《基于 Andropid 源生 AR 应用开发》

实验手册 03

版 本 1.0

文档提供：智能设备教研室 丁盟

目录

第3章　坐标映射

3.1 实验目的

目的一： 掌握坐标映射的原理。

目的二： 掌握摄像视图的调整方法。

目的三： 掌握透视投影的使用方法。

3.2 准备工作

准备一： 在 Android Studio 中根据实验手册 2 创建一个简单的 OpenGL ES 应用程序，本次实验所有操作均是在实验手册 2 的实验成果上进行。

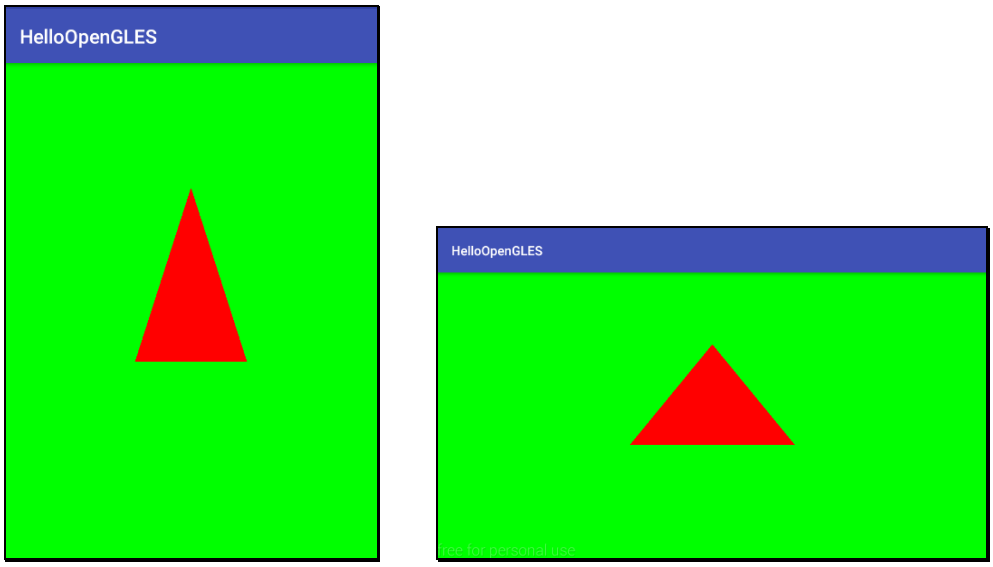

图 3.2.1

实验手册 2 中的实验结果仅仅是绘制一个三角形，但是随着手机横纵屏的切换致使显示画面的宽高发生变化，致使绘制出来的三角形发生了失真，本次实验的目的就是通过坐标映射来解决绘制图形失真的问题。

准备二： 获取投影矩阵函数说明

```
Matrix.frustumM(mProjMatrix, 0,
                left, right,
                bottom, top,
                near, far)
```

- float left, //near 面的 left
- float right, //near 面的 right
- float bottom, //near 面的 bottom
- float top, //near 面的 top
- float near, //near 面距离
- float far //far 面距离

设置这些参数能起到的作用：先是 left，right 和 bottom,top，这 4 个参数会影响图像左右和上下缩放比，所以往往会设置的值分别-(float) width / height 和(float) width / height，top 和 bottom 和 top 会影响上下缩放比，如果 left 和 right 已经设置好缩放，则 bottom 只需要设置为-1，top 设置为 1，这样就能保持图像不变形。也可以将 left，right 与 bottom，top 交换比例，即 bottom 和 top 设置为 -height/width 和 height/width, left 和 right 设置为-1 和 1。

near 和 far 参数稍抽象一点，就是一个立方体的前面和后面，near 和 far 需要结合拍摄相机即观察者眼睛的位置来设置，例如 setLookAtM 中设置 cx = 0, cy = 0, cz = 10，near 设置的范围需要是小于 10 才可以看得到绘制的图像，如果大于 10，图像就会处于了观察这眼睛的后面，这样绘制的图像就会消失在镜头前，far 参数，far 参数影响的是立体图形的背面，far 一定比 near 大，一般会设置得比较大，如果设置的比较小，一旦 3D 图形尺寸很大，这时候由于 far 太小，这个投影矩阵没法容纳图形全部的背面，这样 3D 图形的背面会有部分隐藏掉的。

准备三： 获取视口矩阵函数说明

```
Matrix.setLookAtM(float[] rm, int rmOffset,
           float eyeX,    float eyeY,    float eyeZ,
           float centerX, float centerY, float centerZ,
           float upX,     float upY,     float upZ)
```

- float eyeX //摄像机位置 x
- float eyeY //摄像机位置 y
- float eyeZ //摄像机位置 z
- float centerX //摄像机目标点 x
- float centerY //摄像机目标点 y
- float centerZ //摄像机目标点 z
- float upX //摄像机 UP 向量 X 分量
- float upY //摄像机 UP 向量 Y 分量
- float upZ //摄像机 UP 向量 Z 分量

这个方法看起来很抽象，设几组参数对比一下效果，摄像机目标点，即绘制的 3D 图像，tx，ty，tz,为图像的中心位置设置到原点即 tx = 0,ty = 0，tz = 0；摄像机的位置，即观察者眼睛的位置 我们设置在目标点的正前方（位置 z 轴正方向），cx = 0，cy = 0，cz = 10；接着是摄像机顶部的方向了，如下图，很显然相机旋转，up 的方向就会改变，这样就会会影响到绘制图像的角度。

图 3.2.2

例如设置 up 方向为 y 轴正方向，upx = 0,upy = 1,upz = 0。这是相机正对着目标图像，则绘制出来的效果如下图所示

图 3.2.3

如果设置 up 方向为 x 轴正方向，upx = 1,upy = 0,upz = 0，绘制的图像就会如下图所示

图 3.2.4

显然如果设置在 z 轴方向，图像就会看不见。Matrix.setLookAtM,的作用大体就是这样。

准备四： 矩阵相乘函数说明

```
Matrix.multiplyMM(float[] result, int resultOffset,
                  float[] lhs,    int lhsOffset,
                  float[] rhs,    int rhsOffset)
```

- float[] result　　// 存放结果的变换矩阵
- int resultOffset　// 结果矩阵偏移量，一般为 0
- float[] lhs　　// 参与运算的左矩阵
- int lhsOffset　　// 左矩阵偏移量，一般为 0
- float[] rhs　　// 参与运算的右矩阵
- int rhsOffset　　// 右矩阵偏移量，一般为 0

3.3 实验步骤

步骤一 在 HelloOpenGLES20Renderer 类中添加坐标转换相关的属性成员：

```
// 顶点着色器中坐标转换时所使用到的转换矩阵的索引
private int muMVPMatrixHandle;
```

```
// 坐标转换总矩阵，包含视口转换与投影转换
private float[] mMVPMatrix = new float[16];
// 视口矩阵
private float[] mVMatrix = new float[16];
// 投影矩阵
private float[] mProjMatrix = new float[16];
```

步骤二 修改顶点着色器 vertexShaderCode 的代码，将坐标转换矩阵应用到顶点坐标的转换中，用来实现坐标转换的功能。

```
// 顶点着色器代码
private final String vertexShaderCode =
        "uniform mat4 uMVPMatrix;   \n" +
        "attribute vec4 vPosition;  \n" +
        "void main() {              \n" +
        "   gl_Position = uMVPMatrix * vPosition;   \n" +
        "}                          \n";
```

在顶点着色器源码中添加了 uniform mat4 uMVPMatrix 全局变量，uniform 表示变量用来进行接受数据，mat4 表示为 4*4 的 float 类型的矩阵，通过此矩阵与顶点数据进行运算来达到坐标转换的功能。

步骤三 在 onSurfaceChanged()方法中依次添加**获取顶点着色器中转换矩阵索引、获取视口矩阵、获取投影矩阵**的代码。

```
// 获取顶点着色器中坐标转换矩阵的索引
muMVPMatrixHandle =
GLES20.glGetUniformLocation(mProgram,
        "uMVPMatrix");

// 获取投影矩阵
float ratio = (float) width / height;
Matrix.frustumM(mProjMatrix, 0,
        -ratio, ratio, -1, 1, 3, 7);
```

```
// 获取视口矩阵
Matrix.setLookAtM(mVMatrix, 0,
        0f, 0f, -3f,
        0f, 0f, 0f,
        0f, 1.0f, 0.0f);
```

步骤四 在 onDrawFrame()方法中计算坐标转换总矩阵，并将其传递给顶点着色器的坐标转换矩阵（注意要将如下代码放在 glDrawArrays()绘制函数之前）。

```
// 通过投影矩阵和视口矩阵，获取坐标转换总矩阵
Matrix.multiplyMM(mMVPMatrix, 0, mProjMatrix, 0,
mVMatrix, 0);

// 将坐标转换总矩阵传递给顶点着色器的坐标转换矩阵
GLES20.glUniformMatrix4fv(muMVPMatrixHandle, 1, false,
        mMVPMatrix, 0);
```

3.4 实验结论

当编码工作完成后在模拟器或真机中运行项目，当再次发生横纵屏切换时，图像不再会因为屏幕的变化而发生失真情况。效果如下：

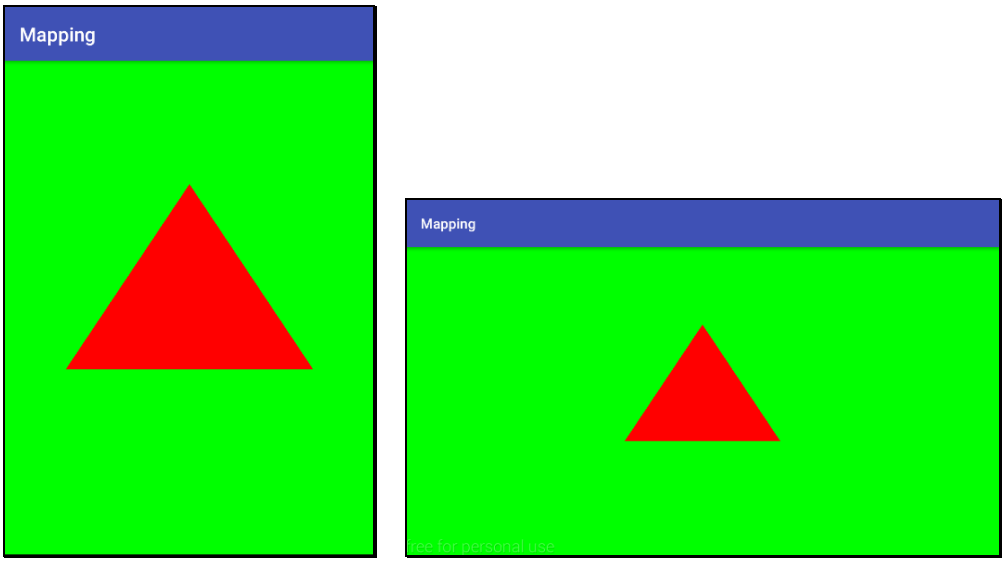

图 3.4.1