如何在GPU中极为快速的渲染256x SSAA抗锯齿画面质量的游戏

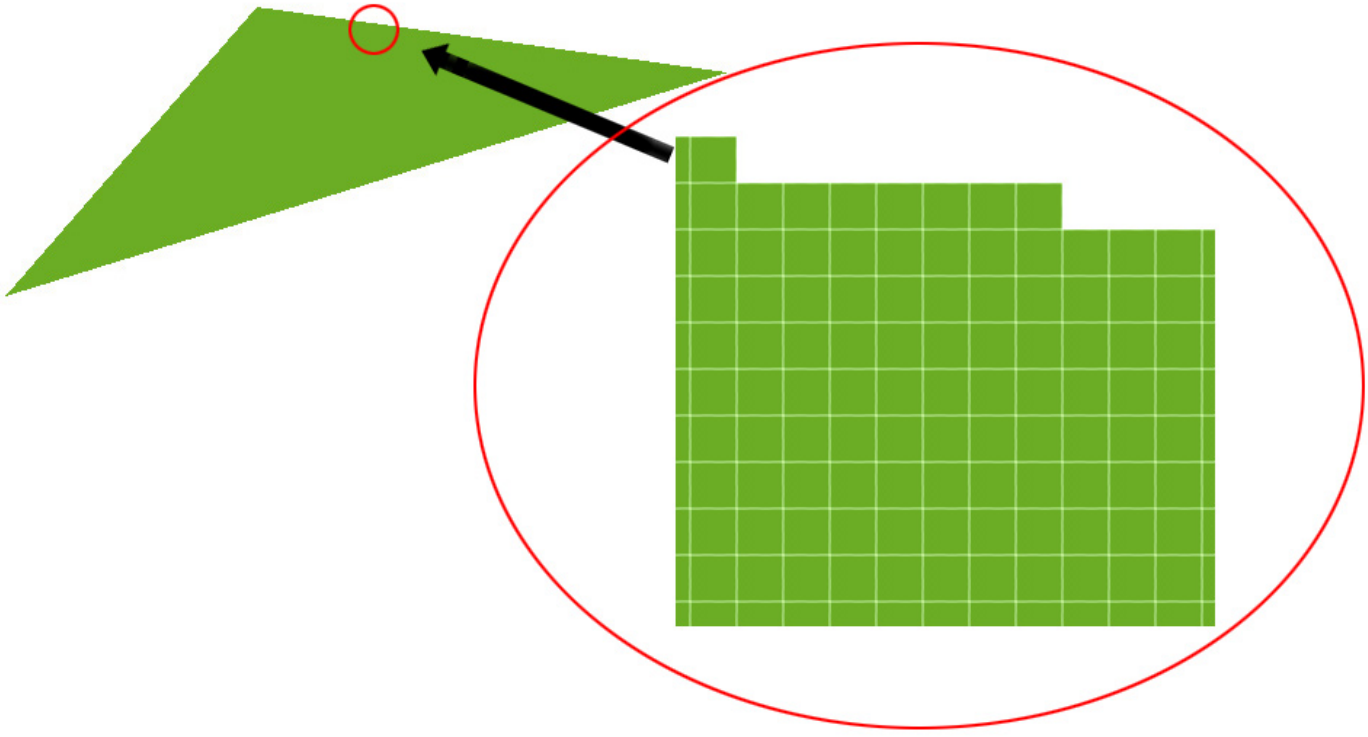
- sulin huang.

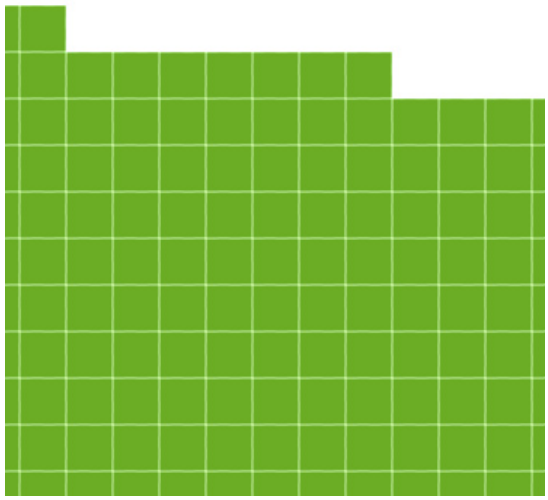
我们都知道游戏画面是由很多个三角形组合而成的，现在让我们深入到最底层的三角形渲染里面并极快的提升抗锯齿的渲染速度以让它达到甚至超过256x SSAA的质量（如果显示器的色深支持）

传统的渲染方式的如果使用256x SSAA画面需要对屏幕的每一个点做256次的取样并平均它们获得最终的像素颜色，这是极为消耗显卡资源的。

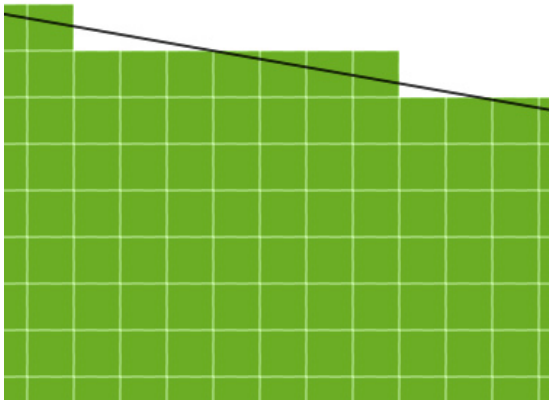
让我们来极大的提升它。

我们把图像放大到一个像素一个像素。

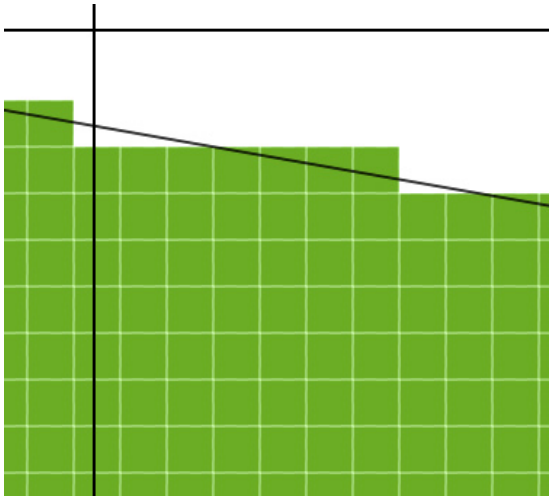




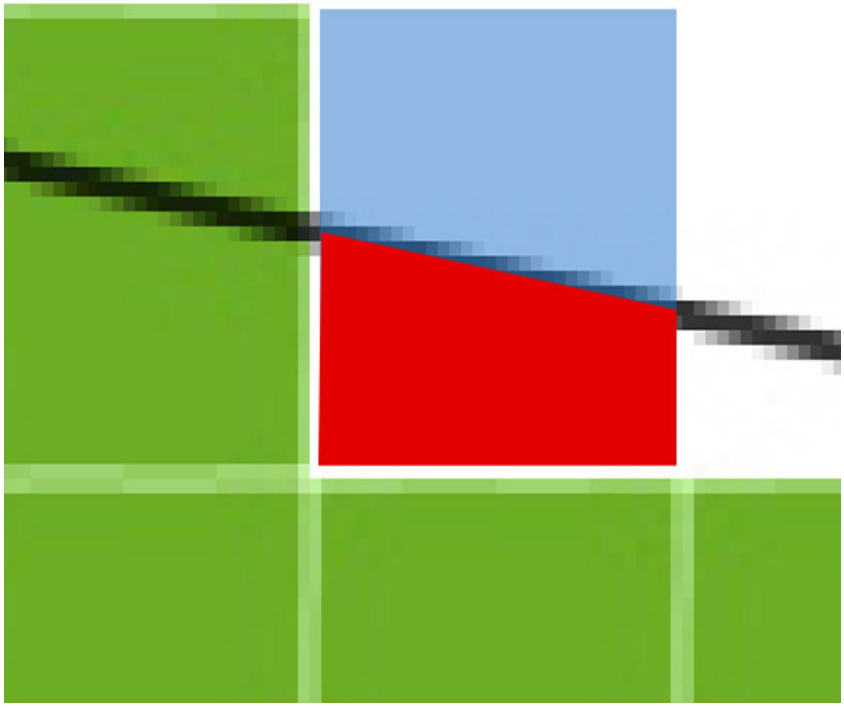
图中黑色的线是数学上三角形此条边经过的像素位置（浮点，非整数）



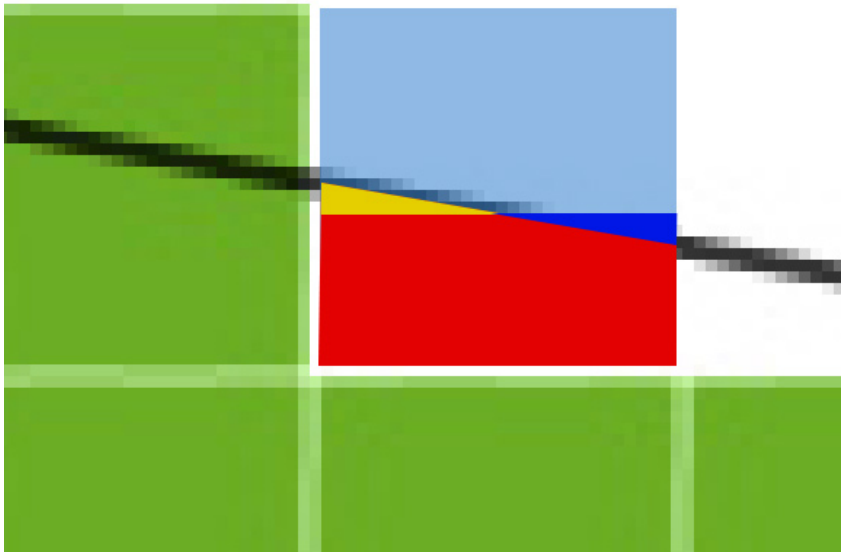
让我们给它画一个坐标系.



实际上我们应该去计算红色区域的面积。我们计算出红色区域占整个像素正方形的比例即可。

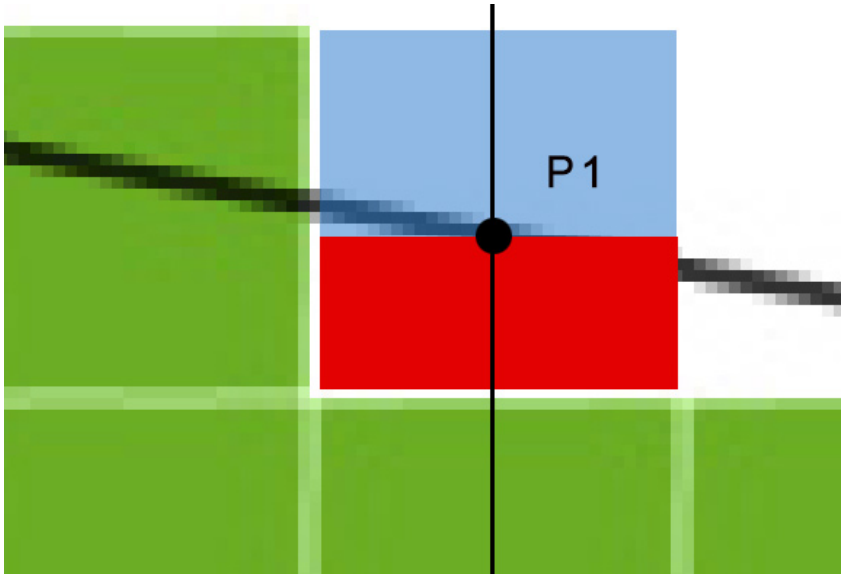


数学上很容易可以严格证明黄色三角形区域和蓝色三角形区域面积是相等的。



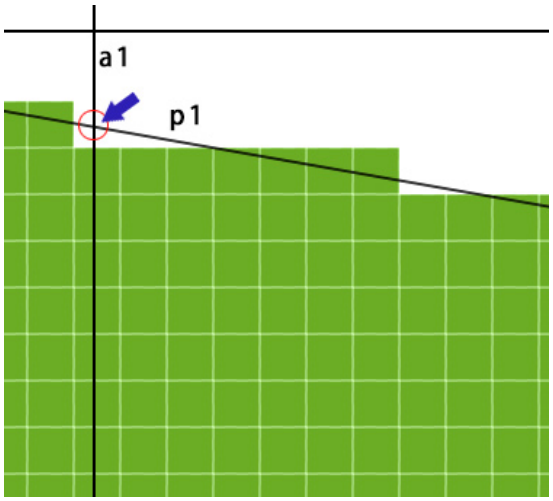
那么，实际上我们计算这样的区域即可。

因为红色长方形和淡蓝色正方形像素块有一条边是等长的，那实际上这个比例就是长方形高度和1px的比例，也就是p1点。



每个三角形的3个顶点是已知的，这意味这这个三角形的每一条边对我们是已知的。

通过简单的线段的交点计算。我们很容易计算出这条斜线在这个x位置它的y的具体值p1。必须计算浮点。

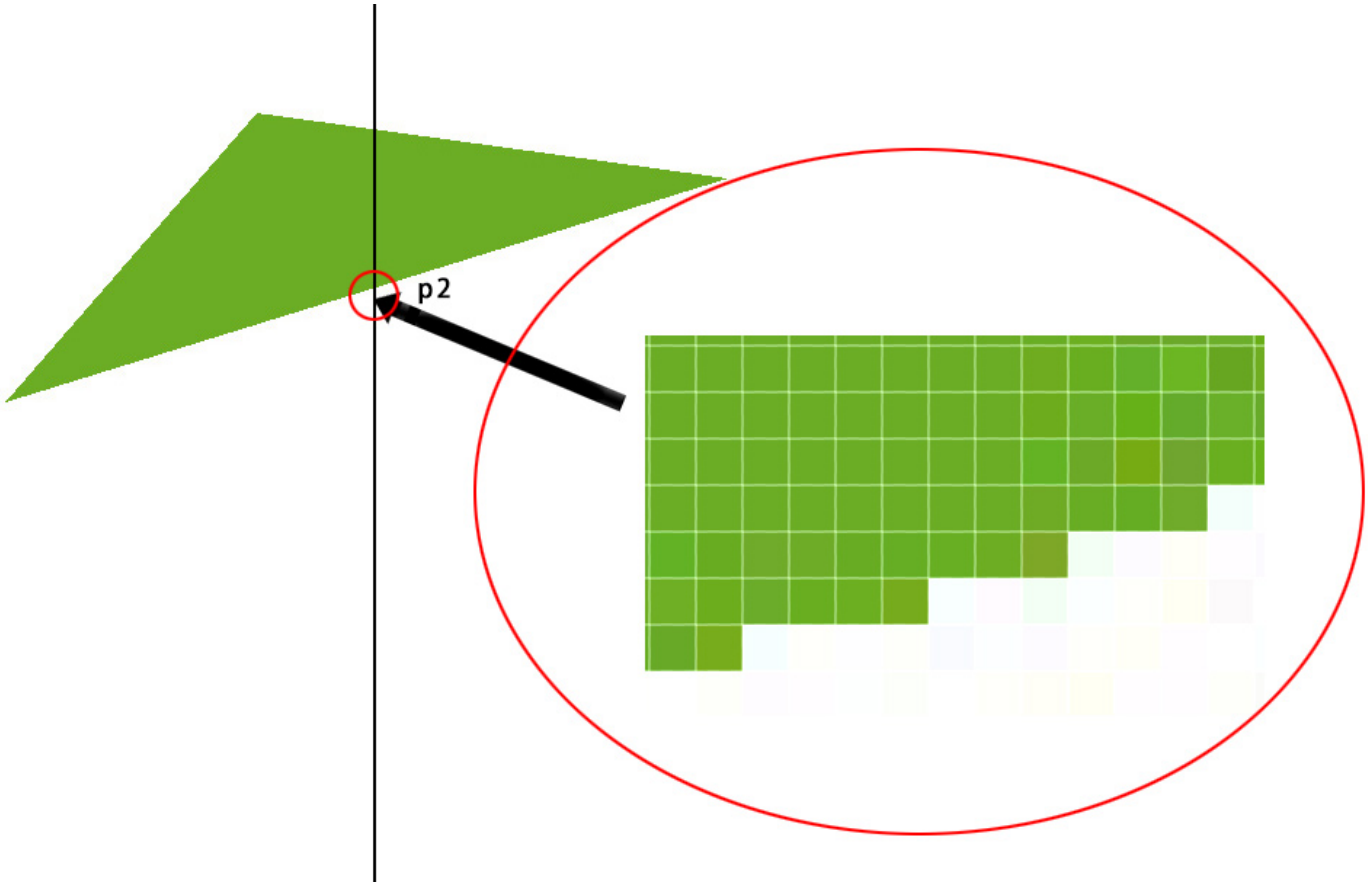


此时我们得到一个数学上的浮点值，假设为5.643注意它的小数部分，这个就是我们需要的数据。实际上(1-0.643)=0.357就是我们想要的比例值。

现在我们来按照0-255的层级来量化它（相对当前显示器的色深而言）

此时，我们建立一个Alpha通道来存储它，把它存储为和图像对应的二维数据（稍后在真正渲染的时候我们需要这个数据来计算)

同样，在这条竖线的底部也同样有一条三角形边，我们可以用同样的方式计算出它的相交之处的浮点值。

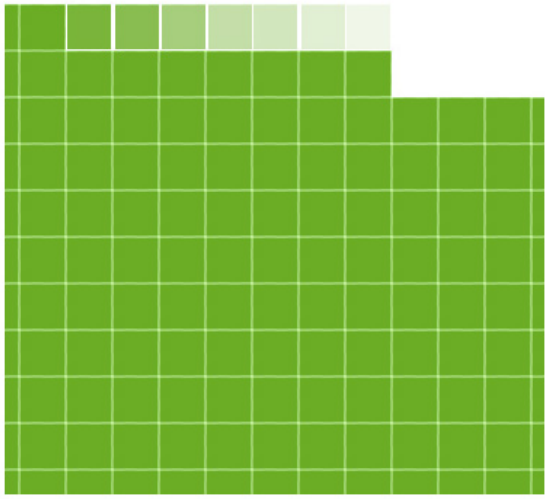


此时，我们就得到了这个竖向的所有像素的Alpha通道值。

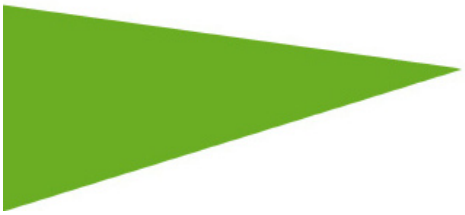
在p1点之上，所有的像素点alpha值为0，不需要渲染。在p1点和p2点直接所有的像素点apha值为255，需要渲染。然后我们得到了两个边缘的像素Alpha值。（假设为0.357和0.85），把他们映射为当前显示器支持的bit存储，假设为0-255

现在对于一个边，我们的计算值假设为 0.357 0.215 .0143 0.05 0.951 0.751 0.65 0.54 0.45

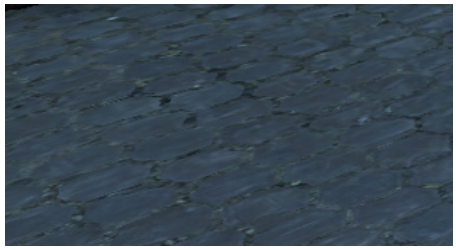
我们就能得到一个阶梯式的Alpha值。



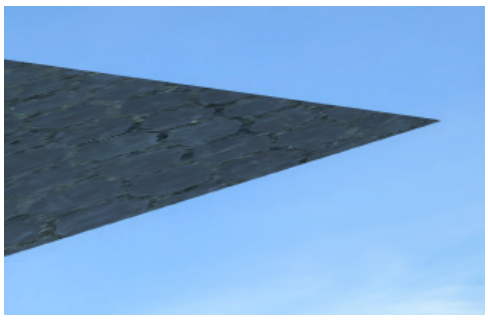
对于三角形的顶点像素，我们可以使用传统的采样多个点来计算它的透明度，（因为像素数量很少，这个计算量很低）
我们缩小来看。



现在，每一个像素我们有一个Alpha值。使用这个Apha值可以很容易的计算当前三角形最终颜色和它后面的颜色在边缘具体每个像素应该按照什么比例混合。



混合它们，最终：



对于所有的三角形而言都是一样的：

至此，完成。

感谢您的阅读。