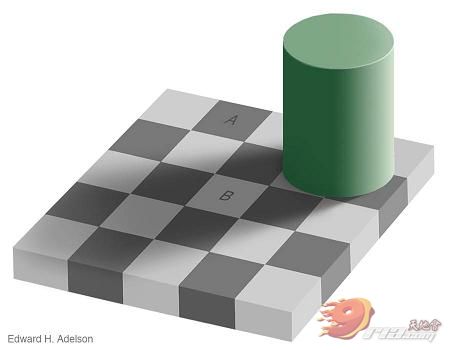
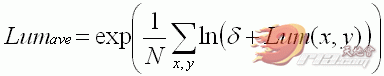
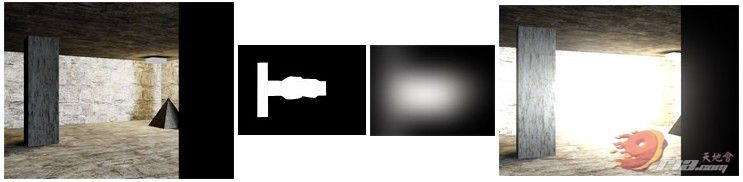
　　　　　　　　　　　　　　ｓ

大家好久不见，欢迎回到久违的“flash 3d图形学科普教室”，我是主讲人白蓝紫……以上扯淡。  
  
精力有限，最近没怎么写帖子和日志，不过时隔多日后，现在为您奉上这篇关于高动态范围渲染的文章，希望您能喜欢并轻松愉快地阅读。  
  
什么？你只是看到这鸟语标题才戳进来的？并且完全不知道我在说些什么？  
唉呀，对于以废话多而闻名的我以及HDR的发明人而言还真是失礼啊，那这次就先让各位看到一个实例，以便获得些直观地感受吧。  
<http://www.vertex-ixd.com/HDR/>  
这是使用使用flare3d制作的flash 3d HDR范例，这次我可没用openGL或者DX的产物糊弄人啊，这可真是swf格式的。  
不过，这个demo不止用到了HDR和ToneMapping技术（但作者说受限没用，天知道他用什么替代的Mapping），还用到了环境反射来模拟水晶的镜面效果。  
最后补充一句，虽然用的是flare3d的范例，但我可没收他们的好处费啊，真的真的！  
  
那么开始进入正题吧。  
  
虽然那个demo很炫，不过各位知道我想让你们注意的是什么吗？  
不知道吗？不知道吧！  
好，先来说说HDR和ToneMapping是什么吧。  
  
HDR=High Dynamic Range=高动态范围  
ToneMapping=色调映射  
简单来说，HDR的目的就是模拟相机的曝光，而ToneMapping则是实现的一种手段。  
于是，具体来说，就是上面那个demo里的高光光晕和对着“太阳”时的高亮处理。（三棱镜彩虹什么不在本篇讨论范围内）  
  
  
为什么要模拟曝光？先来看下对比吧！  
  
很明显，左边使用了HDC的图像要比右边没使用的更逼真，更细腻，也更立体。  
  
为什么会这样呢？  
我们知道，电脑屏幕上的一个像素是由RGB三通道组成的，并且单通道的范围通常表示为0-255。也就是说，从黑（0x000000，都为0）到白（0xFFFFFF，都为255）的过渡只有256阶。  
于是问题来了。假设桌面的亮度为1，那么太阳按正常亮度而言，就该是它的几万倍，这个自然界1-10000的黑白过渡，要如何用电脑的256阶模拟呢？  
  
就是HDR。  
HDR技术会按一个方法（ToneMapping）将1-10000的过渡缩放成0-255的过渡，这样一来，在有着强烈明暗对比的场景中，就不会出现大块黑或者大块白了。  
也就是说，更贴近自然，更贴近人眼。因为HDR会尽可能的让当前的画面对比度柔和地铺开，覆盖最暗到最亮的范围。无论这个画面偏暗还是偏亮，对比度低还是高，它都会处理成看起来最真实最柔和的过渡上。  
HDR的H（高）指的就是自然界的高阶亮度对比度；  
HDR的D（动态）则是指根据当前屏幕画面的明暗对比度随时更改缩放率；  
HDR的R（范围）则是指根据一定范围（场景/屏幕/局部，依需求而定）内的明暗对比度进行计算模拟。  
  
这么说很难理解吧，请看下这张图：  
  
根据我们的直觉，A比B更暗，不是么？但实际上，如果你用画图板什么的切一小块做对比，就会发现A和B其实一样亮！  
这是由于人眼的局部适应所致，也就是说，人眼会因为A周围的较亮而认为A较暗，同理，会因为B周围的较暗而认为B较亮。这样，尽管AB一样亮，但我们的感受却不是这样，因为我们的视觉系统对亮度进行了重新分配，这就是人眼的HDR效果！  
  
于是，如何重新分配亮度，也就是把画面的亮度缩放到0-255的范围以内呢？  
你可能会说：“这还不简单吗？把画面最暗的当作0，最亮的当作255，中间插值就好了嘛。”  
没错，这是最简单的，计算公式就是：  
像素最终亮度=（像素原亮度-当前画面最小亮度）/（当前画面最大亮度-当前画面最小亮度）  
也就是，将当前画面上的最大最小亮度范围映射到0-1的范围上，然后同比例缩放各个像素的亮度  
\*虽然我们常说0-255，但其实对于计算而言它们被表示为0-1的范围，因此后续公式中将使用0-1代替0-255  
  
类似的公式还有更简单的：  
像素最终亮度=像素原亮度/（1+像素原亮度）  
这个公式不仅简单，而且非线性，它是无限趋近于1的递减函数，同时也很难出现得0的情况。不出现0和1这一点上要比上一个公式好得多，因为这样更自然。附带一提，魔兽争霸3的护甲计算也是这个方式。  
  
但是但是，便宜没好货。  
人眼的伟大远超我们的想象，为了模拟出人眼的种种特性，我们需要先求出平均亮度，再根据它进行映射，这就是ToneMapping。  
  
  
那么接下来，让我简单说下在计算机中的实现方法吧。  
为了模拟自然的对比度，我们需要三步：曝光（计算画面平均亮度），缩放（ToneMapping）和光晕（高亮部分模糊叠加）。  
  
我们先说缩放的计算方法。  
缩放后亮度=亮度系数\*原亮度/平均亮度  
\*平均亮度稍后在解说曝光时再提；  
\*亮度系数用于决定画面的整体明暗，可以用于控制画面风格（偏暗偏亮）；  
\*像素亮度=0.27R+0.67G+0.06B，这三个系数是根据人眼对不同色光的敏感度而定的，记住即可  
  
不过当然了，这个“缩放后亮度”并不在0-1的范围内，所以我们需要把这个值当作像素原亮度，代入最初的简单映射公式中，以便让结果位于0-1内，也就是  
像素最终亮度=缩放后亮度/（1+缩放后亮度）  
  
  
最后的重头戏，曝光——平均亮度的计算法：  
  
嗯，一如既往的图形学风格，扭曲的符号，变态的内容。简单来说，这个式子的意义就是：  
对于原始图像每个像素，计算出该像素的亮度值Lum(x,y)，然后求出该亮度值的自然对数。接着对所有像素亮度值的对数求平均值，再求平均值的自然指数值。其中那个δ只是个很小常数，比如0.0001，它的目的只是防止求对数的计算结果趋于负无穷而已。  
不要问我为什么要用自然对数和自然指数，因为底数——e就是这么个神奇的东西，自然是多么的伟大啊！  
当然，如果你想了解的话，可以去看看《Photographic Tone Reproduction for Digital Images》这篇论文，如果你可以看懂的话，最下面有链接供受虐狂使用。  
  
OK，就是这些了……等，还忘了一个？  
那么作为休息，来看看光晕的做法吧，绚丽的光晕效果，其实现比你想的要简单得多，大跌眼镜吧。  
做法就是——  
把高光部分提出来，模糊，贴回去。  
说的更具体点，就是——  
找个临界值，把画面分成黑白两部分，然后缩小一点（为了模糊效果更好）进行高斯模糊，使白色部分扩散到黑色区域中，然后缩放回原大，与画面的RGB值合并。  
就像这样：  
  
  
这样就都讲完了，希望了解这些底层原理有助于你的实现或工作。  
不过当然，一般而言，这些功能会由引擎来实现，不过对flash而言，经常会可遇不可求，所以还是有备无患吧。  
  
更多具体解说请参见：  
<http://dev.gameres.com/Program/Visual/3D/HDRTutorial/HDRTutorial.htm>  
↑这里包含有《Photographic Tone Reproduction for Digital Images》这篇论文，并且通俗易懂  
<http://blog.csdn.net/ccanan/article/details/6745207> ↑CSDN的，与上一个链接不同，这里用的是DX，所以注解有些少