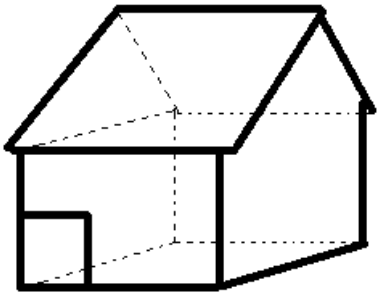
八：消隐

1. 消隐简介

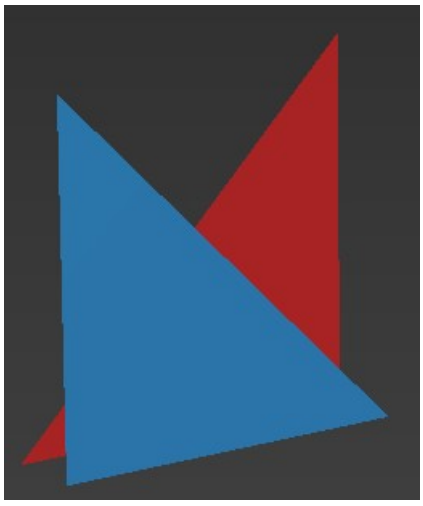
前面几章我们介绍了三维三角形的绘制，但是有一个问题就是不能正确的处理遮挡，类似于下面这个房子：



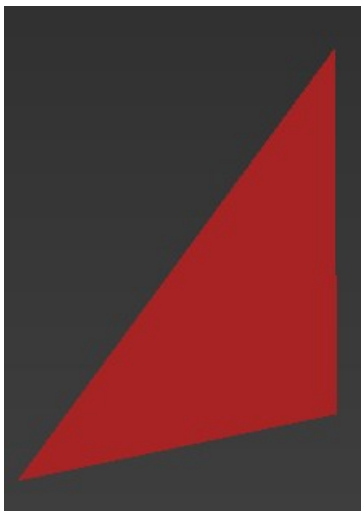
我们从前面斜一点看过去，是只能够看到实线围成的这些面，而虚线这些面虽然是真实的存在，但是却被实线面遮挡住了而看不见，所以我们在绘制多边形的时候也需要考虑这个问题。

2. 画家算法

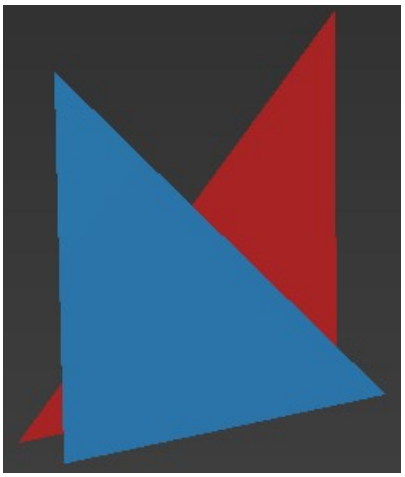
画家算法处理消隐的方法很简单，类似于画家在绘制油画那样在画布上叠加图案，我们可以先将远处的多边形绘制到屏幕上面，然后再绘制离 near 平面近的三角形。例如下面的两个三角形：



在使用画家算法的时候，先绘制较远的那个红色三角形到屏幕

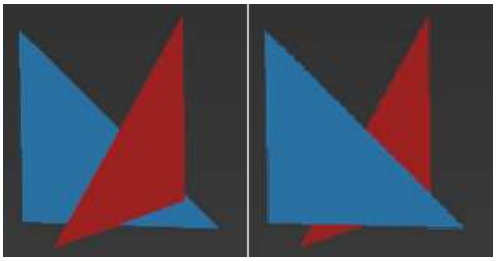


然后再在屏幕上面叠加绘制蓝色三角形



这样就能处理这种情况的遮挡了，但是当三角形存在相交的时候画家算法就不能正确处理了，例如下面这种情况：

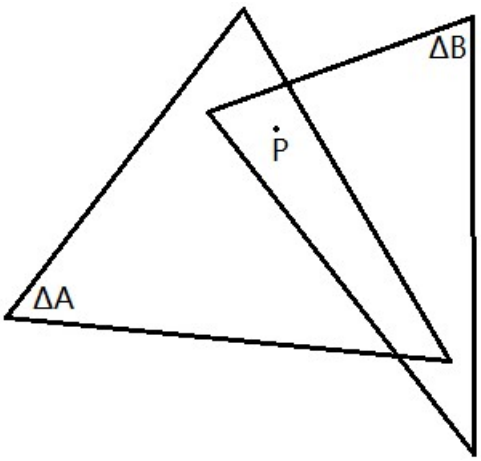
这种情况如果使用画家算法则无法正确绘制，首先就是哪个三角形距离更远不好判断，其次就是不管先绘制哪个三角形，在三角形投影到屏幕上面之后的叠加部分都会出现问题，如下：



左边先绘制蓝色三角形，右边先绘制红色三角形

3. Z-Buffer 算法

Z-Buffer 算法则是为屏幕上面的每个像素创建一个缓冲器，绘制三角形 A 时记录像素当前被绘制时的深度$Z_A$，这个保存的地方叫做 Z 缓冲区。在绘制另一个三角形 B 时，如果需要在同一像素绘制，则计算该像素在三角形 B 所对应的深度值$Z_B$，如果$Z_B > Z_A$，则放弃绘制，否则绘制当前像素。如下图：




假设我们绘制ΔA 时，针对屏幕上的像素 P，我们记录$Z_A$的值，这时候$Z_p = Z_A$，在会在ΔB 时，我们会计算出一个$Z_B$，使用$Z_B$和$Z_P$作对比，我们就可以决定对于像素 P 到底是绘制ΔB 还是保留ΔA 对应的图案。

那么我们现在的问题就变成了在绘制一个三角形的某个像素是怎么选择该像素对应的深度值?还记得我们在第六章介绍裁剪空间和 CVV 时曾经说过，我们在计算裁剪空间中一个顶点的 z 让其在深度值上面单调递增，使用裁剪空间中的 z 值(伪深度)可以作为 Z-Buffer 算法中的深度值使用，因为该值关于相机坐标系中原始顶点中的 Z 值单调递增。实际上我们也可以使用裁剪空间中的ω值作为深度值，因为ω分量就是原始顶点的 z 值，也是关于原始顶点单调递增。类似于 opengl，我们选择了裁剪空间中的 z 分量作为深度值。

相关代码见 chapter8/Z-Buffer/main.cpp

我们在 main 函数中创建一个 double 数组，数组的长度等于屏幕宽度×屏幕高度，这样就可以为每个像素留一个 double 类型的 Z-Buffer。并且将每个像素的 Z-Buffer 都设置为 1，这样任意一个裁剪之后剩余的多边形的深度值都会小于 1，都能够被绘制到屏幕上。

```
Z_Buffer = new double[640*480];//开辟一个数字，为每个像素留一个 double 值作为 Z-buffer
std::fill(Z_Buffer, Z_Buffer+(640 * 480), 1);//预先将所有像素的 Z-Buffer 填充为最远 1
```

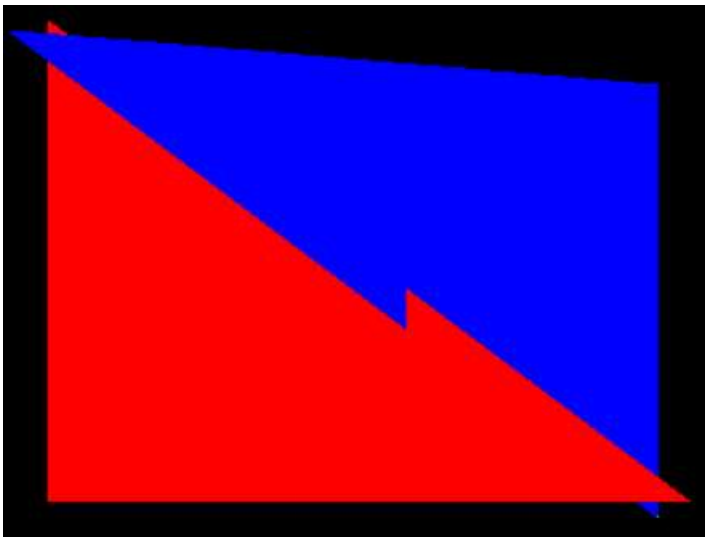
同时我们将 ClipTriangleAndDraw 函数传递给 FillTriangle 函数的顶点中加上 Z 值。最后在屏幕上面渲染的时候，我们加上深度测试(depth test):

```
double Pomega = 1 / ((1 / ps[0].W) * WeightA + (1 / ps[1].W) * WeightB + (1 / ps[2].W)
* WeightC);//求出当前顶点的ω分量

z = Pomega * (ps[0].Z / ps[0].W * WeightA + ps[1].Z / ps[1].W * WeightB + ps[2].Z /
ps[2].W * WeightC);//使用线性插值计算当前绘制像素的Z值
if (z > Z_Buffer[y*640+x])
{
    continue;//跳过当前像素的绘制
}
else
{
    Z_Buffer[y * 640 + x] = z;
}
```

如果当前像素计算出来的 z 值大于该像素的 Z-Buffer，则放弃渲染。这样就可以得到正确的遮挡处理

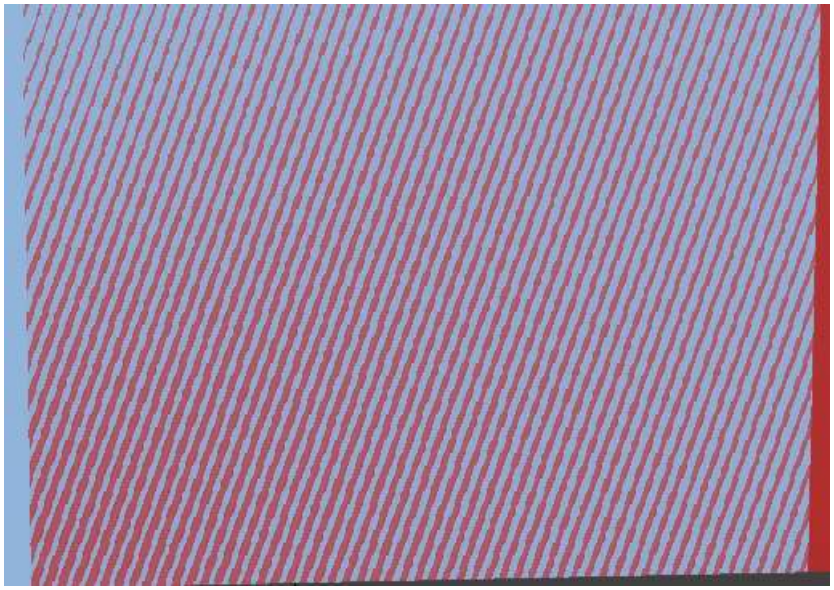
效果：



即使三角形存在相交，也能处理相交部分。

4. Depth-fighting

深度冲突是指我们在上面做深度测试的时候出现无法准确对一个像素的深度作出大小对比，类型下图(只截取了一部分):



在绘制两个片面的时候，如果两个片面很接近，当同一个像素的深度距离小于一个我们使用的数据分辨率时，将无法判断到底是哪个深度更小，应该保留哪一个哪一个片面的渲染数据，这时候就会出现这种情况。假如当前像素的深度为 0.000000000202，而下一个片面在当前像素的深度值为 0.000000000201，理论上后一次应该不会被绘制的，假如我们用的 Z-Buffer 的数据分辨率只能分辨到 0.00000000020，那么这两个深度都会被当初 0.00000000020，而当我们使用 `z > Z_Buffer` 作为放弃条件的话，这个像素在后一次也会被绘制。而前面我们使用 double 作为 Z-Buffer 的数据类型，double 也不能表示无限精度，自然也会出现上述问题。Depth-fighting 和深度缓冲区的数据类型、绘制顺序、测试条件相关。并且如果我们使用裁剪空间中的 Z 值作为深度测试条件，那么离 near 平面越远的点越容易出现深度冲突，因为我们是使用下面的公式计算深度值的(其中 Z'表示在 CVV 中的深度值，Z 表示原始空间中的深度值):

$$z' = \frac{f+n}{f-n} + \frac{2fn}{(n-f)z}$$

可以发现，深度值是和 Z 成反比例函数，那么在 Z 值越大的时候，深度差距反而越小，就越容易出现深度冲突。有时候可以适当的减小 far 平面的范围来避免深度冲突。