《计算机图形学原理实践》

实验报告

|  |  |  |
| --- | --- | --- |
| 实验名称 | ： | 二维图形几何变换算法 |
|  |  |  |
| 姓 名 | ： |  |
| 学 号 | ： |  |
| 专业班级 | ： |  |
| 实验时间 | ： |  |

西南科技大学计算机科学与技术学院

**一、实验目的**

1、掌握二维平移、比例、旋转几何变换矩阵

2、掌握 相对于任一参考点的比例变换和旋转变换

3、了解定时器的使用方法和边界碰撞检测方法

4、了解静态切分视图框架设计方法

**二、实验步骤**

1、将二维图形顶点表达为二维齐次坐标点

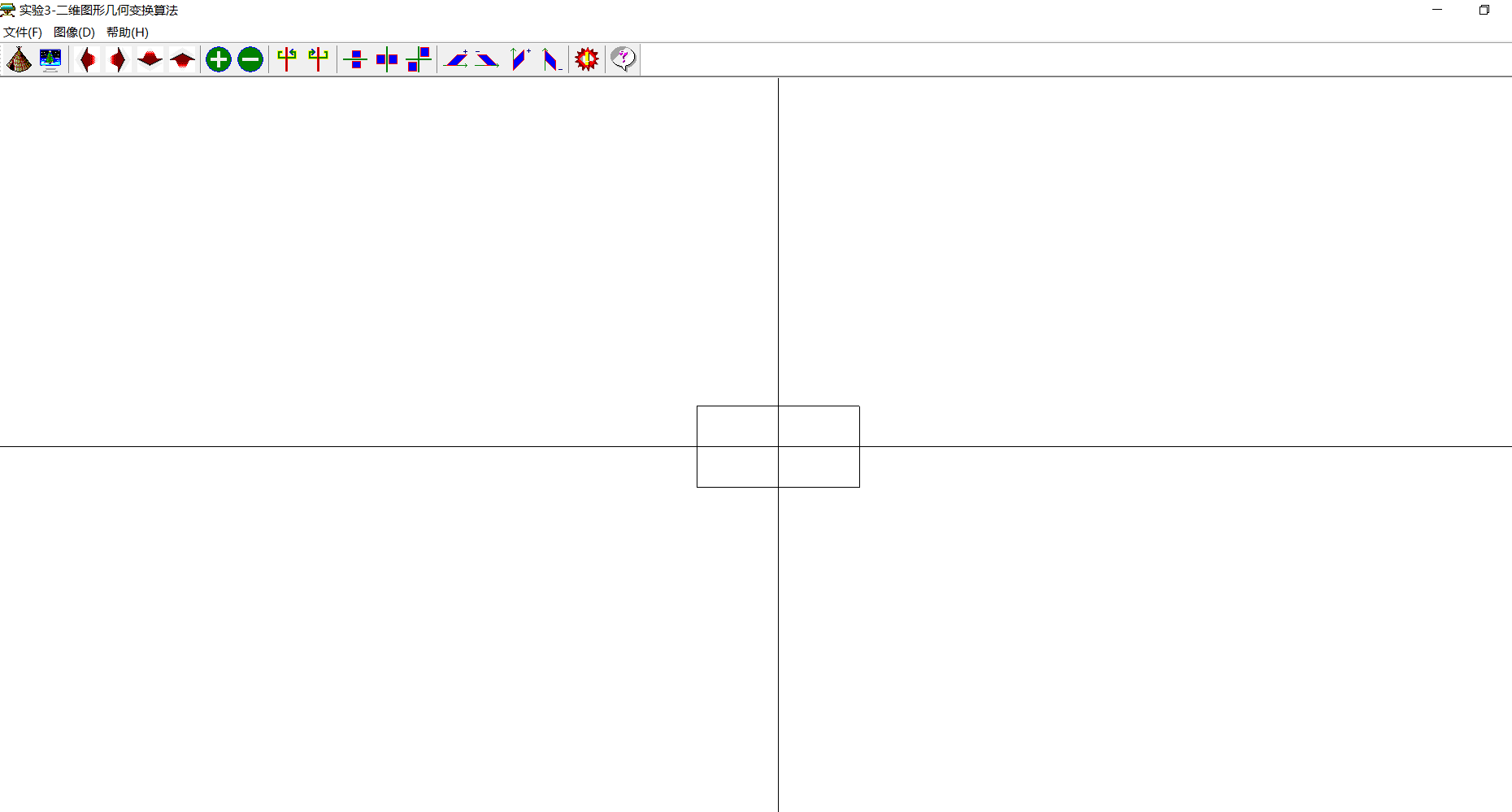
2、使用直线段连接变换前的顶点，绘制变换前的多边形

3、读入单位矩阵，通过修改矩阵来构造平移、比例、旋转、反射和错切矩阵

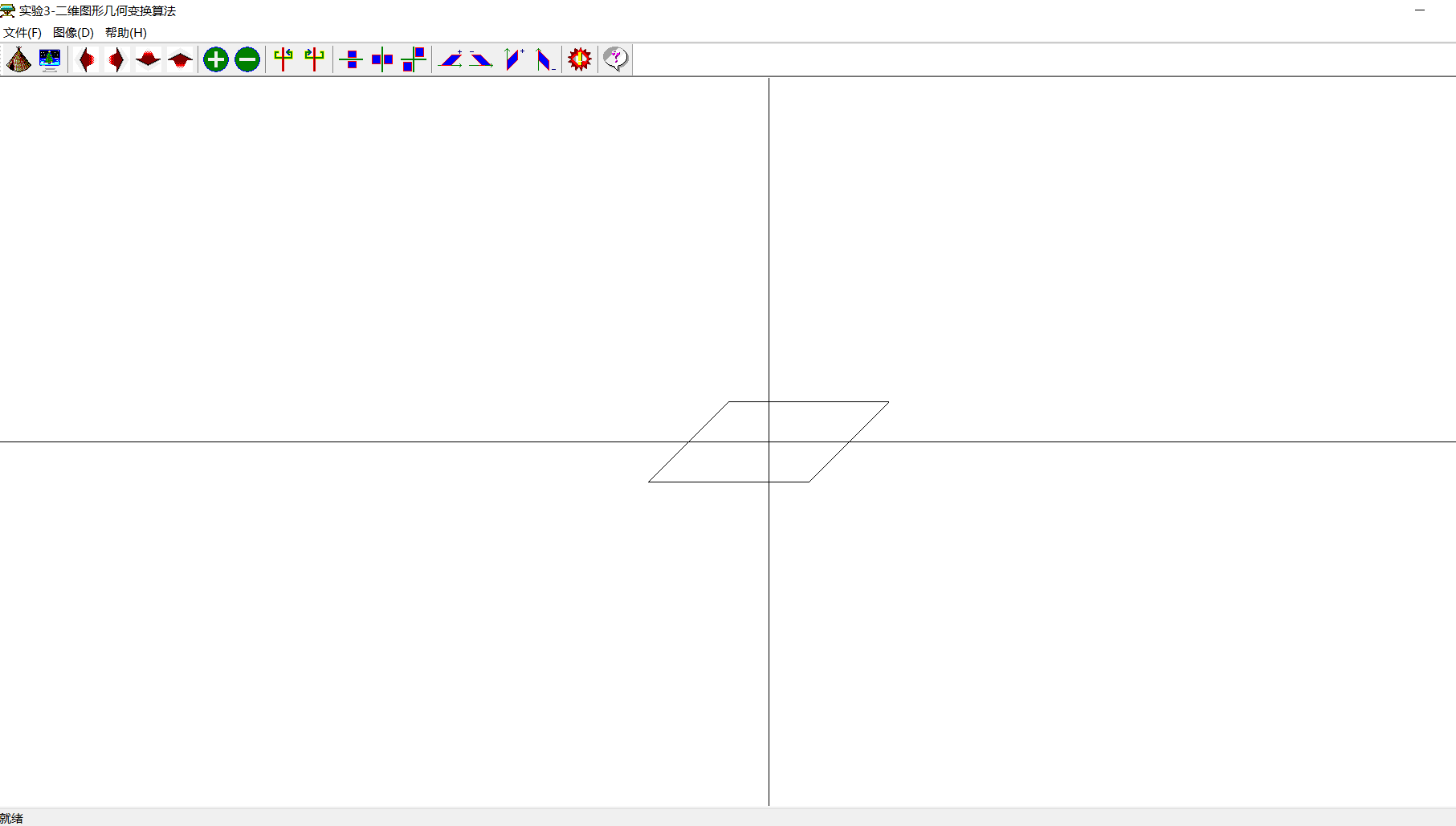
4、将变换前顶点矩阵左乘变换矩阵，得到变换后的顶点矩阵

1. **实验结果**

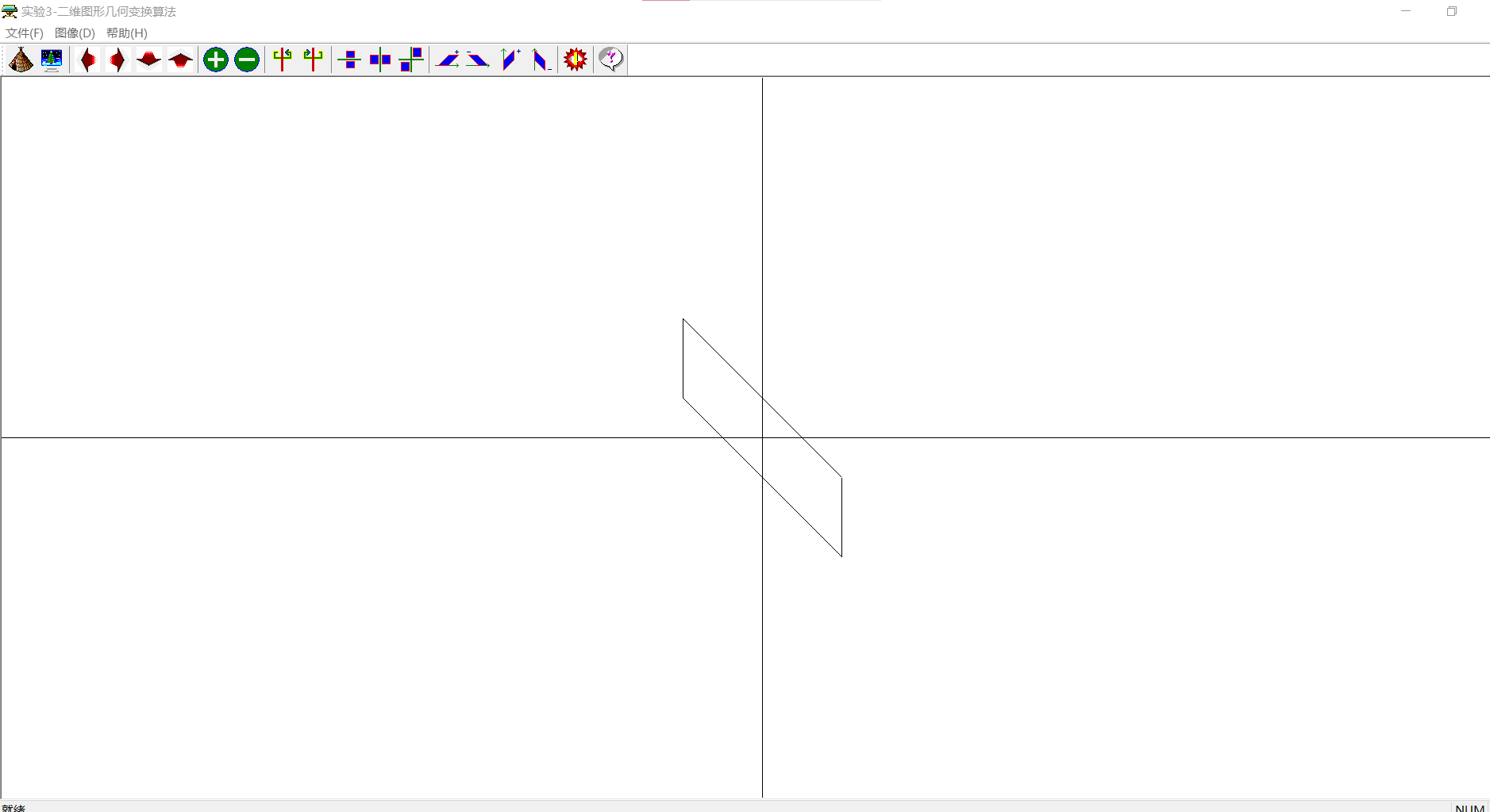
**（1）原始矩阵**



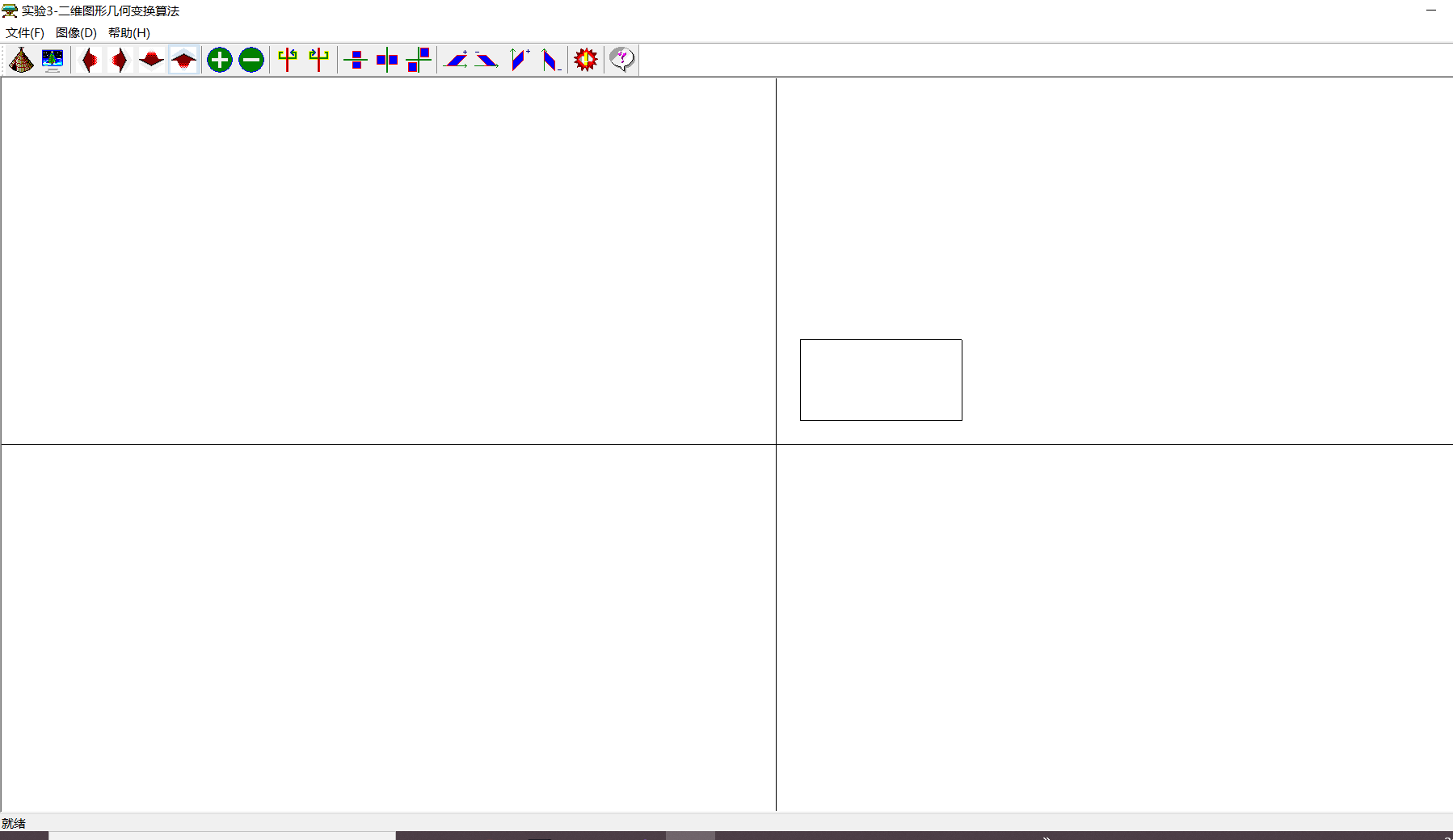
（2）沿X方向错切



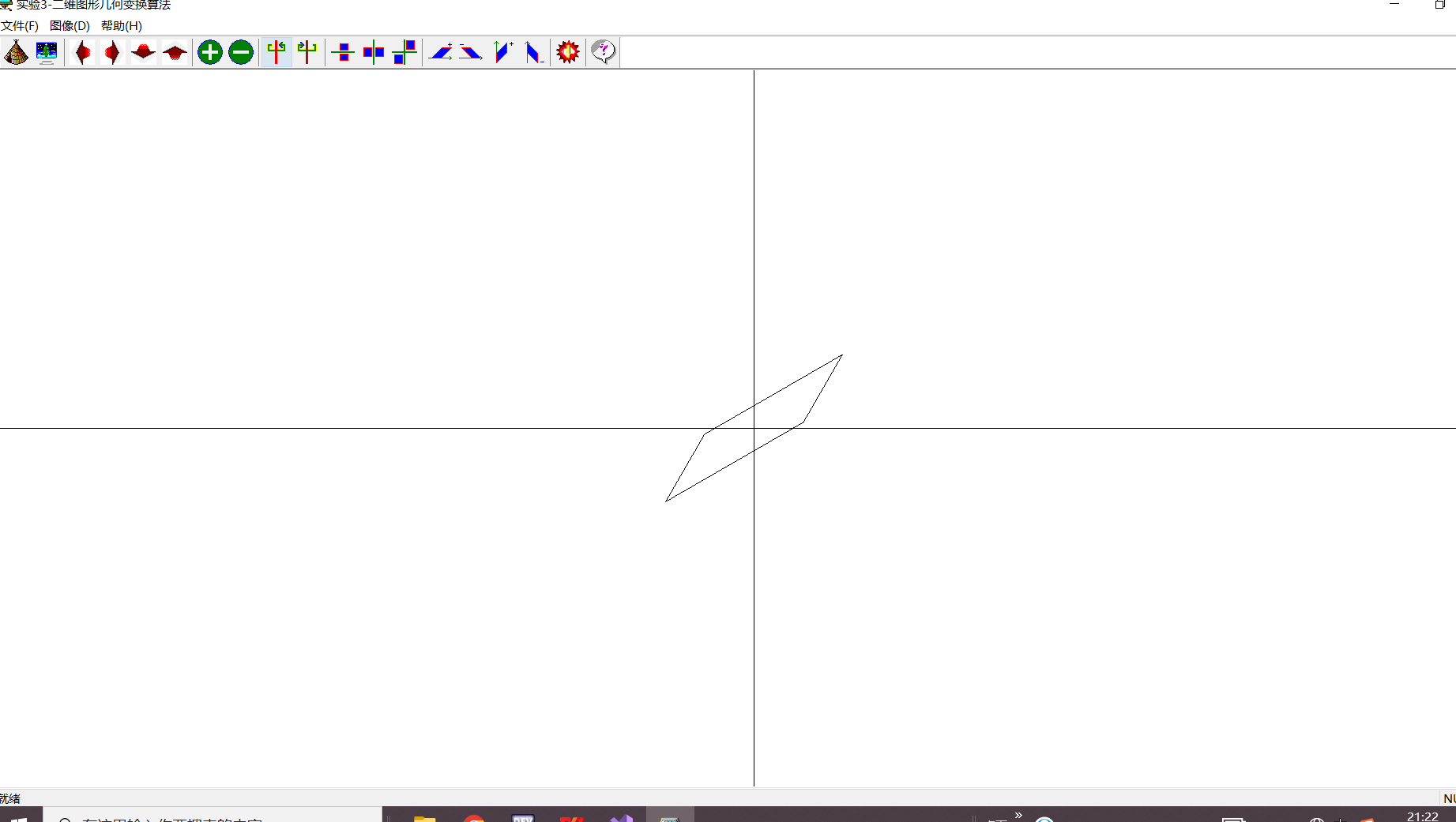
（3）沿Y方向错切



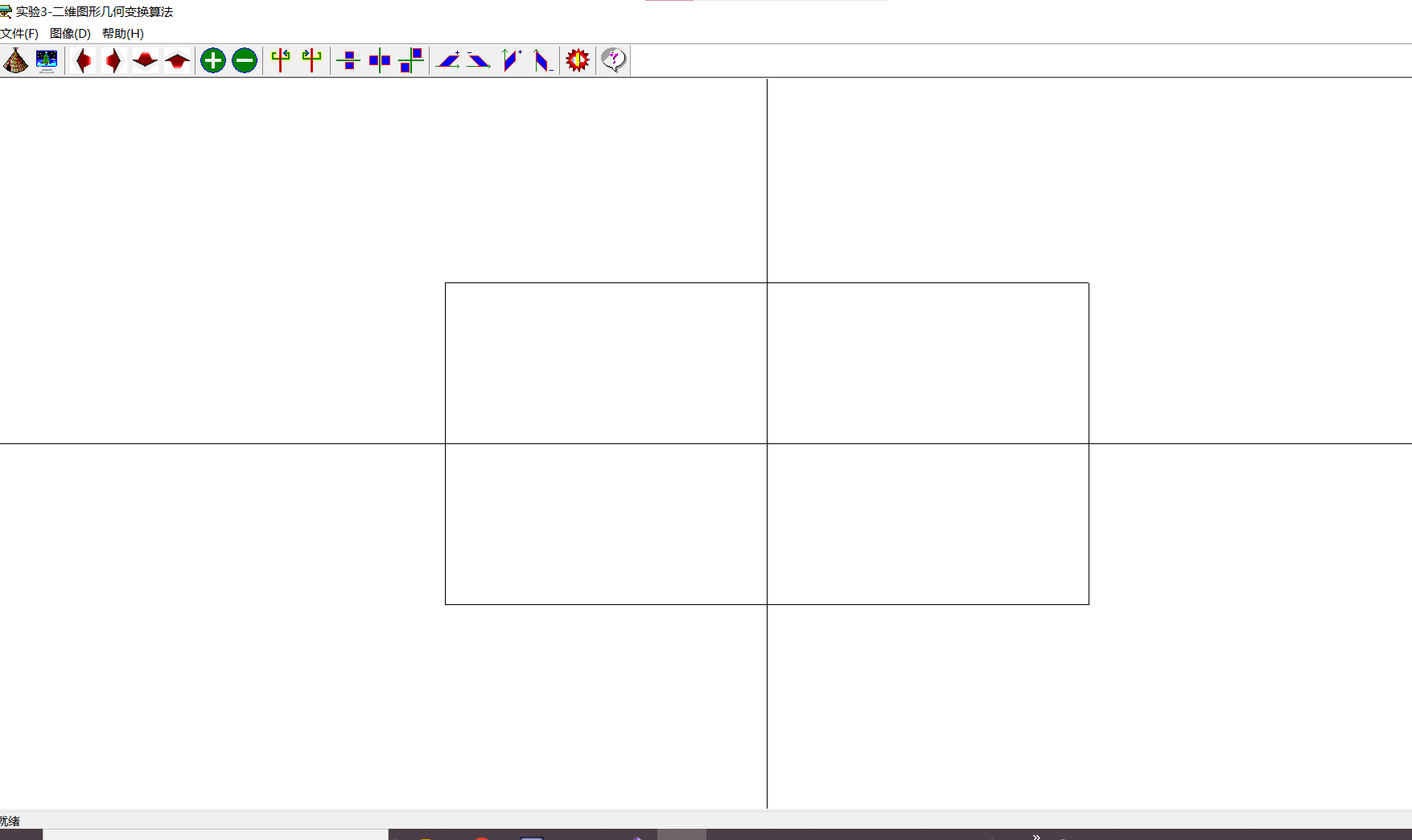
1. 平移



1. 旋转



1. 比例变换



**四、实验总结**

通过本次二维图形变换实验，实现了相关算法原理，绘制出了原始矩阵，并实现其错切、平移、反射、比例变换和旋转。理解了二维变换动画生成办法，即擦除变换前的图形，再用变换后的图形，形成二维变换动画。完成实验过程中，不断试错，最终得出了结果，相信下次的实验会更加的得心应手。

**附录：源代码（关键代码）**

// Transform.h

#if !defined(AFX\_TRANSFORM\_H\_\_C4FA86B2\_0AF5\_409A\_94EB\_B02A1EE13E05\_\_INCLUDED\_)

#define AFX\_TRANSFORM\_H\_\_C4FA86B2\_0AF5\_409A\_94EB\_B02A1EE13E05\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

#include "P2.h"

class CTransform//二维几何变换

{

public:

CTransform();

virtual ~CTransform();

void SetMat(CP2 \*,int);

void Identity();

void Translate(double,double);//平移变换矩阵

void Scale(double,double);//比例变换矩阵

void Scale(double,double,CP2);//相对于任意点的比例变换矩阵

void Rotate(double);//旋转变换矩阵

void Rotate(double,CP2);//相对于任意点的旋转变换矩阵

void ReflectOrg();//原点反射变换矩阵

void ReflectX();//X轴反射变换矩阵

void ReflectY();//Y轴反射变换矩阵

void Shear(double,double);//错切变换矩阵

void MultiMatrix();//矩阵相乘

public:

double T[3][3];

CP2 \*POld;

int num;

};

#endif // !defined(AFX\_TRANSFORM\_H\_\_C4FA86B2\_0AF5\_409A\_94EB\_B02A1EE13E05\_\_INCLUDED\_)

// TransForm.cpp

#include "stdafx.h"

#include "Test.h"

#include "Transform.h"

#include "math.h"

#define PI 3.14159

#ifdef \_DEBUG

#undef THIS\_FILE

static char THIS\_FILE[]=\_\_FILE\_\_;

#define new DEBUG\_NEW

#endif

CTransform::CTransform()

{

}

CTransform::~CTransform()

{

}

void CTransform::SetMat(CP2 \*p,int n)

{

POld=p;

num=n;

}

void CTransform::Identity()//单位矩阵

{

T[0][0] = 1.0;T[0][1] = 0.0;T[0][2] = 0.0;

T[1][0] = 0.0;T[1][1] = 1.0;T[1][2] = 0.0;

T[2][0] = 0.0;T[2][1] = 0.0;T[2][2] = 1.0;

}

void CTransform::Translate(double tx,double ty)//平移变换矩阵

{

Identity();

T[2][0] = tx;

T[2][1] = ty;

MultiMatrix();

}

void CTransform::Scale(double sx,double sy)//比例变换矩阵

{

Identity();

T[0][0] = sx;

T[1][1] = sy;

MultiMatrix();

}

void CTransform::Scale(double sx,double sy,CP2 p)//相对于任意点的整体比例变换矩阵

{

Translate(-p.x ,-p.y );

Scale(sx, sy);

Translate(p.x, p.y);

}

void CTransform::Rotate(double beta)//旋转变换矩阵

{

Identity();

double rad = beta \* PI / 180;

T[0][0] = cos(rad);T[0][1] = sin(rad);

T[1][0] = sin(rad);T[1][1] = cos(rad);

MultiMatrix();

}

void CTransform::Rotate(double beta,CP2 p)//相对于任意点的旋转变换矩阵

{

/\*平移变换，旋转变换矩阵，矩阵相乘\*/

Translate(p.x,p.y);

Rotate(beta);

Translate(p.x, p.y);

}

void CTransform::ReflectOrg()//原点反射变换矩阵

{

Identity();

T[0][0] = -1;

T[1][1] = -1;

MultiMatrix();

}

void CTransform::ReflectX()//X轴反射变换矩阵

{

Identity();

T[0][0] = 1;

T[1][1] = -1;

MultiMatrix();

}

void CTransform::ReflectY()//Y轴反射变换矩阵

{

Identity();

T[0][0] = -1;

T[1][1] = -1;

MultiMatrix();

}

void CTransform::Shear(double b,double c)//错切变换矩阵

{

Identity();

T[0][1] = b;

T[1][0] = c;

MultiMatrix();

}

void CTransform::MultiMatrix()//矩阵相乘

{

CP2\* PNew = new CP2[num];

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

{

PNew[i] = POld[i];

}

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

{

POld[j].x = PNew[j].x \* T[0][0] + PNew[j].y \* T[1][0] + PNew[j].w \* T[2][0];

POld[j].y = PNew[j].x \* T[0][1] + PNew[j].y \* T[1][1] + PNew[j].w \* T[2][1];

POld[j].w = PNew[j].x \* T[0][2] + PNew[j].y \* T[1][2] + PNew[j].w \* T[2][2];

}

delete []PNew;

}