**成都信息工程大学计算机学院**

**课**

**程**

**实**

**验**

**报**

**告**

|  |  |
| --- | --- |
| **实验课程：** | **计算机图形学** |
| **实验项目：** | 二维、三维图形变换 |
| **指导教师：** |  |
| **学生姓名:** |  |
| **学生学号：** |  |
| **班 级：** |  |
| **实验地点：** |  |
| **实验时间：** |  |
| **实验成绩：** |  |

**一【上机实验内容】**

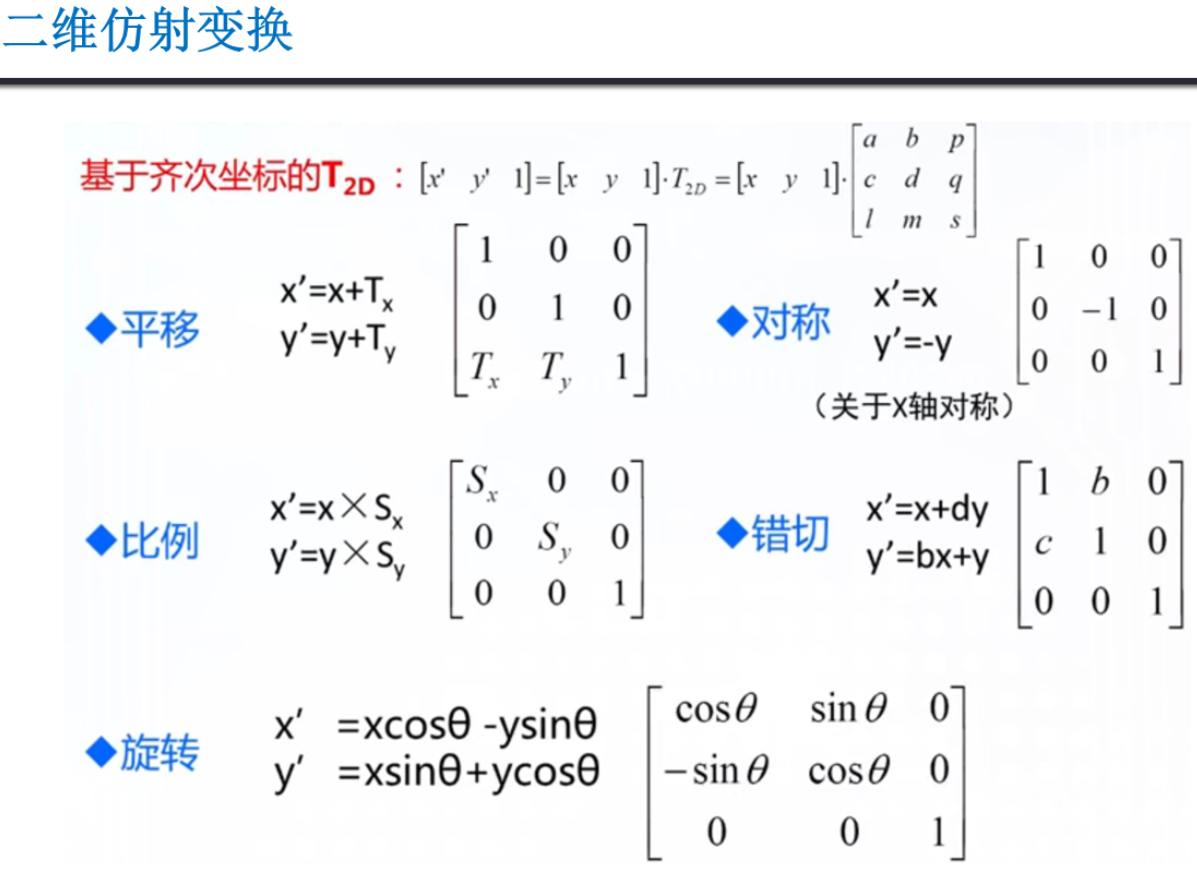
**课内实验：（100分 ）**

1. 实现三角形的平移。

2. 实现三角形的旋转。

3. 实现三角形的缩放。

**二【二维图形基本几何变换矩阵】（重点）(20分)**

请写出二维图形平移变换矩阵、比例变换矩阵和缩放变换矩阵的一般形式。

**三【上机调试中出现的错误信息、错误原因及解决办法】(10分)**

（记录下你调试程序中出现的错误信息的英文提示，分析出错原因及解决办法）

错误信息：

error LNK2019: 无法解析的外部符号 "public: \_\_cdecl GMatrix::GMatrix(int,int)" (??0GMatrix@@QEAA@HH@Z)，函数 "void \_\_cdecl Trans(class QImage \*,class GPoint,class GPoint,class GPoint,unsigned int)" (?Trans@@YAXPEAVQImage@@VGPoint@@11I@Z) 中引用了该符号

出错原因：

未确认，根据网络搜索，排除了如下原因：

定义了这个函数却没有去实现它，

引入了别的工程的文件，却没有将文件添加到这个项目中，导致调用这个函数时编译器找不到这个函数的实现，

头文件中的函数定义在编译为 C 程序的文件中，而头文件是在 C++ 文件中不带 extern “C” 修饰符声明的。在此情况下，需要添加extern "C"修饰符。，

VS缺少ws2\_32.lib这个库

lib文件用的是64位的，而dll文件是32位的

解决方法：

暂无

**四【上机实验中的其他它问题及心得】（重点）(10分)**

根据网上资料学习了bresenham算法，下面简单介绍一下

在DDA的学习中了解在光栅上画直线，是在y=kx+b上的直线附近选靠近的点，具体做法是不断自增x，然后根据解析式计算出相应的y，四舍五入选点，但是这种算法有浮点数运算，较慢，所以出现了bresenham来更快的解决这个问题，首先只考虑0<=k<=1的情况，那么x不断自增，y只会变成y或y+1，这里设dx=x1-x0,dy=y1-y0,p=f(x+1)-y-[y+1-f(x+1)]，如果p>0,说明更靠近y+1，反之靠近y，但是用解析式计算f(x+1)有浮点数k，所以设u{i}=p\*dx=2\*dy\*x-2\*dx\*y+c，c是常数和，递推出u{i+1}=u{i}+2\*dy-2\*dx\*(f(x+1)-y))，对u{i}用y=kx+b，k=dy/dx，以及把c展开，得到u{0}=2\*dy-dx，然后就可以递推得出u{i}，由于0<=k<=1，所以dx>0，所以u和p符号相同，所以u的正负性决定了y会变成y或y+1，0<=k<=1的情况就解决了，整个坐标系可以被分为8个区域，米字形分布，其他区域的情况和上面那个情况类似，所以只需要考虑自增或自减的是x还是y

**五【源程序、测试数据及结果】**

1. 实现三角形的平移

程序**(10分)**

void Trans(QImage\* image, GPoint apoint, GPoint bpoint, GPoint cpoint, QRgb value)

{

GMatrix ThreeThreeMatrix(3, 3), ap(1, 3), bp(1, 3), cp(1, 3);

GMatrix moveap(1, 3), movebp(1, 3), movecp(1, 3);

ThreeThreeMatrix.SetTranslationThree();

ap.SetoneThree(apoint.getX(), apoint.getY());

bp.SetoneThree(bpoint.getX(), bpoint.getY());

cp.SetoneThree(cpoint.getX(), cpoint.getY());

moveap = ap \* ThreeThreeMatrix;

movebp = bp \* ThreeThreeMatrix;

movecp = cp \* ThreeThreeMatrix;

GPoint Apoint(moveap.GetoneThreea(), moveap.GetoneThreeb());

GPoint Bpoint(movebp.GetoneThreea(), movebp.GetoneThreeb());

GPoint Cpoint(movecp.GetoneThreea(), movecp.GetoneThreeb());

BresenhamF(image, Apoint.x, Apoint.y, Bpoint.x, Bpoint.y, value);

BresenhamF(image, Cpoint.x, Cpoint.y, Bpoint.x, Bpoint.y, value);

BresenhamF(image, Apoint.x, Apoint.y, Cpoint.x, Cpoint.y, value);

}

结果图**(10分)**



1. 实现三角形的旋转。

程序**(10分)**

void rotate(QImage\* image, GPoint apoint, GPoint bpoint, GPoint cpoint, QRgb value) {

GMatrix ThreeThreeMatrix(3, 3), ap(1, 3), bp(1, 3), cp(1, 3);

GMatrix moveap(1, 3), movebp(1, 3), movecp(1, 3);

ThreeThreeMatrix.SetRotateThree();

ap.SetoneThree(apoint.getX(), apoint.getY());

bp.SetoneThree(bpoint.getX(), bpoint.getY());

cp.SetoneThree(cpoint.getX(), cpoint.getY());

moveap = ap \* ThreeThreeMatrix;

movebp = bp \* ThreeThreeMatrix;

movecp = cp \* ThreeThreeMatrix;

GPoint Apoint(moveap.GetoneThreea(), moveap.GetoneThreeb());

GPoint Bpoint(movebp.GetoneThreea(), movebp.GetoneThreeb());

GPoint Cpoint(movecp.GetoneThreea(), movecp.GetoneThreeb());

BresenhamF(image, Apoint.x, Apoint.y, Bpoint.x, Bpoint.y, value);

BresenhamF(image, Cpoint.x, Cpoint.y, Bpoint.x, Bpoint.y, value);

BresenhamF(image, Apoint.x, Apoint.y, Cpoint.x, Cpoint.y, value);

}

结果图**(10分)**



3、实现三角形的缩放。

程序**(10分)**

void scale(QImage\*image, GPoint apoint, GPoint bpoint, GPoint cpoint, QRgb value) {

GMatrix ThreeThreeMatrix(3, 3), ap(1, 3), bp(1, 3), cp(1, 3);

GMatrix moveap(1, 3), movebp(1, 3), movecp(1, 3);

ThreeThreeMatrix.SetScaleThree();

ap.SetoneThree(apoint.getX(), apoint.getY());

bp.SetoneThree(bpoint.getX(), bpoint.getY());

cp.SetoneThree(cpoint.getX(), cpoint.getY());

moveap = ap \* ThreeThreeMatrix;

movebp = bp \* ThreeThreeMatrix;

movecp = cp \* ThreeThreeMatrix;

GPoint Apoint(moveap.GetoneThreea(), moveap.GetoneThreeb());

GPoint Bpoint(movebp.GetoneThreea(), movebp.GetoneThreeb());

GPoint Cpoint(movecp.GetoneThreea(), movecp.GetoneThreeb());

BresenhamF(image, Apoint.x, Apoint.y, Bpoint.x, Bpoint.y, value);

BresenhamF(image, Cpoint.x, Cpoint.y, Bpoint.x, Bpoint.y, value);

BresenhamF(image, Apoint.x, Apoint.y, Cpoint.x, Cpoint.y, value);

}

结果图**(10分)**

