相同的位置代码，表示在裁剪区外部

{

return;

}

else if (p0\_code == 0 && p1\_code == 0) //表示两个点都在裁剪区内，不需要裁剪

{

CClientDC dc(this);

dc.MoveTo(x0, y0);

dc.LineTo(x1, y1);

return;

}

else

{

int x2, y2;

x2 = (x0 + x1) / 2;

y2 = (y0 + y1) / 2;

if (x0 - x2 > 1 || x2 - x0 > 1)

{

cut\_line(x0, y0, x2, y2);

}

if (x1 - x2 > 1 || x2 - x1 > 1)

{

cut\_line(x2, y2, x1, y1);

}

}

return;

}

运行抓图：



## 16、三维视图平移

算法代码：

void CzuoyeView::OnSanPingyi()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1, pen2;

pen1.CreatePen(PS\_SOLID, 5, RGB(0, 0, 100));

pen2.CreatePen(PS\_SOLID, 5, RGB(0, 0, 200));

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]);

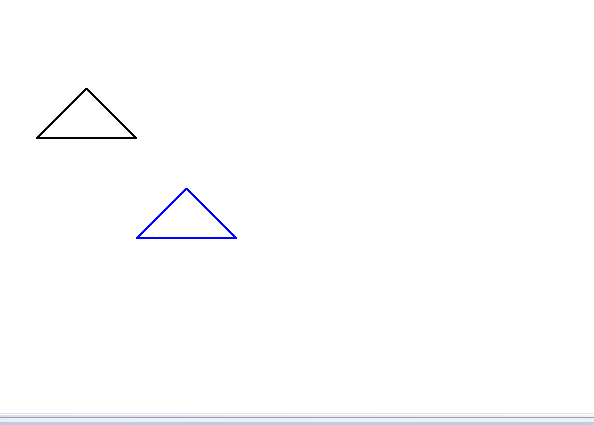
}

pen1.DeleteObject();

pen2.DeleteObject();

}

## 运行截图



## 17、三维视图对称

算法代码：

void CzuoyeView::OnSanDuichen()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1, pen2;

pen1.CreatePen(PS\_SOLID, 5, RGB(0, 0, 100));

pen2.CreatePen(PS\_SOLID, 5, RGB(0, 0, 200));

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]);

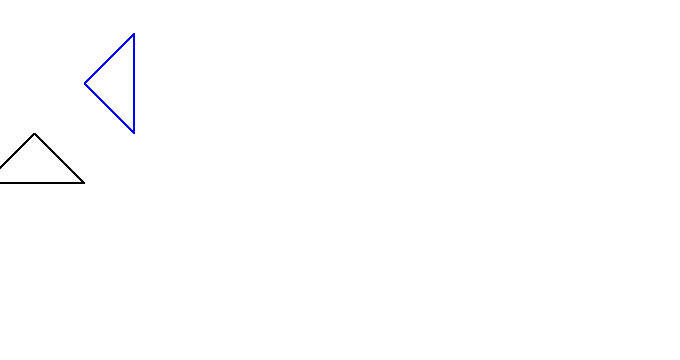
}

pen1.DeleteObject();

pen2.DeleteObject();

}

## 运行截图



## 18、三维视图错切

算法代码：

void CzuoyeView::OnSanCuoqie()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1, pen2;

pen1.CreatePen(PS\_SOLID, 5, RGB(0, 0, 100));

pen2.CreatePen(PS\_SOLID, 5, RGB(0, 0, 200));

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]);

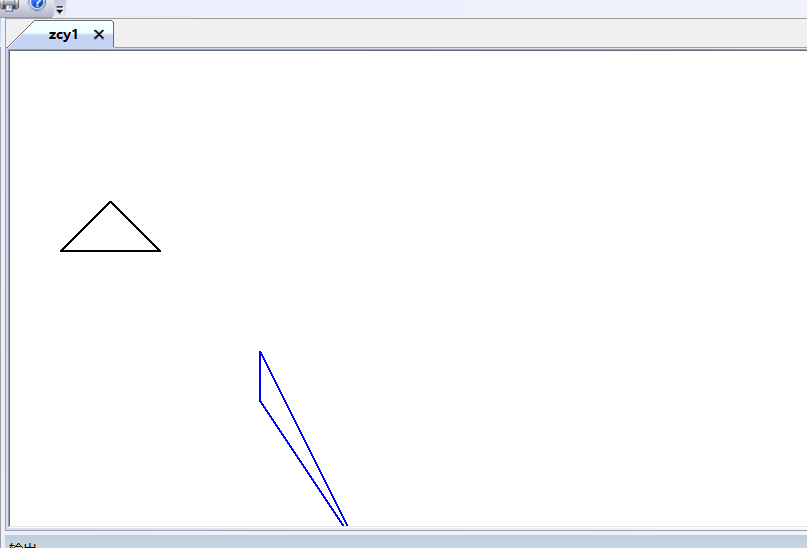
}

pen1.DeleteObject();

pen2.DeleteObject();

}

## 运行截图



## 19、三维视图旋转

算法代码：

void CzuoyeView::OnSanXuanzhuan()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1, pen2;

pen1.CreatePen(PS\_SOLID, 5, RGB(0, 0, 100));

pen2.CreatePen(PS\_SOLID, 5, RGB(0, 0, 200));

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));

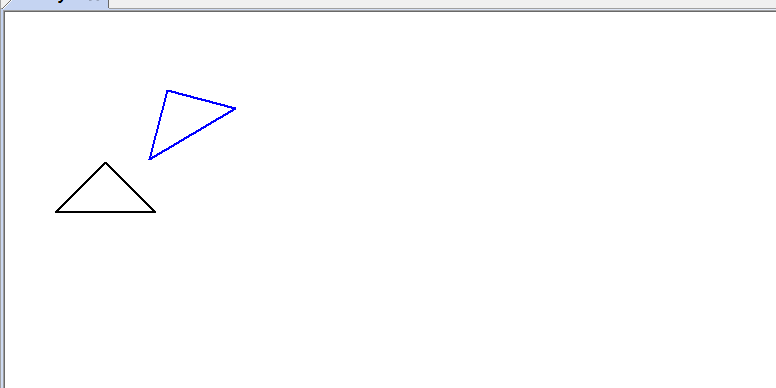
}

pen1.DeleteObject();

pen2.DeleteObject();

}

## 运行截图



## 20、三维视图比例

算法代码：

void CzuoyeView::OnSanBili()

{

// TODO: 在此添加命令处理程序代码

CClientDC dc(this);

CPen pen1, pen2;

pen1.CreatePen(PS\_SOLID, 5, RGB(0, 0, 100));

pen2.CreatePen(PS\_SOLID, 5, RGB(0, 0, 200));

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]);

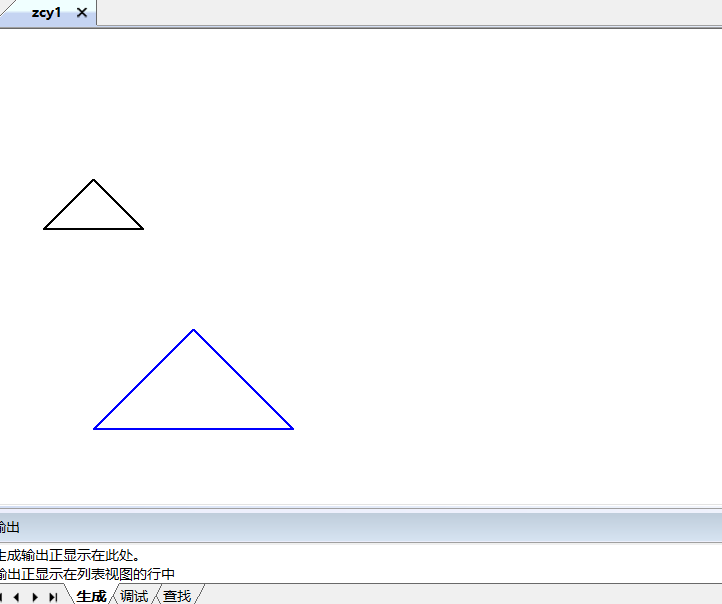
}

pen1.DeleteObject();

pen2.DeleteObject();

}

## 运行截图



## 21 三视图

算法代码：

void CzuoyeView::OnSanSanshitu()

{

// TODO: 在此添加命令处理程序代码

CDC\* pDC = GetDC();

int a[10][4] = { {0,0,0,1},{0,0,2,1},{1,0,2,1},{1,0,0,1},{0,1,0,1},{0,1,1,1},{0,1,2,1},{1,1,2,1},{1,2,1,1},{1,2,0,1} };

int T1[4][4] = { {1,0,0,0},{0,1,0,0},{0,0,0,0},{0,0,0,1} };

int T2[4][4] = { {0,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1} };

int Ry[4][4] = { {0,0,-1,0},{0,1,0,0},{1,0,0,0},{0,0,0,0} };

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

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

int t[4][4], b[10][4], c[10][4], d[10][4];

int i, j, k, temp;

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

{

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

{

temp = 0;

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

{

temp = temp + T2[i][k] \* Mx[k][j];

}

t[i][j] = temp;

}

}

int Th[4][4] = { {1,0,0,0},{0,0,0,0},{0,-1,0,0},{0,5,0,1} };

for (i = 0; i < 10; i++)

{

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

{

temp = 0;

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

{

temp = temp + a[i][k] \* T1[k][j];

}

b[i][j] = temp;

}

}

CPen pen1(PS\_SOLID, 1, RGB(255, 0, 0));

CPen\* p1, \* p2;

p1 = pDC->SelectObject(&pen1);

//1-2

pDC->MoveTo(b[0][0] \* 50 \* 2, b[0][1] \* 50 \* 2);

pDC->LineTo(b[1][0] \* 50 \* 2, b[1][1] \* 50 \* 2);

//1-4

pDC->MoveTo(b[0][0] \* 50 \* 2, b[0][1] \* 50 \* 2);

pDC->LineTo(b[3][0] \* 50 \* 2, b[3][1] \* 50 \* 2);

//1-5

pDC->MoveTo(b[0][0] \* 50 \* 2, b[0][1] \* 50 \* 2);

pDC->LineTo(b[4][0] \* 50 \* 2, b[4][1] \* 50 \* 2);

//2-3

pDC->MoveTo(b[1][0] \* 50 \* 2, b[1][1] \* 50 \* 2);

pDC->LineTo(b[2][0] \* 50 \* 2, b[2][1] \* 50 \* 2);

//2-7

pDC->MoveTo(b[1][0] \* 50 \* 2, b[1][1] \* 50 \* 2);

pDC->LineTo(b[6][0] \* 50 \* 2, b[6][1] \* 50 \* 2);

//3-4

pDC->MoveTo(b[2][0] \* 50 \* 2, b[2][1] \* 50 \* 2);

pDC->LineTo(b[3][0] \* 50 \* 2, b[3][1] \* 50 \* 2);

//3-8

pDC->MoveTo(b[2][0] \* 50 \* 2, b[2][1] \* 50 \* 2);

pDC->LineTo(b[7][0] \* 50 \* 2, b[7][1] \* 50 \* 2);

//4-10

pDC->MoveTo(b[3][0] \* 50 \* 2, b[3][1] \* 50 \* 2);

pDC->LineTo(b[9][0] \* 50 \* 2, b[9][1] \* 50 \* 2);

//5-6

pDC->MoveTo(b[4][0] \* 50 \* 2, b[4][1] \* 50 \* 2);

pDC->LineTo(b[5][0] \* 50 \* 2, b[5][1] \* 50 \* 2);

//5-10

pDC->MoveTo(b[4][0] \* 50 \* 2, b[4][1] \* 50 \* 2);

pDC->LineTo(b[9][0] \* 50 \* 2, b[9][1] \* 50 \* 2);

//6-7

pDC->MoveTo(b[5][0] \* 50 \* 2, b[5][1] \* 50 \* 2);

pDC->LineTo(b[6][0] \* 50 \* 2, b[6][1] \* 50 \* 2);

//6-8

pDC->MoveTo(b[5][0] \* 50 \* 2, b[5][1] \* 50 \* 2);

pDC->LineTo(b[7][0] \* 50 \* 2, b[7][1] \* 50 \* 2);

//6-9

pDC->MoveTo(b[5][0] \* 50 \* 2, b[5][1] \* 50 \* 2);

pDC->LineTo(b[8][0] \* 50 \* 2, b[8][1] \* 50 \* 2);

//7-8

pDC->MoveTo(b[6][0] \* 50 \* 2, b[6][1] \* 50 \* 2);

pDC->LineTo(b[7][0] \* 50 \* 2, b[7][1] \* 50 \* 2);

//8-9

pDC->MoveTo(b[7][0] \* 50 \* 2, b[7][1] \* 50 \* 2);

pDC->LineTo(b[8][0] \* 50 \* 2, b[8][1] \* 50 \* 2);

//9-10

pDC->MoveTo(b[8][0] \* 50 \* 2, b[8][1] \* 50 \* 2);

pDC->LineTo(b[9][0] \* 50 \* 2, b[9][1] \* 50 \* 2);

for (i = 0; i < 10; i++)

{

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

{

temp = 0;

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

{

temp = temp + a[i][k] \* Tw[k][j];

}

c[i][j] = temp;

}

}

CPen pen2(PS\_SOLID, 1, RGB(128, 128, 0));

p2 = pDC->SelectObject(&pen2);

//1-2

pDC->MoveTo(c[0][0] \* 50 \* 2, c[0][1] \* 50 \* 2);

pDC->LineTo(c[1][0] \* 50 \* 2, c[1][1] \* 50 \* 2);

//1-4

pDC->MoveTo(c[0][0] \* 50 \* 2, c[0][1] \* 50 \* 2);

pDC->LineTo(c[3][0] \* 50 \* 2, c[3][1] \* 50 \* 2);

//1-5

pDC->MoveTo(c[0][0] \* 50 \* 2, c[0][1] \* 50 \* 2);

pDC->LineTo(c[4][0] \* 50 \* 2, c[4][1] \* 50 \* 2);

//2-3

pDC->MoveTo(c[1][0] \* 50 \* 2, c[1][1] \* 50 \* 2);

pDC->LineTo(c[2][0] \* 50 \* 2, c[2][1] \* 50 \* 2);

//2-7

pDC->MoveTo(c[1][0] \* 50 \* 2, c[1][1] \* 50 \* 2);

pDC->LineTo(c[6][0] \* 50 \* 2, c[6][1] \* 50 \* 2);

//3-4

pDC->MoveTo(c[2][0] \* 50 \* 2, c[2][1] \* 50 \* 2);

pDC->LineTo(c[3][0] \* 50 \* 2, c[3][1] \* 50 \* 2);

//3-8

pDC->MoveTo(c[2][0] \* 50 \* 2, c[2][1] \* 50 \* 2);

pDC->LineTo(c[7][0] \* 50 \* 2, c[7][1] \* 50 \* 2);

//4-10

pDC->MoveTo(c[3][0] \* 50 \* 2, c[3][1] \* 50 \* 2);

pDC->LineTo(c[9][0] \* 50 \* 2, c[9][1] \* 50 \* 2);

//5-10

pDC->MoveTo(c[4][0] \* 50 \* 2, c[4][1] \* 50 \* 2);

pDC->LineTo(c[9][0] \* 50 \* 2, c[9][1] \* 50 \* 2);

//5-6

CPen pen3(PS\_DASH, 1, RGB(128, 128, 0));

p2 = pDC->SelectObject(&pen3);

pDC->MoveTo(c[4][0] \* 50 \* 2, c[4][1] \* 50 \* 2);

pDC->LineTo(c[5][0] \* 50 \* 2, c[5][1] \* 50 \* 2);

//6-7

pDC->MoveTo(c[5][0] \* 50 \* 2, c[5][1] \* 50 \* 2);

pDC->LineTo(c[6][0] \* 50 \* 2, c[6][1] \* 50 \* 2);

//6-8

pDC->MoveTo(c[5][0] \* 50 \* 2, c[5][1] \* 50 \* 2);

pDC->LineTo(c[7][0] \* 50 \* 2, c[7][1] \* 50 \* 2);

//6-9

pDC->MoveTo(c[5][0] \* 50 \* 2, c[5][1] \* 50 \* 2);

pDC->LineTo(c[8][0] \* 50 \* 2, c[8][1] \* 50 \* 2);

//7-8

pDC->MoveTo(c[6][0] \* 50 \* 2, c[6][1] \* 50 \* 2);

pDC->LineTo(c[7][0] \* 50 \* 2, c[7][1] \* 50 \* 2);

CPen pen4(PS\_SOLID, 1, RGB(128, 128, 0));

p2 = pDC->SelectObject(&pen4);

//8-9

pDC->MoveTo(c[7][0] \* 50 \* 2, c[7][1] \* 50 \* 2);

pDC->LineTo(c[8][0] \* 50 \* 2, c[8][1] \* 50 \* 2);

//9-10

pDC->MoveTo(c[8][0] \* 50 \* 2, c[8][1] \* 50 \* 2);

pDC->LineTo(c[9][0] \* 50 \* 2, c[9][1] \* 50 \* 2);

for (i = 0; i < 10; i++)

{

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

{

temp = 0;

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

{

temp = temp + a[i][k] \* Th[k][j];

}

d[i][j] = temp;

}

}

CPen pen5(PS\_SOLID, 1, RGB(0, 0, 255));

p1 = pDC->SelectObject(&pen5);

//1-2

pDC->MoveTo(d[0][0] \* 50 \* 2, d[0][1] \* 50 \* 2);

pDC->LineTo(d[1][0] \* 50 \* 2, d[1][1] \* 50 \* 2);

//1-4

pDC->MoveTo(d[0][0] \* 50 \* 2, d[0][1] \* 50 \* 2);

pDC->LineTo(d[3][0] \* 50 \* 2, d[3][1] \* 50 \* 2);

//1-5

pDC->MoveTo(d[0][0] \* 50 \* 2, d[0][1] \* 50 \* 2);

pDC->LineTo(d[4][0] \* 50 \* 2, d[4][1] \* 50 \* 2);

//2-3

pDC->MoveTo(d[1][0] \* 50 \* 2, d[1][1] \* 50 \* 2);

pDC->LineTo(d[2][0] \* 50 \* 2, d[2][1] \* 50 \* 2);

//2-7

pDC->MoveTo(d[1][0] \* 50 \* 2, d[1][1] \* 50 \* 2);

pDC->LineTo(d[6][0] \* 50 \* 2, d[6][1] \* 50 \* 2);

//3-4

pDC->MoveTo(d[2][0] \* 50 \* 2, d[2][1] \* 50 \* 2);

pDC->LineTo(d[3][0] \* 50 \* 2, d[3][1] \* 50 \* 2);

//3-8

pDC->MoveTo(d[2][0] \* 50 \* 2, d[2][1] \* 50 \* 2);

pDC->LineTo(d[7][0] \* 50 \* 2, d[7][1] \* 50 \* 2);

//4-10

pDC->MoveTo(d[3][0] \* 50 \* 2, d[3][1] \* 50 \* 2);

pDC->LineTo(d[9][0] \* 50 \* 2, d[9][1] \* 50 \* 2);

//5-6

pDC->MoveTo(d[4][0] \* 50 \* 2, d[4][1] \* 50 \* 2);

pDC->LineTo(d[5][0] \* 50 \* 2, d[5][1] \* 50 \* 2);

//5-10

pDC->MoveTo(d[4][0] \* 50 \* 2, d[4][1] \* 50 \* 2);

pDC->LineTo(d[9][0] \* 50 \* 2, d[9][1] \* 50 \* 2);

//6-7

pDC->MoveTo(d[5][0] \* 50 \* 2, d[5][1] \* 50 \* 2);

pDC->LineTo(d[6][0] \* 50 \* 2, d[6][1] \* 50 \* 2);

//6-8

pDC->MoveTo(d[5][0] \* 50 \* 2, d[5][1] \* 50 \* 2);

pDC->LineTo(d[7][0] \* 50 \* 2, d[7][1] \* 50 \* 2);

//6-9

pDC->MoveTo(d[5][0] \* 50 \* 2, d[5][1] \* 50 \* 2);

pDC->LineTo(d[8][0] \* 50 \* 2, d[8][1] \* 50 \* 2);

//7-8

pDC->MoveTo(d[6][0] \* 50 \* 2, d[6][1] \* 50 \* 2);

pDC->LineTo(d[7][0] \* 50 \* 2, d[7][1] \* 50 \* 2);

//8-9

pDC->MoveTo(d[7][0] \* 50 \* 2, d[7][1] \* 50 \* 2);

pDC->LineTo(d[8][0] \* 50 \* 2, d[8][1] \* 50 \* 2);

//9-10

pDC->MoveTo(d[8][0] \* 50 \* 2, d[8][1] \* 50 \* 2);

pDC->LineTo(d[9][0] \* 50 \* 2, d[9][1] \* 50 \* 2);

}

运行抓图：

