# 《计算机图形学与数字地图》 上机实验报告（2018级）

**姓名 赵开宇**

**班级 地信18-1班**

**学号 07182450**

**环境与测绘学院**

实验一：金刚石图案的绘制

1. 实验目的

对CDC类有初步了解。

1. 实验内容

1、利用 CDC 类提供的功能实现金刚石图案的绘制

2、利用 CDC 类的文本绘制功能添加一个“金刚石”的文本

1. 实验思路

建立两个数组，分别存储每个点的x,y坐标值，而后利用两层for循环多次调用moveto,lineto函数，依次画出每个线条，需要注意每个点只需要与它后面的点连线。

1. 关键代码

对OnDraw函数添加代码：

void C地信1班\_071824550\_1View::OnDraw(CDC\* pDC)

{

C地信1班\_071824550\_1Doc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

CPen mypen1(6, 2, RGB(155, 80, 255)); //创建画笔

CPen\* oldpen = pDC->SelectObject(&mypen1); //将画笔引入当前窗口

pDC->Ellipse(300, 100, 1000, 800); //画圆

double pi = 3.1415926; //定义圆周率

int mypointx[30], mypointy[30]; //两个数组分别存储x和y坐标

for (int i = 0; i < 30; i++)//记录点坐标 //求取每个点的坐标值

{

mypointx[i] = 650 + 350 \* cos(2 \* pi / 30 \* i);

mypointy[i] = 450 + 350 \* sin(2 \* pi / 30 \* i);

}

for (int j = 0; j < 30; j++) //依次连线

{

for (int k = j + 1; k < 30; k++) //每个点只需要与之前未连过线的点连线

{

pDC->MoveTo(mypointx[j], mypointy[j]);

pDC->LineTo(mypointx[k], mypointy[k]);

}

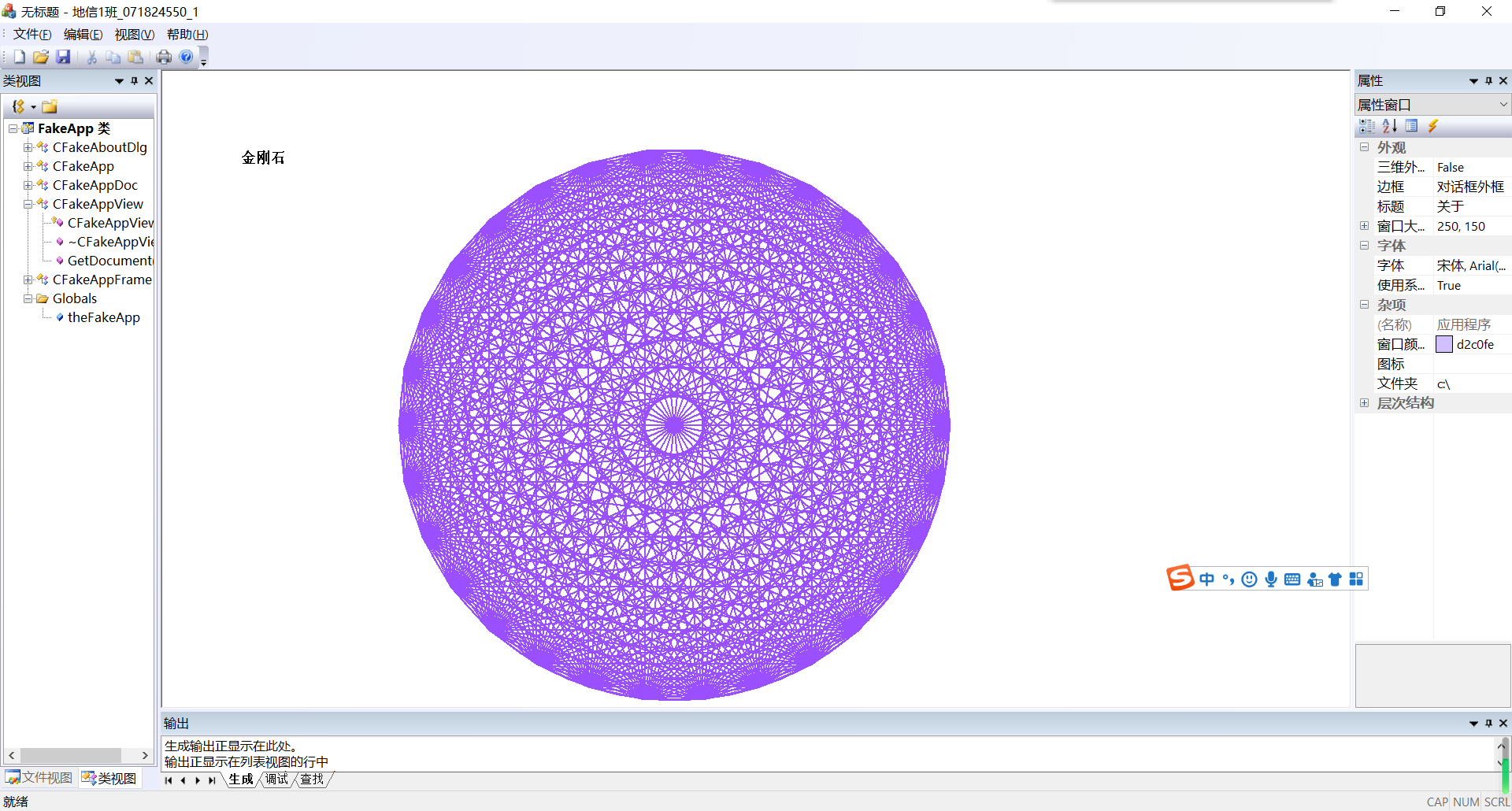
}

pDC->SelectObject(oldpen); //还原笔

pDC->TextOut(100, 100, L"金刚石"); // 显示汉字

}

1. 实验结果



1. 实验体会

MFC工程可以方便的进行点，线和各种矢量图形的绘制

实验二：地图数据库与地图绘制

1. 实验目的

了解地图数据库基本原理，掌握地图绘制编程方法

1. 实验内容

1）对于给定的地图数据库，将地图数据库中的线要素及点要素读取出来， 并绘制相应地图至屏幕区域上，如下图所示。

2）将点的名称以注记形式显示在地图上。

1. 实验思路

定义两个类，分别是边境点类和城市点类，把这两个个类的指针作为CDC的成员变量，绘图时直接调用这些点的坐标信息即可。定义一个read函数，用于从文件中读取边境点数据和城市点数据，然后添加鼠标动作，选择菜单栏中的文件-打开菜单绘图。

1. 关键代码

在头文件“地信1班\_07182450\_2View.h”中添加如下类，并包含所需头文件：

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

class chinapoint //边境点

{

public:

int id; //边境点id

int x; //边境点横坐标

int y; //边境点纵坐标

void setchinapoint(int id1,int x1,int y1) //对点进行设置

{id=id1;

x=x1;

y=y1;}

};

class citypoint //城市点

{

public:

int id; //城市点ID

string name; //城市名称

int rank,x,y; //rank，横坐标，纵坐标

void setcitypoint(int id1,string name1,int rank1,int x1,int y1) //设置城市点

{

id=id1;

name=name1;

rank=rank1;

x=x1;

y=y1;

}

};

在class C地信1班\_07182450\_2View : public CView中添加如下成员函数和成员变量：

void read();

int chinapcount1; //边界点数量

int citypcount1; //城市点数量

chinapoint \*chnp1; //边界点指针

citypoint \*ctp1; //城市点指针

在地信1班\_07182450\_2View.cpp中添加如下代码：

C地信1班\_07182450\_2View::C地信1班\_07182450\_2View()

{

// TODO: 在此处添加构造代码

chinapcount1=0; //初始时边界点数量为0

citypcount1=0; //初始时城市点数量为0

chinapoint \*chnp1=NULL; //指针置空

citypoint \*ctp1=NULL;

}

C地信1班\_07182450\_2View::~C地信1班\_07182450\_2View()

{

delete []chnp1; //释放存储空间

delete []ctp1;

}

void C地信1班\_07182450\_2View::read()

{

ifstream openchinap;//打开china数据文件

openchinap.open("China.txt",ios::in|ios::binary);//监测是否正常打开

if(!openchinap)

{

cerr<<"China文件打开出错";

}

ifstream opencityp;//打开china数据文件

opencityp.open("City.txt",ios::in);//监测是否正常打开

if(!opencityp)

{

cerr<<"City文件打开出错";

}

openchinap>>chinapcount1; //获取边境点数量

opencityp>>citypcount1; //获取城市点数量

chnp1=new chinapoint[chinapcount1]; //使pDC的chnp1指向一片连续的存储空间，存储各个边境点

ctp1=new citypoint[citypcount1]; //使pDC的ctp1指向一片连续的存储空间，存储各个城市点

char c; //用于跳过无用字符

for(int i=0;i<12;i++) //跳过边境点表头

{

openchinap>>c;

}

int id1,x1,y1; //用于存储当前读取的边境点的id，横坐标，纵坐标

char temp; //用于跳过逗号

for(int i=0;i<chinapcount1;i++)

{

openchinap>>id1;

openchinap>>temp;

openchinap>>x1;

openchinap>>temp;

openchinap>>y1;

chnp1[i].setchinapoint(id1,x1,y1); //使用 id1,x1,y1初始化设置当前边境点

}

openchinap.close(); //关闭边境点文件

for(int i=0;i<22;i++) //跳过城市点表头

{

opencityp>>c;

}

string name1; //存储当前读取的城市名字

int rank1; //存储当前读取的城市的rank

for(int i=0;i<citypcount1;i++) //读取数据

{

opencityp>>id1 ;

opencityp>>temp;

opencityp>>name1;

opencityp>> temp;

opencityp>>rank1;

opencityp>> temp;

opencityp>>x1;

opencityp>>temp;

opencityp>>y1;

ctp1[i].setcitypoint(id1,name1,rank1,x1,y1);

//使用id1,name1,rank1,x1,y1初始化设置当前城市点

}

opencityp.close();//关闭城市点文件

}

void C地信1班\_07182450\_2View::OnDraw(CDC\* pDC)

{

C地信1班\_07182450\_2Doc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

CPen mypen1(6, 2, RGB(255, 0, 0)); //定义笔

CPen\* oldpen = pDC->SelectObject(&mypen1); //笔引入当前环境变量

for (int i=0;i< chinapcount1-1;i++) //绘制边境

{

pDC->MoveTo(chnp1[i].x,600-chnp1[i].y);

pDC->LineTo( chnp1[i+1].x,600 - chnp1[i+1].y );

}

pDC->SelectObject(oldpen); // 还原笔

for (int i=0;i<citypcount1;i++) //绘制城市点

{

pDC->Ellipse( (ctp1[i].x - ctp1[i].rank),

( 600 - (ctp1[i].y -ctp1[i].rank ) ) ,

( (ctp1[i].x +ctp1[i].rank) ),

( 600 - (ctp1[i].y + ctp1[i].rank ) ) );

//改变文本类型

#ifdef \_UNICODE //如果是unicode工程

USES\_CONVERSION;

CString tempStr(ctp1[i].name.c\_str());

#else //如果是多字节工程

CString ans;

tempStr.Format("%s", ctp[i].name.c\_str());

#endif // \_UNICODE

//显示城市名

pDC->TextOut(ctp1[i].x+3 ,600 - (ctp1[i].y+3),tempStr);

}

}

添加鼠标动作：

void C地信1班\_07182450\_2View::OnFileOpen() //菜单栏打开

{

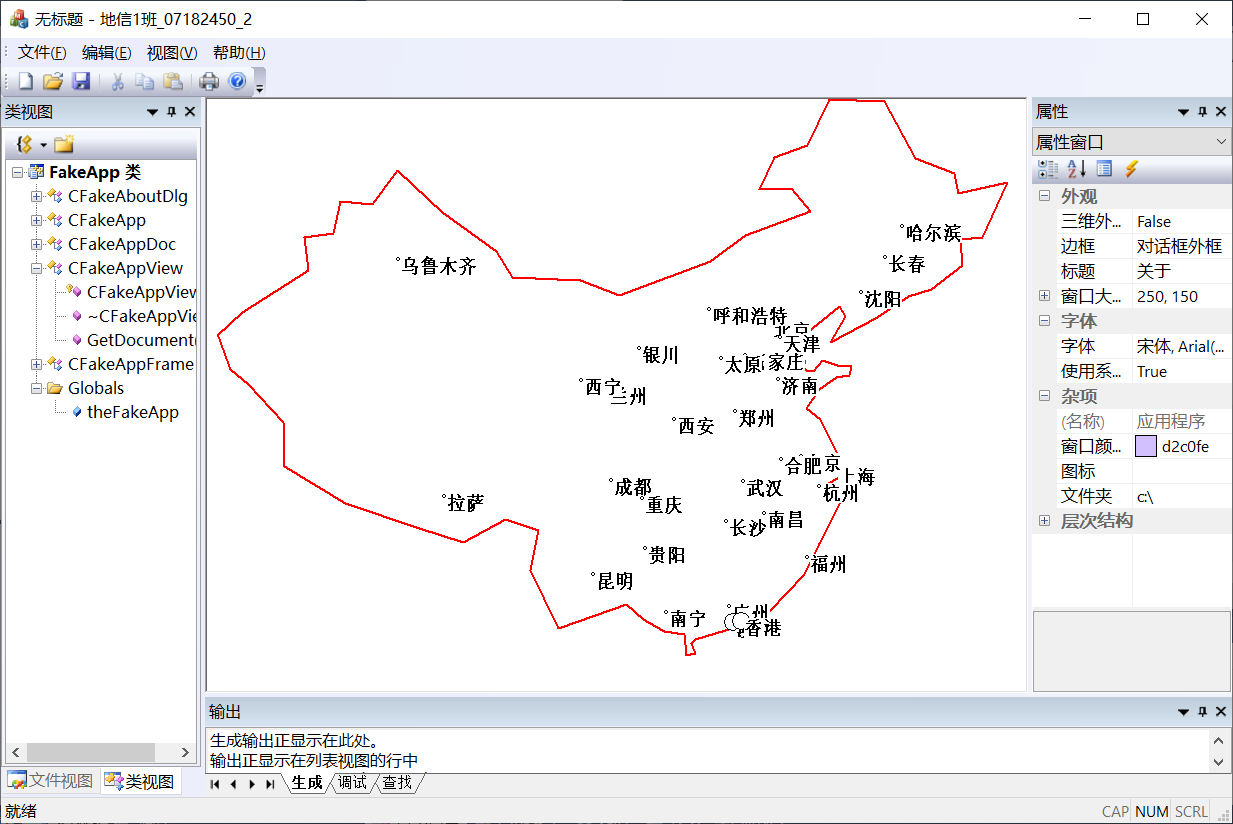
// TODO: 在此添加命令处理程序代码

read(); //读取点数据

Invalidate(); //重新调用OnDraw函数

}

1. 实验结果



1. 实验体会

地图数据库的基本原理就是从数据库中读取相关信息，而后调用绘图语句进行地图绘制。定义一个单独的read函数用于读取文件中的点信息，而后再利用OnDraw绘图，充分体现的面向对象的封装性。

实验三：中点画线算法直线绘制

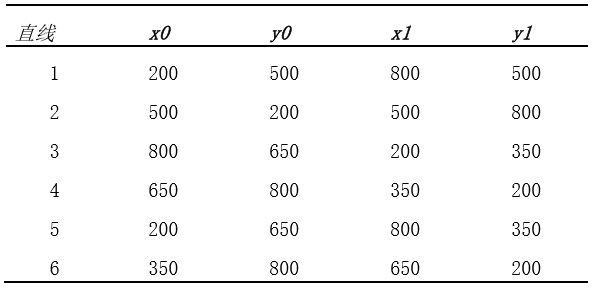
1. 实验目的

1）了解中点画线算法的基本原理；

2）学会使用 VC++实现直线生成算法编程.

1. 实验内容

对于任意输入的两点（斜率任意、起始点顺序任意），利用中点画线算法实 现的直线生成与绘制，要求能正确绘制以下 6 条直线。



1. 实验思路

直线的方向可以分为八个，根据k与1的关系可分为1/4/5/8和2/3/6/7/两组，对于前者x变化更快，应每次变化x，确定y是保持不变还是变化一，后者则是应每次变化y，确定x是保持不变还是变化一；根据起点与终点的关系，可分为4/5/6/7和1/2/3/8两组，第一组交换起点终点后，及和第二组等价。分析过这些后，我们发现可以把八个方向归化为1/8/2/3四个方向，而1/8之间，2/3之间的差别仅在一个负号而已，实质就只需要就1,2两个方向进行分析。

1. 关键代码

void C地信1班\_07182450\_3View::drawline(int x1,int y1,int x2,int y2)

{

CDC \*pDC=GetDC(); //引入当前笔

int x,y,d0,dd1,dd2,a,b,dx,dy,temp;

dx=x2-x1;dy=y2-y1;

//1458象限

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

{

//以下用于把45象限变换为18象限

if (dx<0) //交换起点和终点

{

temp=x1;x1=x2;x2=temp;

temp=y1;y1=y2;y2=temp;

temp=-dx;dx=temp;

temp=-dy;dy=temp;

}

//45象限变换为18象限完毕

a=y1-y2;b=x2-x1;

//开始画线

//第1象限

if(dy>=0)

{

x=x1;y=y1;

d0= 2\*a+b;

dd1=2\*a;dd2=2\*(a+b);

pDC->SetPixel( x,y, RGB(0,0,0)); //绘制起点

while(x<x2) //每次移动x一像素，判断是否到达终点

{

if(d0<0) //y取像正方向的下一点

{x++;y++;d0+=dd2;}

else //y保持不变

{x++;d0+=dd1;}

pDC->SetPixel( x,y, RGB(0,0,0));//绘制下一点

}

}

//第8象限

else

{

x=x1;y=y1;

d0= 2\*a-b;

dd1=2\*a;dd2=2\*(a-b) ;

pDC->SetPixel(x,y, RGB(0,0,0)); //绘制起点

while(x<x2) //每次移动x一像素，判断是否到达终点

{

if(d0<0) //y取像负方向的下一点

{x++;d0+=dd1;}

else //y保持不变

{x++;y--; d0+=dd2;}

pDC->SetPixel( x,y, RGB(0,0,0)); //绘制下一点

}

}

}

//2367象限

else

{

//以下用于把67象限变换为23象限

if (dy<0) //交换起点终点

{

temp=x1;x1=x2;x2=temp;

temp=y1;y1=y2;y2=temp;

temp=-dx;dx=temp;

temp=-dy;dy=temp;

}

//67象限变换为23象限完毕

a=y1-y2;b=x2-x1;

//开始画线

//第2象限

if(dx>=0)

{

x=x1;y=y1;

d0= 2\*b+a;

dd1=2\*b;dd2=2\*(a+b);

pDC->SetPixel( x,y, RGB(0,0,0)); //绘制起点

while(y<y2) //每次移动y一像素，判断是否到达终点

{

if(d0>0) //x取像正方向的下一点

{y++;x++;d0+=dd2;}

else //x不变

{y++;d0+=dd1;}

pDC->SetPixel( x,y, RGB(0,0,0));//绘制下一点

}

}

// 第3象限

else

{

x=x1;y=y1;

d0= 2\*b-a;

dd1=2\*b;dd2=2\*(b-a) ;

pDC->SetPixel(x,y, RGB(0,0,0));

while(y<y2) //每次移动y一像素，判断是否到达终点

{

if(d0>0) //x不变

{y++;d0+=dd1;}

else //x取负正方向的下一点

{y++;x--; d0+=dd2;}

pDC->SetPixel( x,y, RGB(0,0,0));

}

}

}

ReleaseDC(pDC);

}

void C地信1班\_07182450\_3View::OnDraw(CDC\* pDC)

{

C地信1班\_07182450\_3Doc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

//定义笔

CPen newpen(6,1,RGB(255,0,0));

CPen \*oldpen=pDC->SelectObject(&newpen);

//定义笔完毕

//中点画线算法

int \_x1[6]={200,500,800,650,200,350}; //起点横坐标

int \_y1[6]={500,200,650,800,650,800}; //起点纵坐标

int \_x2[6]= {800,500,200,350,800,650}; //终点横坐标

int \_y2[6]={ 500,800,350,200,350,200}; //终点横坐标

for(int i=0;i<6;i++)

{

drawline(\_x1[i],\_y1[i],\_x2[i],\_y2[i]); //调用重点画线算法的函数绘图

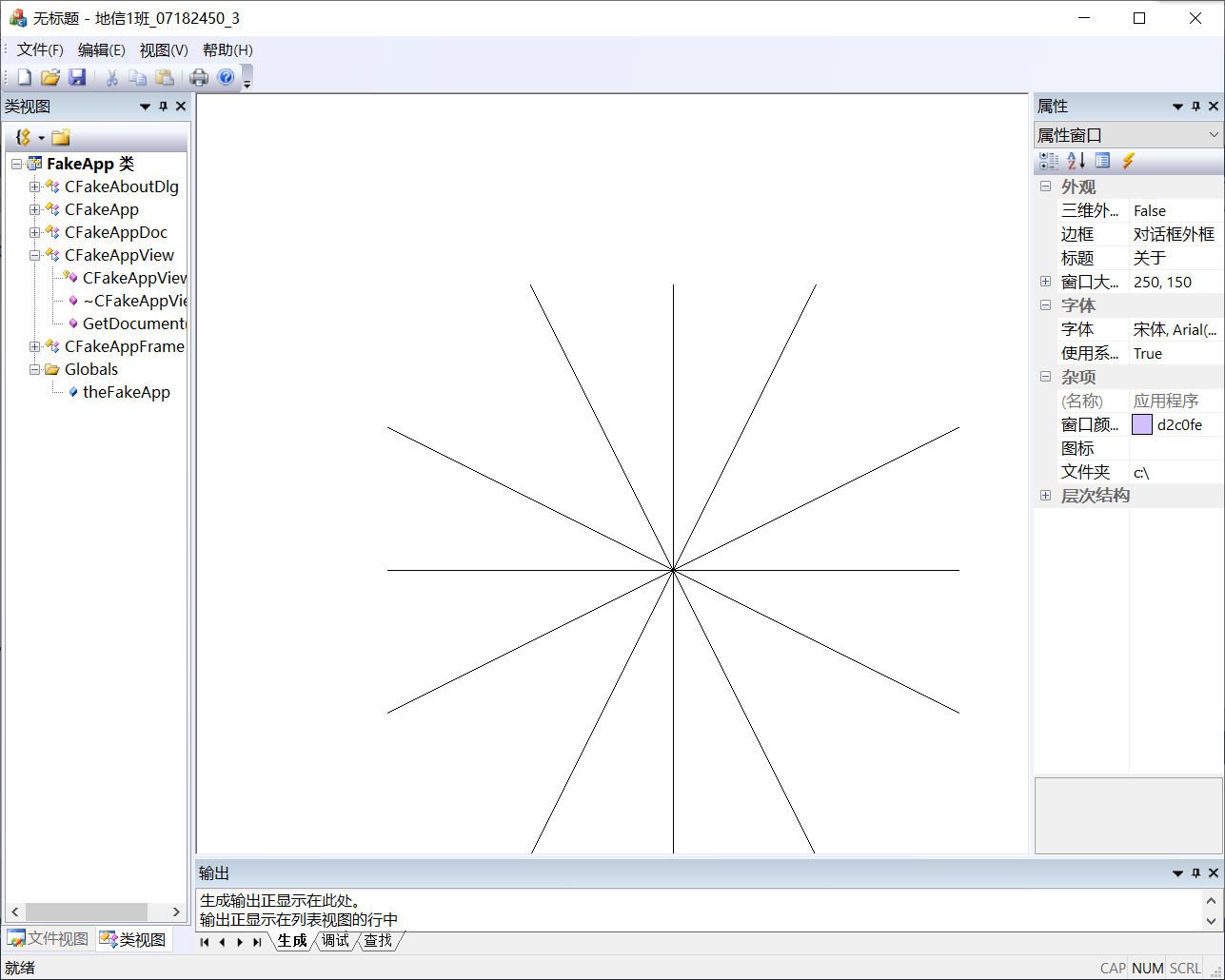
}

//还原笔

pDC->SelectObject(&oldpen);

}

1. 实验结果



1. 实验体会

在开始实践之前，先对八个方向进行分析，找到互通之处，对于可以归于一类处理的方向尽量归于一类，可以大大减少分类讨论的工作量。

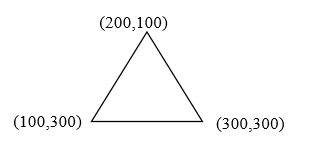
实验四：二维图形变换的编程实现

1. 实验目的

1）了解二维图形变换的基本原理；

2）学会使用 VC++实现图形变换

1. 实验内容
2. 利用实验三的直线绘制功能，绘制下列坐标的三角形。



1. 用程序实现上述三角形关于点（300,300）做顺时针 90°的旋转变换，并 绘制变换后的图形。
2. 实验思路

先定义矩阵乘法函数。而后将三角形平移到原点，而后旋转，再平移回去，就可以得到变化矩阵。将三角形坐标矩阵扩展为三维，与变化矩阵相乘即可得到变化后的点坐标，而后绘制新的三角形

1. 关键代码

在实验三基础上添加如下代码：

void C地信1班\_07182450\_4View::OnDraw(CDC\* pDC)

{

C地信1班\_07182450\_4Doc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

//临时存储矩阵相乘后的结果

double as1[3][3]={{0,0,0},{0,0,0},{0,0,0}};

double as2[3][3]={{0,0,0},{0,0,0},{0,0,0}};

double news[3][3]={{0,0,0},{0,0,0},{0,0,0}};

//原三角形

double s[3][3]={{200,100,1},{100,300,1},{300,300,1}};

//平移，旋转，平移

double t1[3][3]={{1,0,0},{0,1,0},{-300,-300,1}} ;

double t2[3][3]={{0,1,0},{-1,0,0},{0,0,1}};

double t3[3][3]={{1,0,0},{0,1,0},{300,300,1}};

//调用三次矩阵乘法

transform(as1,t1,t2);

transform(as2,as1,t3);

transform(news,s,as2);

//绘制原三角形

drawline (s[0][0],s[0][1],s[1][0],s[1][1]);

drawline (s[1][0],s[1][1],s[2][0],s[2][1]);

drawline (s[0][0],s[0][1],s[2][0],s[2][1]);

//绘制新三角形

drawline (news[0][0],news[0][1],news[1][0],news[1][1]);

drawline (news[1][0],news[1][1],news[2][0],news[2][1]);

drawline (news[0][0],news[0][1],news[2][0],news[2][1]);

}

void C地信1班\_07182450\_4View::transform(double a[][3],double b[][3],double c[][3])

{

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

{

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

{

int temp=0;

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

{

temp=b[i][k]\*c[k][j];

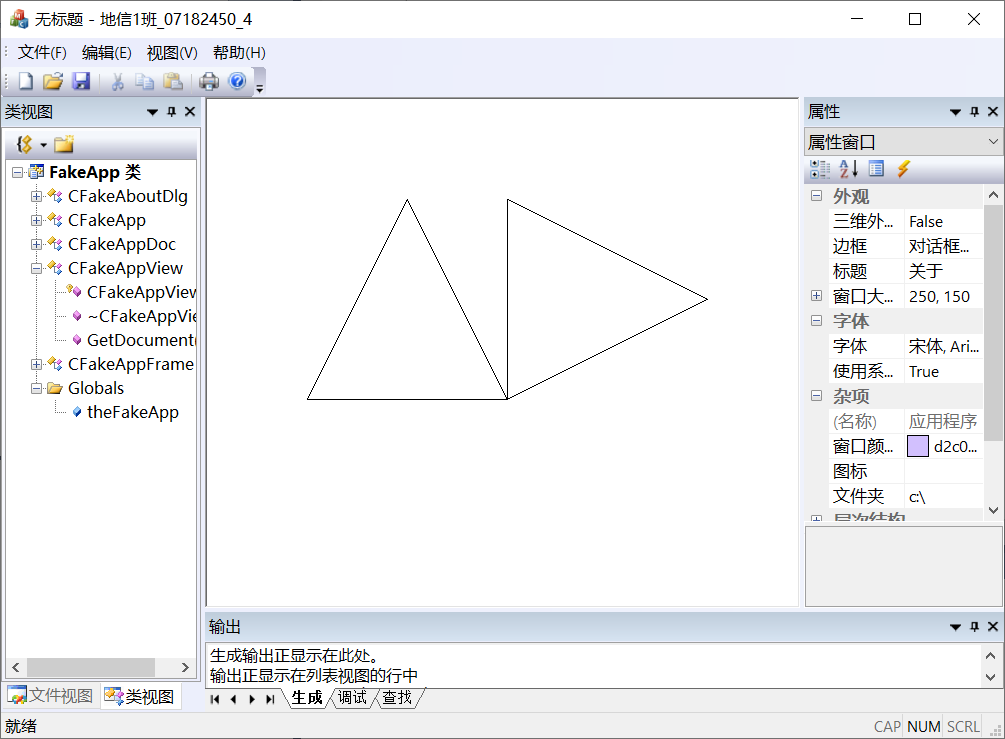
a[i][j]+=temp;

}

}

}

1. 实验结果



1. 实验体会

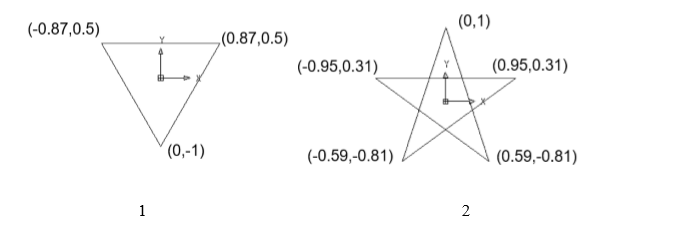
二维图形的基本变换，实质就是定义一个变换矩阵，并且为矩阵设置合适参数，再把二位图型的顶点构成的矩阵和变换矩阵相乘。

实验五：点状地物的符号化

1. 实验目的

了解地图符号化基本原理，掌握点状地图符号的绘制方法。

1. 实验内容
2. 创建一个地图符号库，设计一个统一的地图符号数据结构，并在地图符 号库中添加以下两种地图符号的信息。



2）在实验二的基础上，根据地图符号库的地图符号以及 City 表的 Rank 属性，将不同城市以不同的地图符号绘制出来。

三、 实验思路

定义一个符号类，在cdc类中添加一个符号类的指针作为成员变量，同时定义一个readsym函数，从文件中读取每个符号的信息，利用这个函数初始化cdc中的符号类指针，使它指向一片存储有所有符号信息的一片连续的存储空间，以便在程序运行的开始就加载完所有符号。同时定义一个drawsym函数，把符号平移到指定位置，并进行适当缩放。在OnDraw函数中调用drawsym函数进行地物符号的绘制。

1. 关键代码

在实验二的基础上添加如下代码：

class sym

{

public:

int symnum; //编号

int pcount;//点数量

double\* px;//数组

double\* py;

} ;

virtual ~C地信1班\_07182450\_5View();

void read();

void readsym();

void drawsym(sym tempsym,int tempx,int tempy,int s,CDC\* mydc);

int chinapcount1;

int citypcount1;

int symcount1;

chinapoint \*chnp1;

citypoint \*ctp1;

sym \*mysym;

C地信1班\_07182450\_5View::~C地信1班\_07182450\_5View()//析构

{

for(int i=0;i<symcount1;i++)

{

delete[] mysym[i].px;

delete[] mysym[i].py;

}

delete[] mysym;

}

void C地信1班\_07182450\_5View::readsym()//读入符号信息

{

ifstream opensym;

opensym.open("地图符号库.txt",ios::in||ios::binary);

opensym>>symcount1;

mysym=new sym[symcount1] ;

for(int i=0;i<symcount1;i++)

{

opensym>>mysym[i].symnum;

opensym>>mysym[i].pcount;

mysym[i].px=new double[mysym[i].pcount];

mysym[i].py=new double[mysym[i].pcount];

for(int j=0;j<mysym[i].pcount;j++)

{

opensym>>mysym[i].px[j];

opensym>>mysym[i].py[j];

}

}

opensym.close();

}

void C地信1班\_07182450\_5View::drawsym(sym tempsym,int tempx,int tempy,int s,CDC\* mydc)//绘制

{

int symi=0;

double \*ptr\_x=tempsym.px;//读取

double \*ptr\_y=tempsym.py;

int t=tempsym.pcount;

while(symi<t-1)

{

mydc->MoveTo( ptr\_x[symi]\*s+tempx,//参数列表应该放=0

600-(ptr\_y[symi]\*s+tempy) );

symi++;

mydc->LineTo(ptr\_x[symi]\*s+tempx,

600-(ptr\_y[symi]\*s+tempy));

}

mydc->MoveTo( ptr\_x[t-1]\*s+tempx,

600-(ptr\_y[t-1]\*s+tempy) );//返回原点？

mydc->LineTo(ptr\_x[0]\*s+tempx,

600-(ptr\_y[0]\*s+tempy));

}

void C地信1班\_07182450\_5View::OnDraw(CDC\* pDC)

{

C地信1班\_07182450\_5Doc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

CPen mypen1(6, 2, RGB(255, 0, 0));

CPen\* oldpen = pDC->SelectObject(&mypen1);

for (int i=0;i< chinapcount1-1;i++)

{

pDC->MoveTo(chnp1[i].x,600-chnp1[i].y);

pDC->LineTo( chnp1[i+1].x,600 - chnp1[i+1].y );

}

pDC->SelectObject(oldpen);

readsym();

for (int i=0;i<citypcount1;i++)

{

#ifdef \_UNICODE //如果是unicode工程

USES\_CONVERSION;

CString tempStr(ctp1[i].name.c\_str());

#else //如果是多字节工程

CString ans;

tempStr.Format("%s", ctp1[i].name.c\_str());

#endif // \_UNICODE

pDC->TextOut(ctp1[i].x+3 ,600 - (ctp1[i].y+3),tempStr);

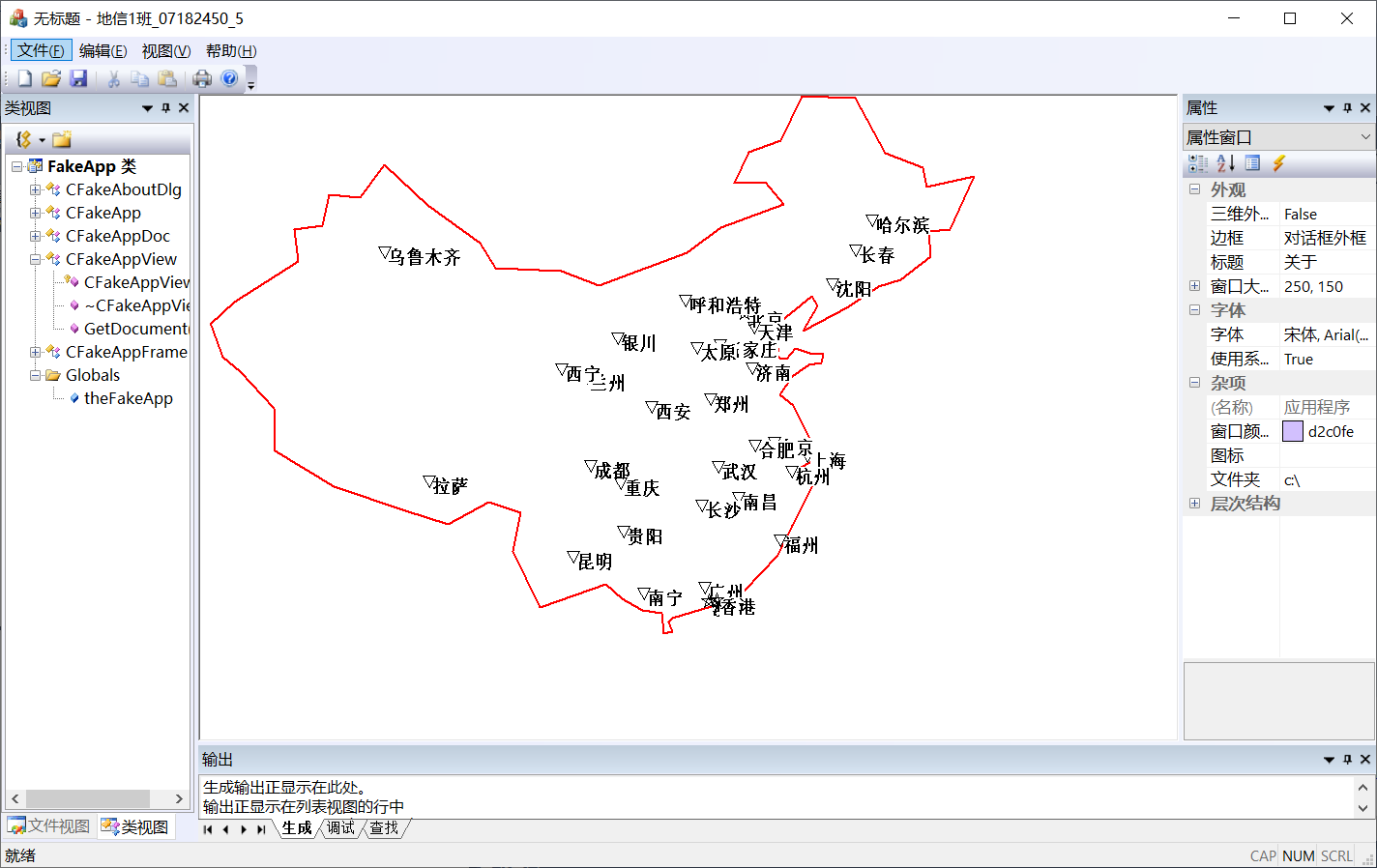
int temprank=ctp1[i].rank-1;

drawsym(mysym[temprank], ctp1[i].x,ctp1[i].y,8,pDC);

}

}

1. 实验结果



1. 实验体会

将一个指向地图符号sym类的动态数组的指针作为CDC的成员变量，将指向存储坐标的动态数组的指针作为sym类的成员变量，通过这两个动态数组的“嵌套”，就可以方便的在调用cdc类的时候通过readsym函数读取地物符号的信息，并且把它全都引入CDC类。