九：剔除

在屏幕空间上绘制三角形的时候，实际上有些三角形是没必要绘制的。

1. 面积为 0 的三角形剔除

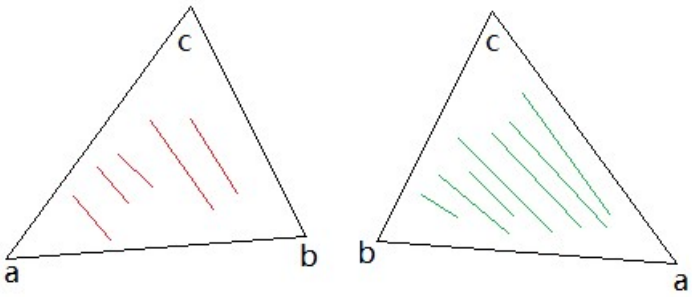
如果一个三角形在屏幕空间上的面积为 0，则该三角形不会对任何像素产生作用，可以不绘制。实际上我们的 FillTriangle 函数一直都剔除掉面积为 0 的三角形。

```
Vector3 ab(ps[1].X - ps[0].X, ps[1].Y - ps[0].Y, 0.0);//ps[0]->ps[1]
Vector3 bc(ps[2].X - ps[1].X, ps[2].Y - ps[1].Y, 0.0);//ps[1]->ps[2]
Vector3 ca(ps[0].X - ps[2].X, ps[0].Y - ps[2].Y, 0.0);//ps[2]->ps[0]
Vector3 ac(ps[2].X - ps[0].X, ps[2].Y - ps[0].Y, 0.0);//ps[2]->ps[0]
double square = ab.X * ac.Y - ab.Y * ac.X;//得到三角形有向面积的2倍
if (square == 0)//三角形面积为0则不进行后面的绘制处理
{
    return;
}
```

我们在绘制一个三角形的时候已经判断过面积了，如果三角形在屏幕空间上的面积为 0，则放弃绘制该三角形，根据第四章对有向面积的介绍，实际上这段代码中的 Square 是有向面积的 2 倍，但是不影响我们判断面积是否为 0。

2. 背面的剔除

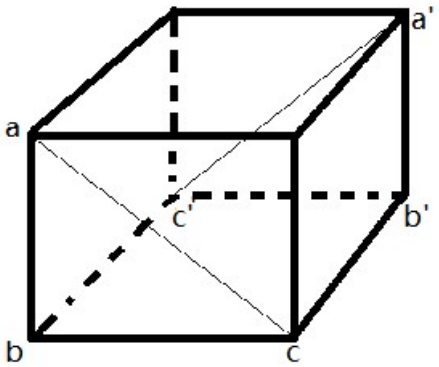
一个面积不为 0 的三角形在屏幕空间上必定是保持顺时针或者逆时针的顺序的，如下图：



左为逆时针、右图为顺时针

我们可以选择性的绘制逆时针三角形而放弃绘制顺时针三角形，又或者选择性的绘制顺时针三角形而放弃绘制逆时针三角形，为什么呢？

假设上对于上图中我们选择了绘制逆时针三角形，则表示我们认为红色面是正面，而右边这种情况则是将左边经过一定的旋转之后，变成了绿色面也就是背面朝向屏幕。我们可以完全放弃背面的绘制。因为假设一个立方体的外面我们全部定义为逆时针三角形，如下图：



我们暂时只考虑Δabc，如果该立方体旋转 180°，则Δabc 会变成上图中的Δa'b'c'，由逆时针变成了顺时针，或者说变成了背面朝向屏幕。对于一个封闭的几何体的某个面来说，一旦变成了背面面向屏幕，则该面必然会被遮挡，所以我们可以选择放弃绘制背面，这样可以节省很多计算资源。

根据我们在第四章对有向面积的介绍，我们可以用有向面积的符号作为顺逆时针的判断。

所以我们只需要对判断三角形面积为 0 的这行代码稍作修改即可：

```
if (square >= 0)//三角形面积大于等于0则不进行后面的绘制处理
{
    return;
}
```

这里我们将有向面积大于0的三角形也进行了剔除，这样将不绘制逆时针三角形。同上square是有向面积的两倍，但是不影响符号判断。并且如果要绘制逆时针三角形而剔除顺时针三角形的话，改变square的判断条件即可。

在opengl中，绘制顺时针和逆时针作为一个开关选项，使用GL_CCW或者GL_CW可以控制程序绘制逆时针还是顺时针，其中CCW是CounterClockWise的缩写，CW是ClockWise的缩写。使用glFrontFace函数来设置到底是顺时针还是逆时针三角形为正面。