**实验名称：分解复合变换**

1. **具体内容**

**分解复合变换**

1. **设计思路**

**将复合矩阵分解成两个基本矩阵，然后对三角形分别实施这两个基本矩阵**

1. **主要函数及其简要说明**

//齐次旋转  
void my\_rotate\_homogeneous(struct my\_v\_inhomogeneous\* polygon, int polygon\_vertex\_count, double theta)  
{  
    theta = pi \* theta / 180;  
    //装配生成矩阵   
    double translate\_matrix[3][3];  
    memset(translate\_matrix, 0, sizeof(double) \* 9);  
    translate\_matrix[0][0] = translate\_matrix[1][1] = cos(theta);  
    translate\_matrix[0][1] = -sin(theta);  
    translate\_matrix[1][0] = sin(theta);  
    translate\_matrix[2][2] = 1;  
  
    //遍历多边形每个顶点  
    for (int vIndex = 0; vIndex < polygon\_vertex\_count; vIndex++)  
    {  
        struct my\_v\_homogeneous input\_v;  
        input\_v.x = polygon[vIndex].x;  
        input\_v.y = polygon[vIndex].y;  
        input\_v.ratio = 1;  
        input\_v = matrix\_multiply\_vector(translate\_matrix, input\_v); //矩阵作用到每个顶点  
        polygon[vIndex].x = input\_v.x;  
        polygon[vIndex].y = input\_v.y;  
    }  
}

//齐次缩放  
void my\_scale\_homogeneous(struct my\_v\_inhomogeneous\* polygon, int polygon\_vertex\_count, double sx, double sy)  
{  
    //装配生成矩阵   
    double translate\_matrix[3][3];  
    memset(translate\_matrix, 0, sizeof(double) \* 9);  
    translate\_matrix[0][0] = sx;  
    translate\_matrix[1][1] = sy;  
    translate\_matrix[2][2] = 1;  
  
    //遍历多边形每个顶点  
    for (int vIndex = 0; vIndex < polygon\_vertex\_count; vIndex++)  
    {  
        struct my\_v\_homogeneous input\_v;  
        input\_v.x = polygon[vIndex].x;  
        input\_v.y = polygon[vIndex].y;  
        input\_v.ratio = 1;  
        input\_v = matrix\_multiply\_vector(translate\_matrix, input\_v); //矩阵作用到每个顶点  
        polygon[vIndex].x = input\_v.x;  
        polygon[vIndex].y = input\_v.y;  
    }  
}

1. **遇到的主要问题及解决思路：**

**无**

1. **程序运行结果**

