Unite 2018 | 《崩坏3》：在Unity中实现高品质的卡通渲染（下）

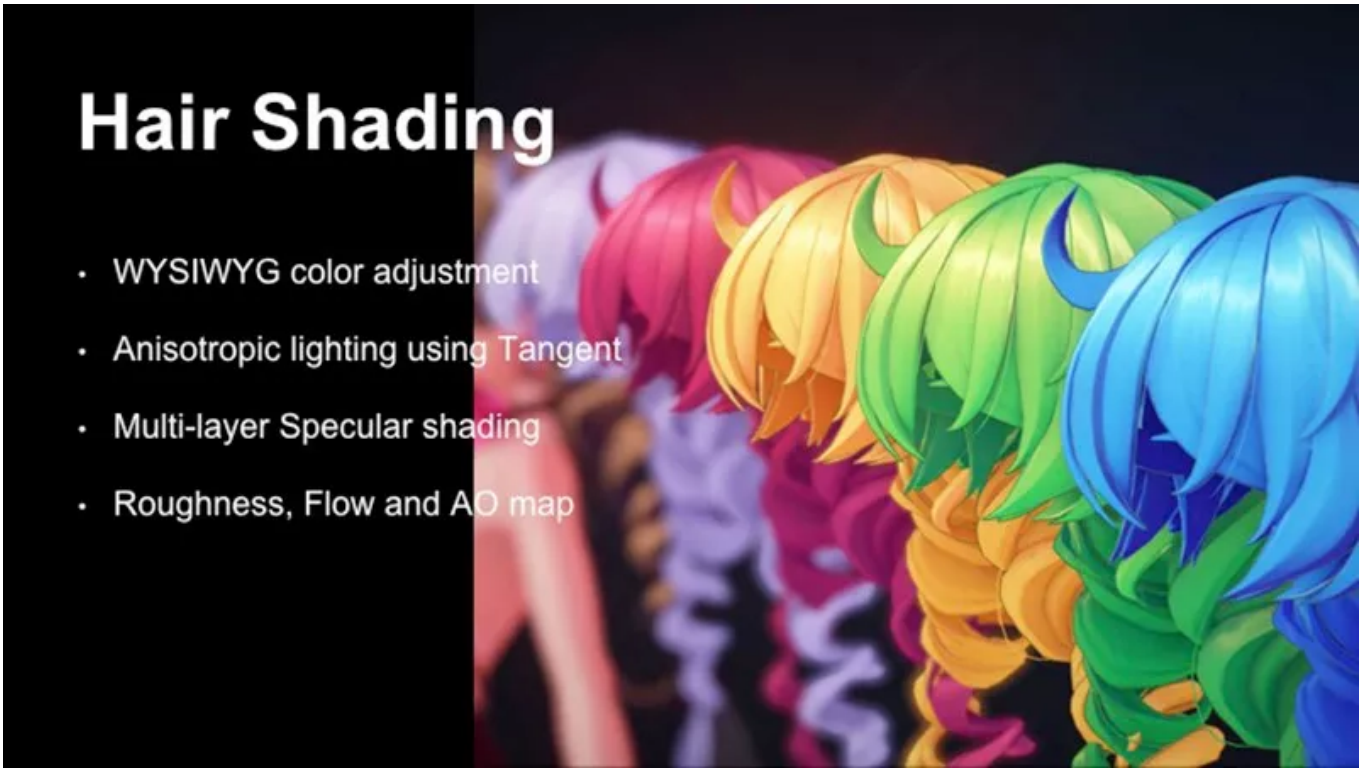
贺甲 Unity官方平台 2018-05-24

今天我们继续分享米哈游技术总监贺甲在Unite Beijing 2018大会上的演讲《在Unity上实现高品质卡通渲染的效果》下篇，上篇请点击此处阅读。

下面为演讲内容：

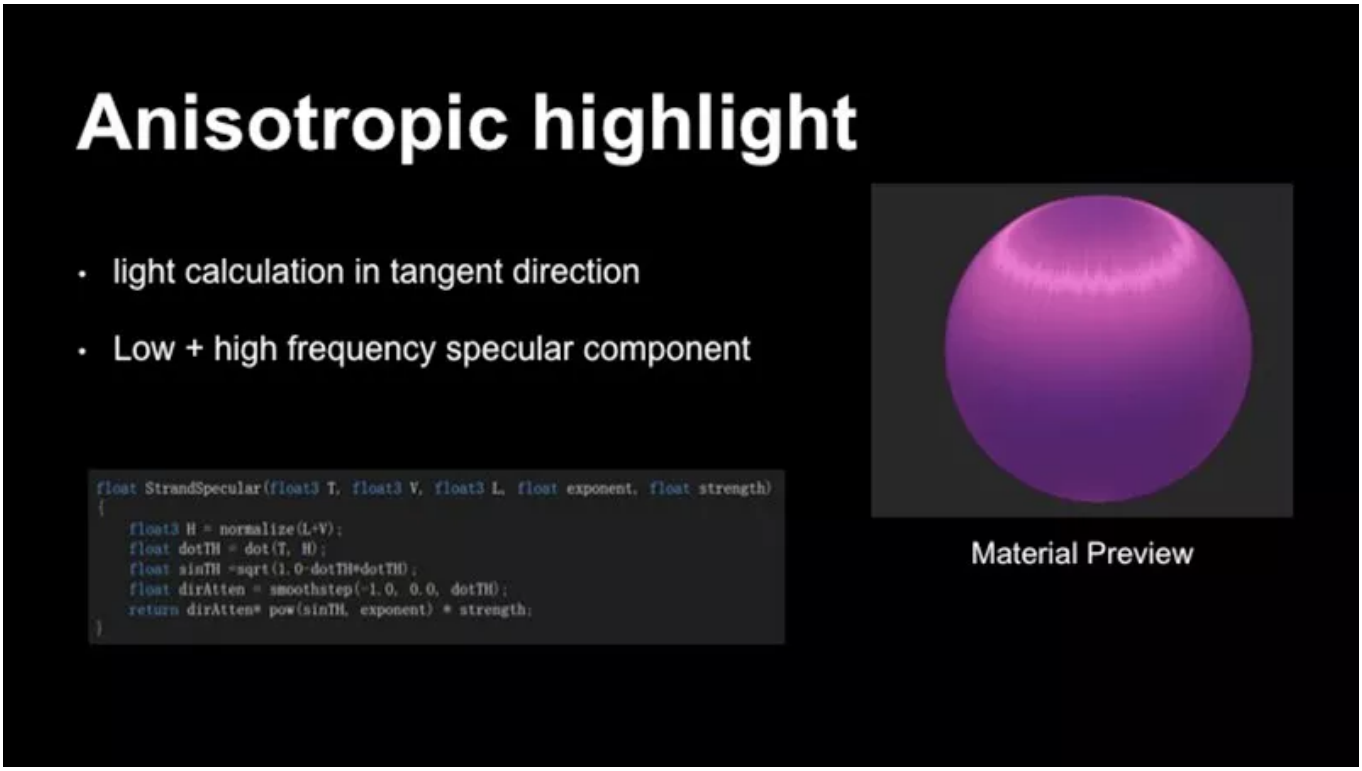
接下来我们就来介绍一下头发的渲染。头发是卡通渲染角色较为重要且独特的部分。我们想要实现根据光源动态变化的高光和阴影渐变，并且这个实现还应具备直观的所见即所得的色彩调节能力。

和皮肤的材质一样，对于头发的漫反射渲染我们同样使用了Multi-ramp的方法，而镜面反射高光我们则使用了二层高光做叠加，通过组合高低频的高光成分在一起，我们可以得到满意的结果。此外，我们还使用Glossy Map和AO纹理来进一步增强头发的质感。

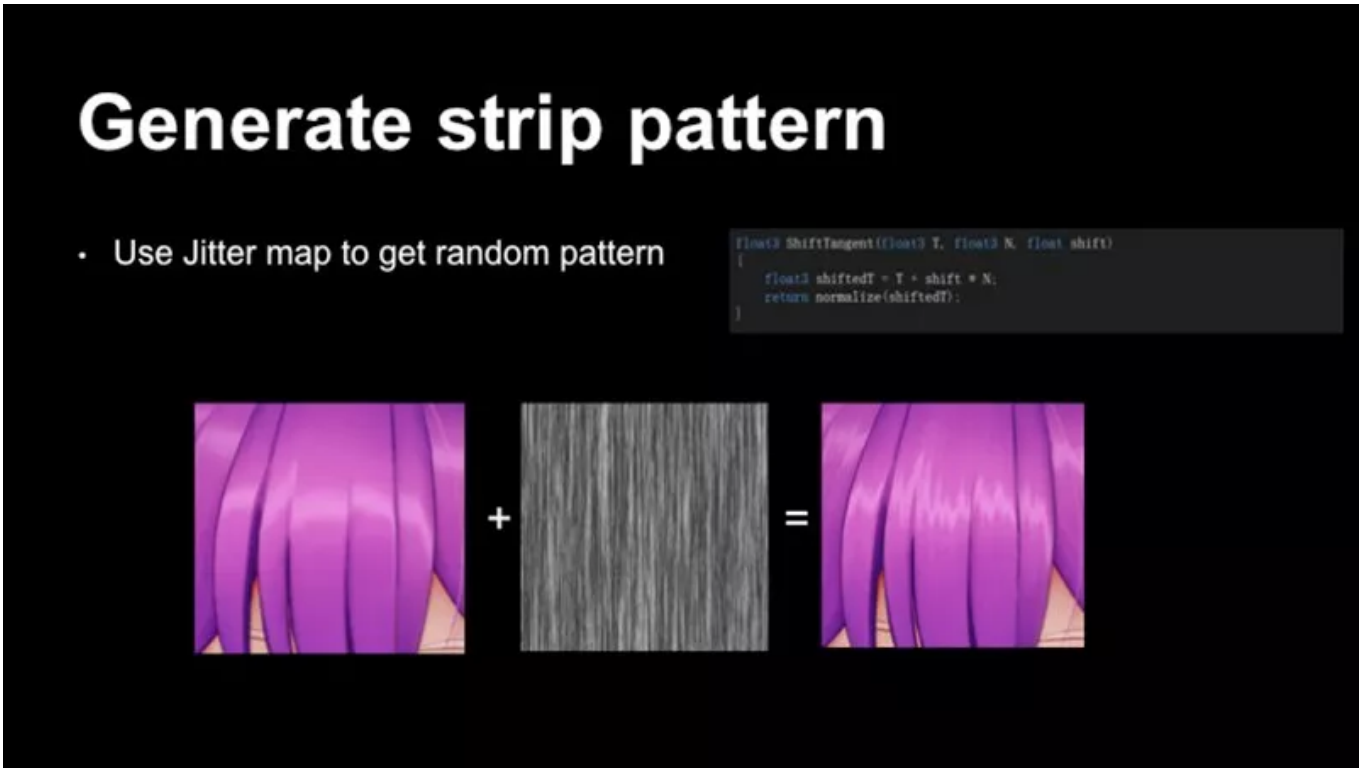


头发的高光渲染使用了各向异性高光，相比普通的高光使用Normal计算光照，各项异性使用Tangent作为计算基础，因此可以使高光显示出垂直于发丝方向的形状。

我们在制作头发模型的时候，如果模型拓扑较为复杂，UV展开较难做到全部垂直，我们也可以使用Flowmap来梳理高光的形状。



我们还使用Jittermap抖动贴图用来增强卡通渲染头发的质感。通过扰动切线方向来达到模拟发丝细节的高光效果。另外，通过调整Jittermap的UV scale还可以做到调整发丝的高光粗细。

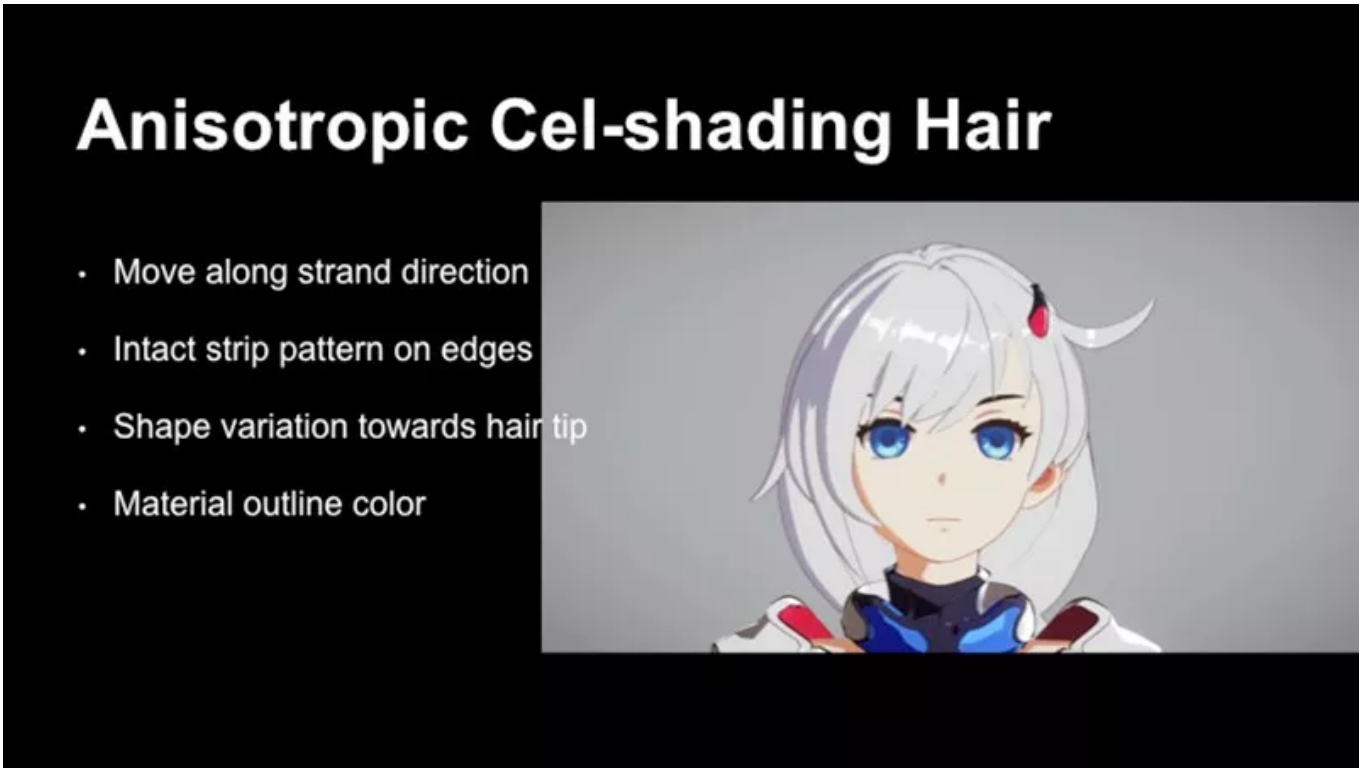


这四张图分解展示了各个高光成分对渲染结果的影响。而右下角的则是最终的图像。我们可以看到，结合了低频和高频成分的高光显示，头发看起来更具表现力。

接下来让我们看看另外一种实现Cel-shading头发高光的实现。

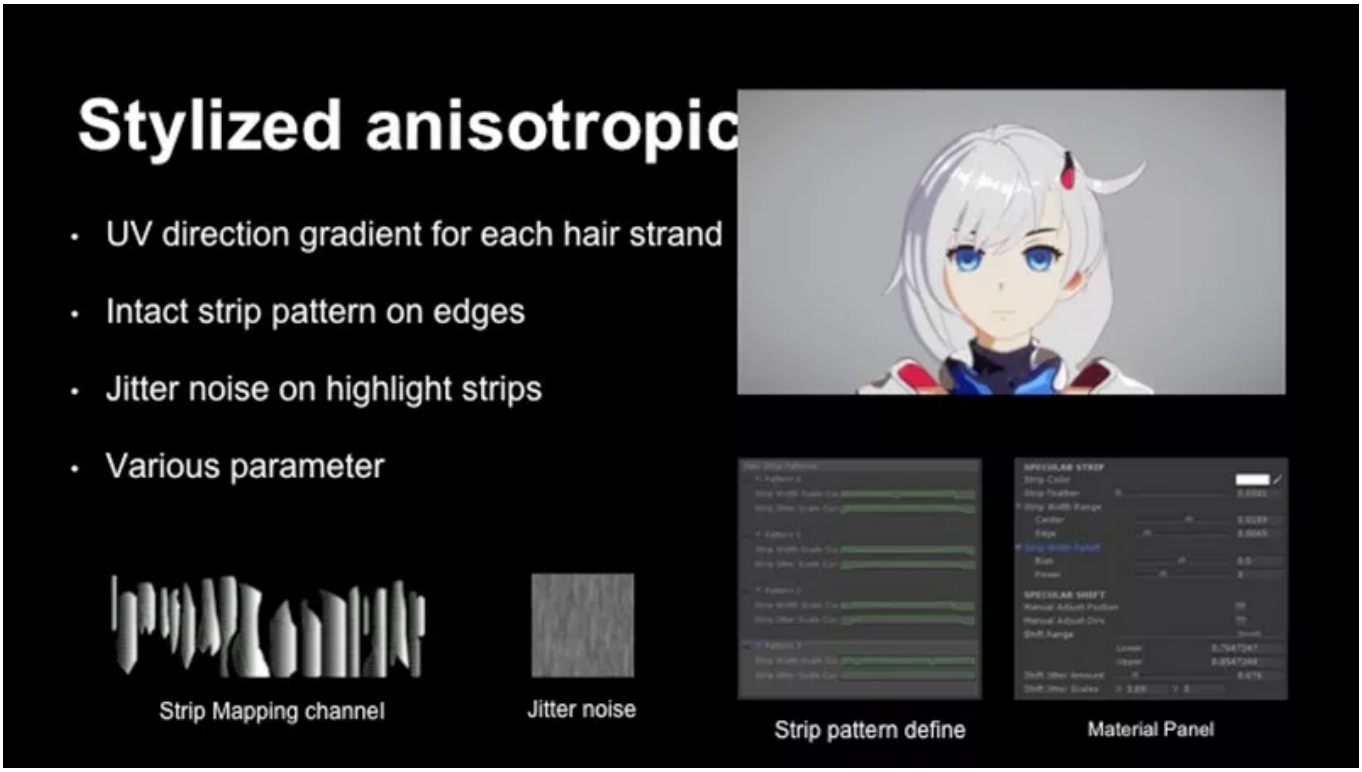
我们的目标同样是使其可以动态化，高光应根据光源和相机位置沿发丝方向移动，形状也应该在移动中有着动态的形态变化。

Cel-shading风格的头发高光较为独特的形态，很难用传统的高光计算方法来描述。同样我们需要使用切线方向而不是法线来进行高光计算，并且需要更为特殊的方法去表现高光形状。



首先，我们要把每缕头发模型在垂直方向进行UW展开，以便高光可以沿着每根发束移动。然后从将每一缕从左侧向右侧填充0到1，用来标识动态生成的高光形状的起始和结束位置，我们使用几个曲线定义的模板来描述头发高光的基本形状，然后使用抖动噪声纹理来调制头发高光的粗细变化，

材质方面有很多参数用来控制生成图案的形状。位置、偏移、宽度、抖动比例等，通过调整这些参数，我们可以根据需要获得各种不同的形状。



我们来看另一种各向异性材质的例子：丝绸。这次我们使用了副法线方向来计算了高光反射，并使用三个高光层合成在一起获得最终的渲染效果，我们为每一层分别设置不同的颜色，以便最终材质看起来色彩层次较为丰富。

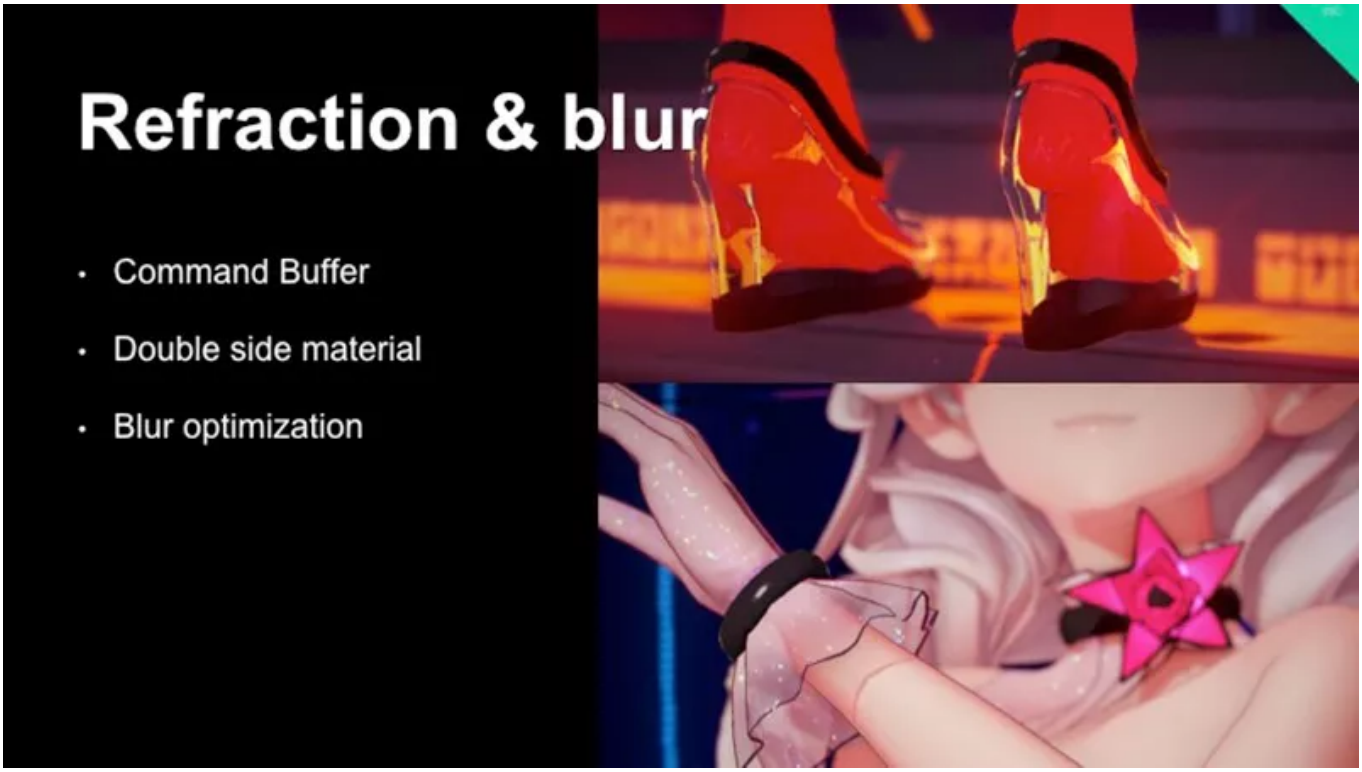
这里显示了在不同的视角下，丝绸各向异性高光的反射变化。

我们的角色材质中还包括其它特殊的材质，例如：水晶和纱巾等半透明材质，直接使用Alpha混合不能表现出应有的质感，这就需要我们实现折射和模糊效果，这二个效果都依赖于Unity的Command buffer。

实现折射效果时，Command buffer在渲染折射前获取已经渲染好的Backbuffer作为背景，用于折射采样, Rgb通道设置不同折射系数，分别采样三次来模拟色散效果。

对于模糊效果，则是用Command buffer将Backbuffer降采样并做模糊，生成4张尺寸依次减半模糊度递增的RenderTexture，然后根据相机距离和FOV以及材质固有的模糊参数，确定模糊程度，选择对应的RenderTexture来完成模糊效果。

我们还对这二者的实现做了一定的优化，不对直接对Backbuffer使用全屏模糊，把物体本身作为Proxy mesh，只处理需要画的部分。

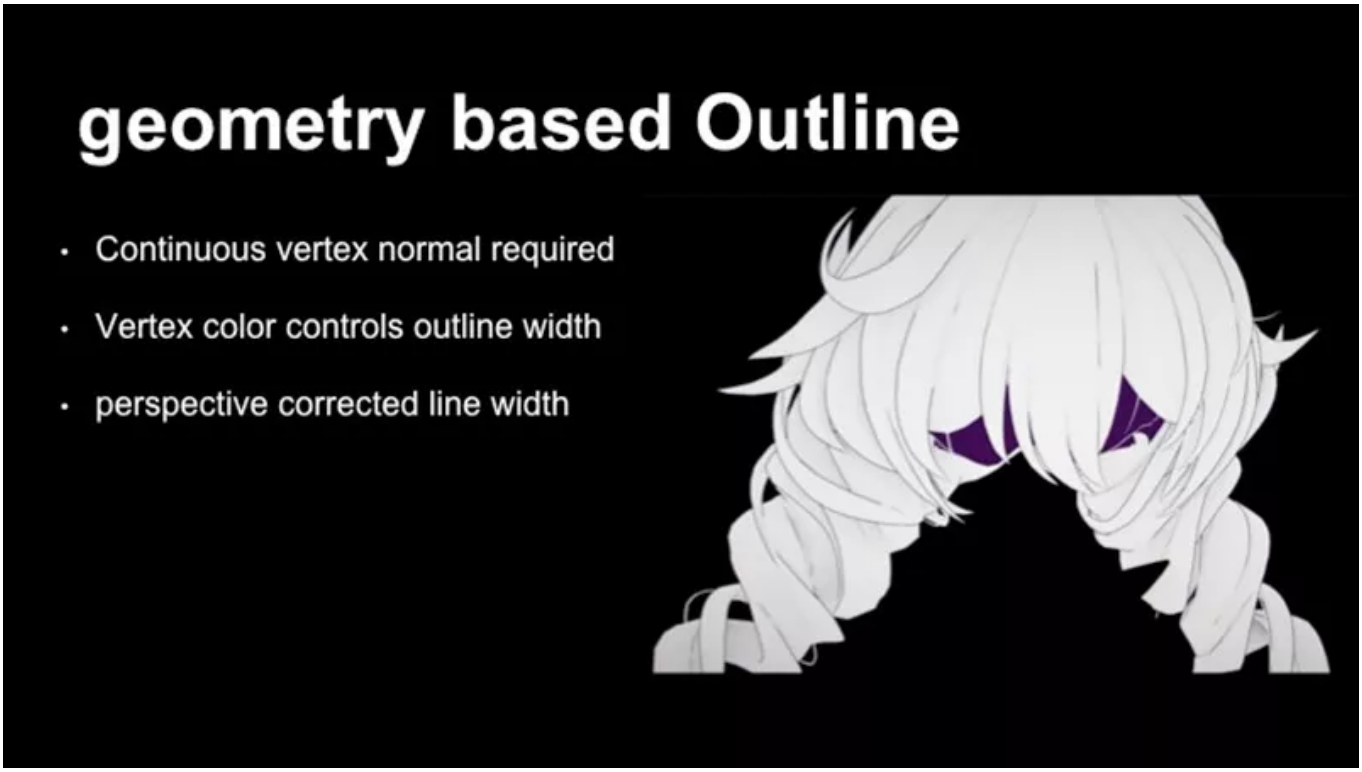


接下来让我们来谈谈高品质勾线的方法。

对于角色和动态物体我们使用Backface勾线方法，并使用顶点色对勾线的宽度进行控制，勾线本身需要连续的顶点法线才能在锐角边不会出现断层，因此我们将平滑过的法线存储在另一套顶点色里。

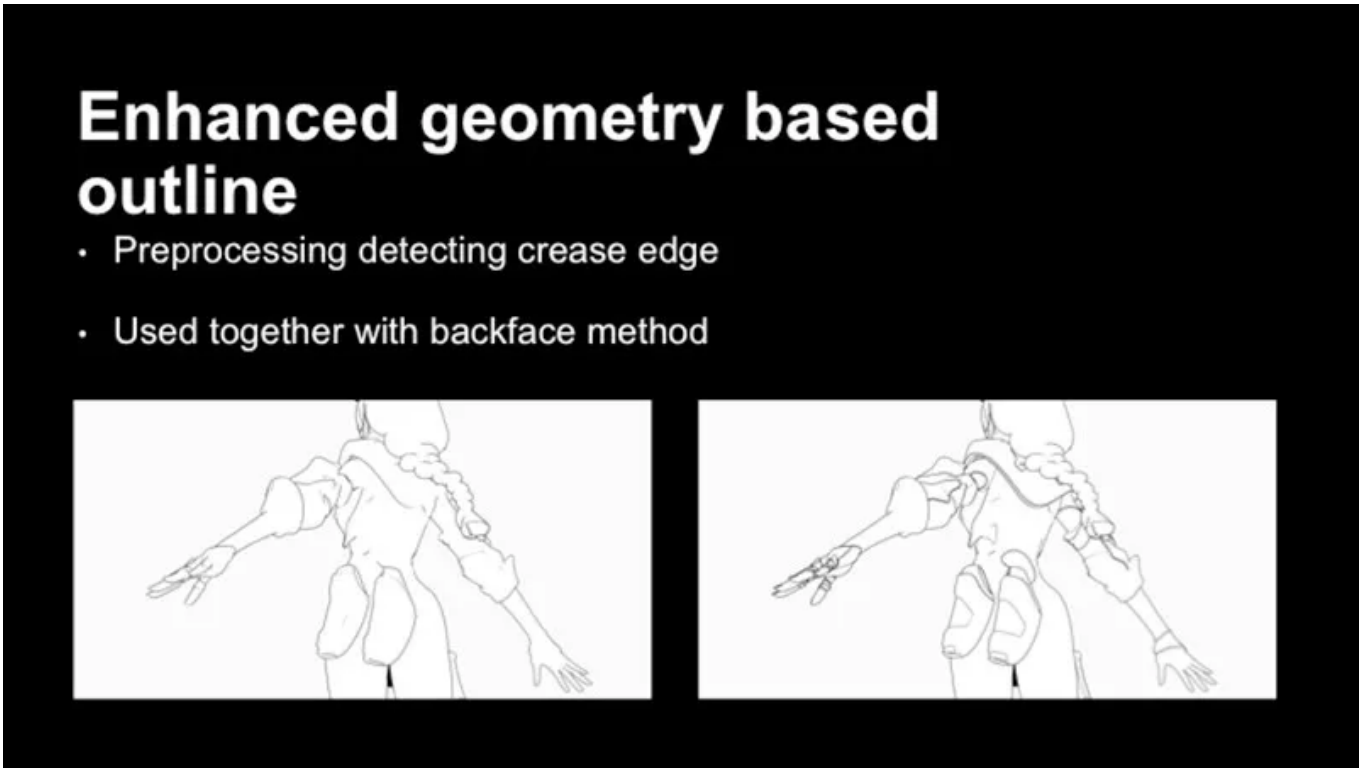
此外，我们也使用顶点色来控制勾线宽度，例如：发尖处勾线会逐渐变细，我们通过在顶点颜色填充渐变为0的值以使线条宽度逐渐过渡到零。

另外，根据相机与物体之间的距离，还应有基于距离修正的勾线宽度。每种材质上也应该有对应的不同勾线颜色，所有这些功能都是高品质的勾线所必需的。



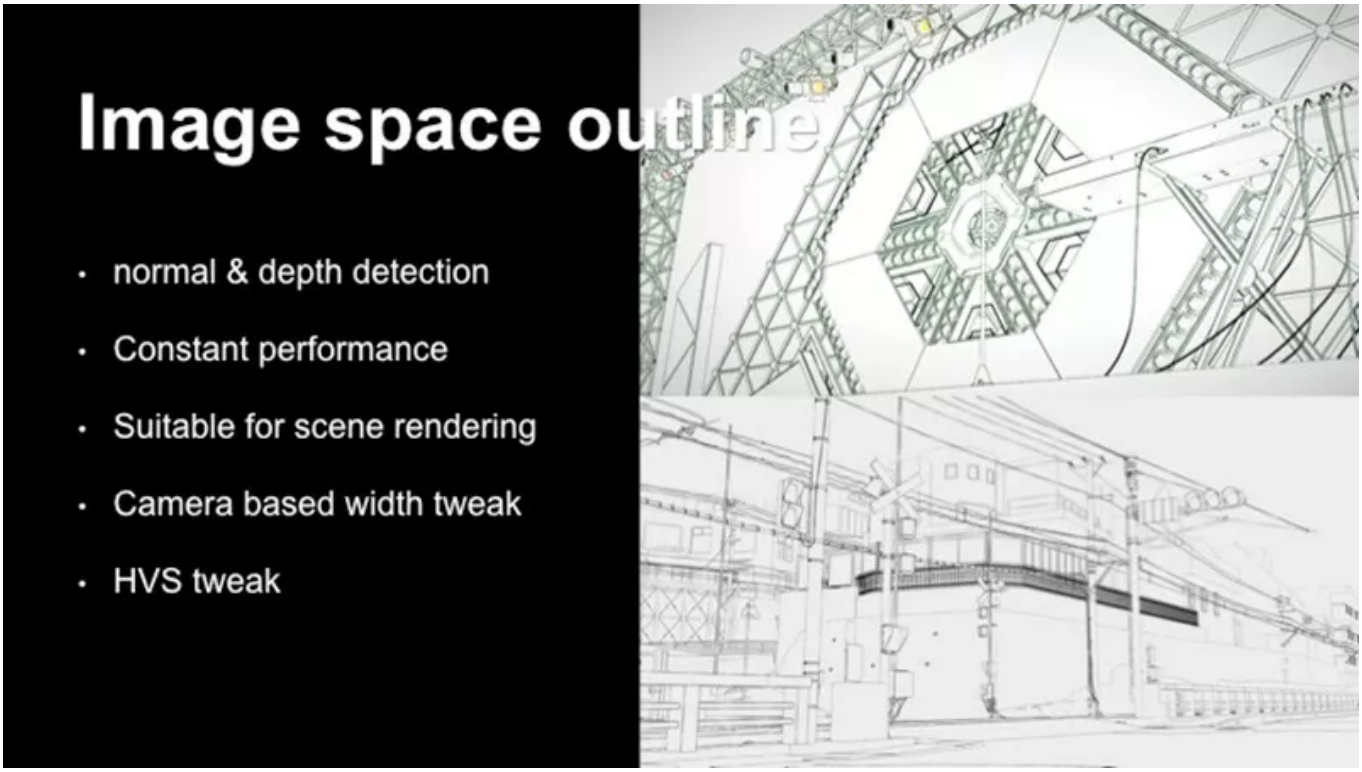
Backface勾线方法虽然可以做到较为细致的勾线还原。但也有着自身的固有缺陷，那就是不能在非边缘的尖锐折线处产生勾线。 而这些折线在硬表面模型上是很常见。

为了解决这个问题，我们添加一个预处理过程来提取这些边缘，并将它们保存到额外的Mesh资源中，并使用Geometry shader绘制它们。对于这些折线我们使用了和Backface法类似的调整参数，从而使它们看起来完全相同。增加了折线的绘制之后，我们可以看到右侧的图片捕获到了更多的勾线细节。



勾线另一种常见方法就是在图像空间中生成轮廓线。通过检测场景图像中Normal和Depth的不连续性，我们可以获得细节较为丰富的勾线。无论场景的复杂性如何，这种方法的性能都是恒定的，我们还添加了对勾线颜色的色相、明度、饱和度的调整，使勾线更为自然。

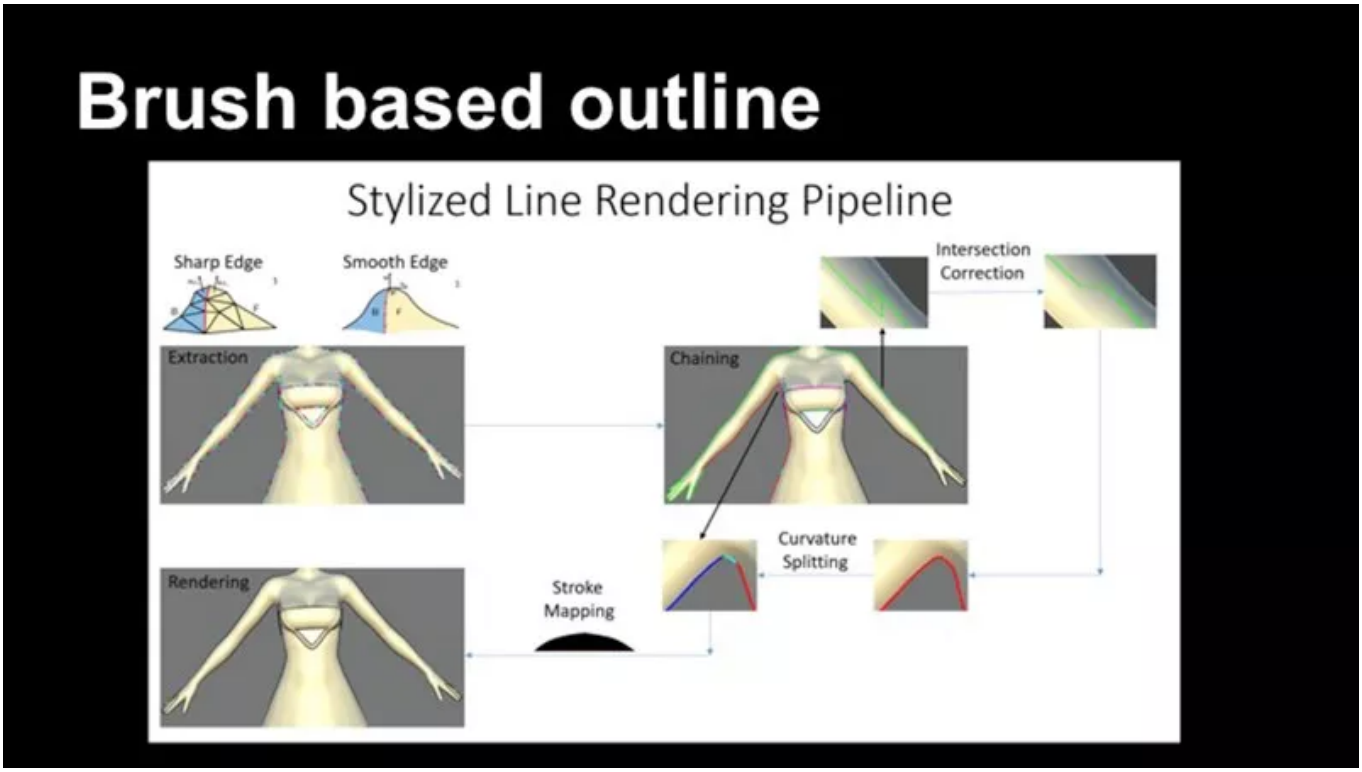
这种方法的缺点则是较难控制勾线的宽度，如果我们想实现距离相关的线宽，我们只能在几个像素的范围内调整它，因此基于图像的方法主要适用于场景轮廓渲染，对于靠近摄像头很近的物体，我们最好使用Backface的方法。



最后一种做法是基于笔刷的购线方法，这在离线渲染中使用的比较多，通常分为以下几步。

- 轮廓线提取：从Mesh上提取轮廓边，主要分为Sharp Edge和Smooth Edge二种。
- 连接轮廓线：根据模型的拓补关系，将相邻的轮廓边连接成尽可能长的轮廓线。
- 轮廓线分段：在步骤2的基础上，根据轮廓线上曲率和可见性的变化，将轮廓线在曲率或可见性的突变处分开。
- 笔触映射：将想要添加的笔触制作成纹理，根据对应的纹理坐标映射到步骤3的轮廓线上。

这种方法可以达到更为风格化，笔触更明显的勾线方式，Pencil+ blender里Freestyle render基本都是采用类似的方法，性能开销较大，可以用于CG品质渲染，但不适合直接在游戏中使用。



接下来我们来看看其它特殊效果的实现，这些渲染效果在场景刻画中同样起到重要的作用。

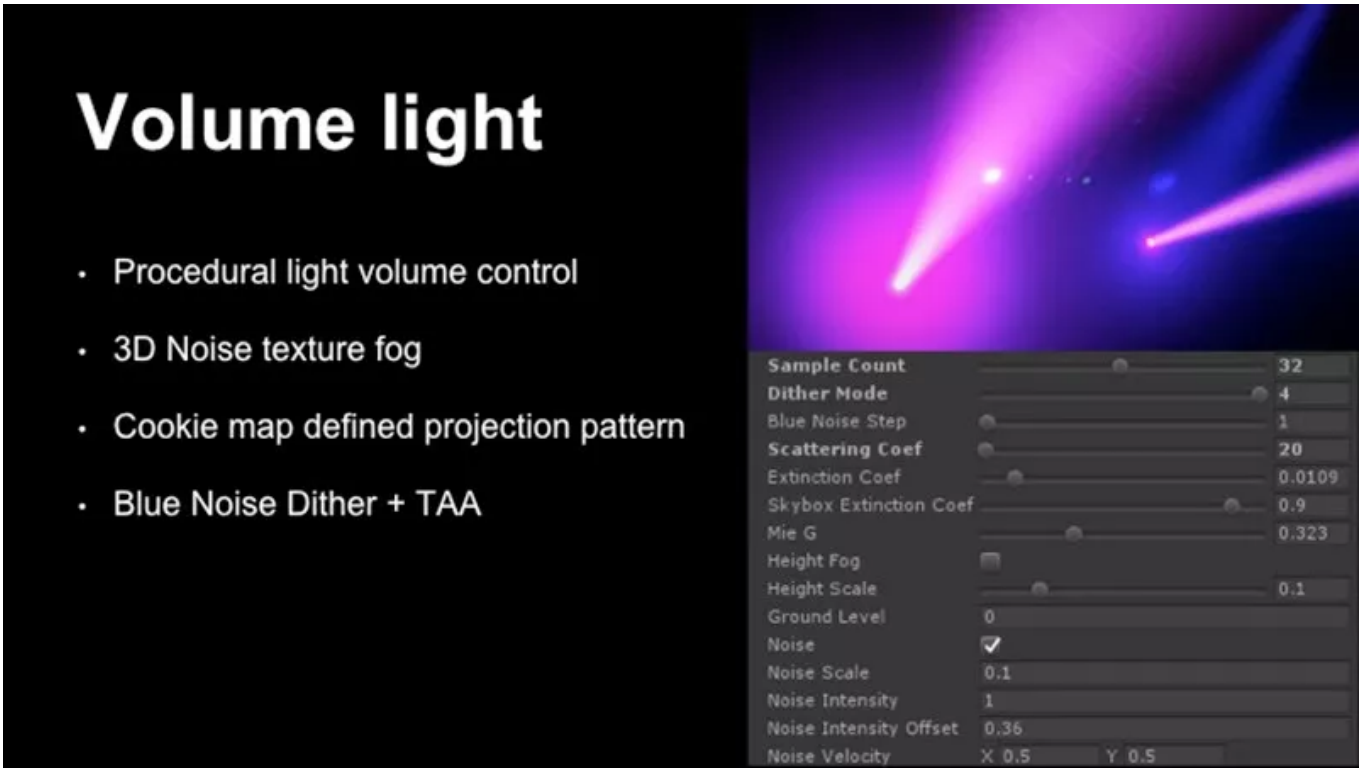
这是一段用来展示体积光的场景。我们可以看到，具有雾效的体积光配合bloom一起使用，场景表现出了较强层次和氛围感。

下面就来看看体积光的实现细节。

我们使用Unity内置的曲线来体积光的形状，这在运行时也方便调整形状，强度参数变化同样由曲线定义。

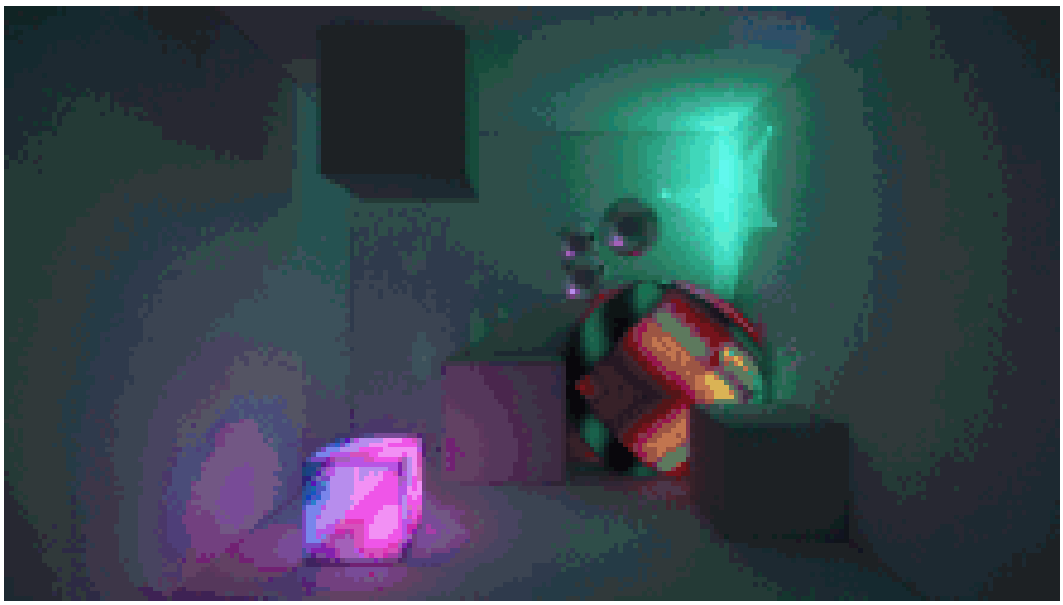
为了进一步模拟烟雾效果，我们还使用3D noise纹理来模拟动态烟雾流动的效果。Noise烟雾本身也有一些参数可调.。例如：粒度大小、尺寸比例、噪声强度、流动速度等。

此外，配合Cookie map还可以自定义体积光投影形状，使用Cookie map后同时也引入了高频的变化成分，这就需要对应增加采样数来减少走样，使用抖动算法可以减少采样不足导致的走样，我们实现了二种抖动方式：Bayer pattern和Blue noise， 通过实验发现Blue noise配合Temporal AA可以在较低的采样数下实现处较好的体积光效果。

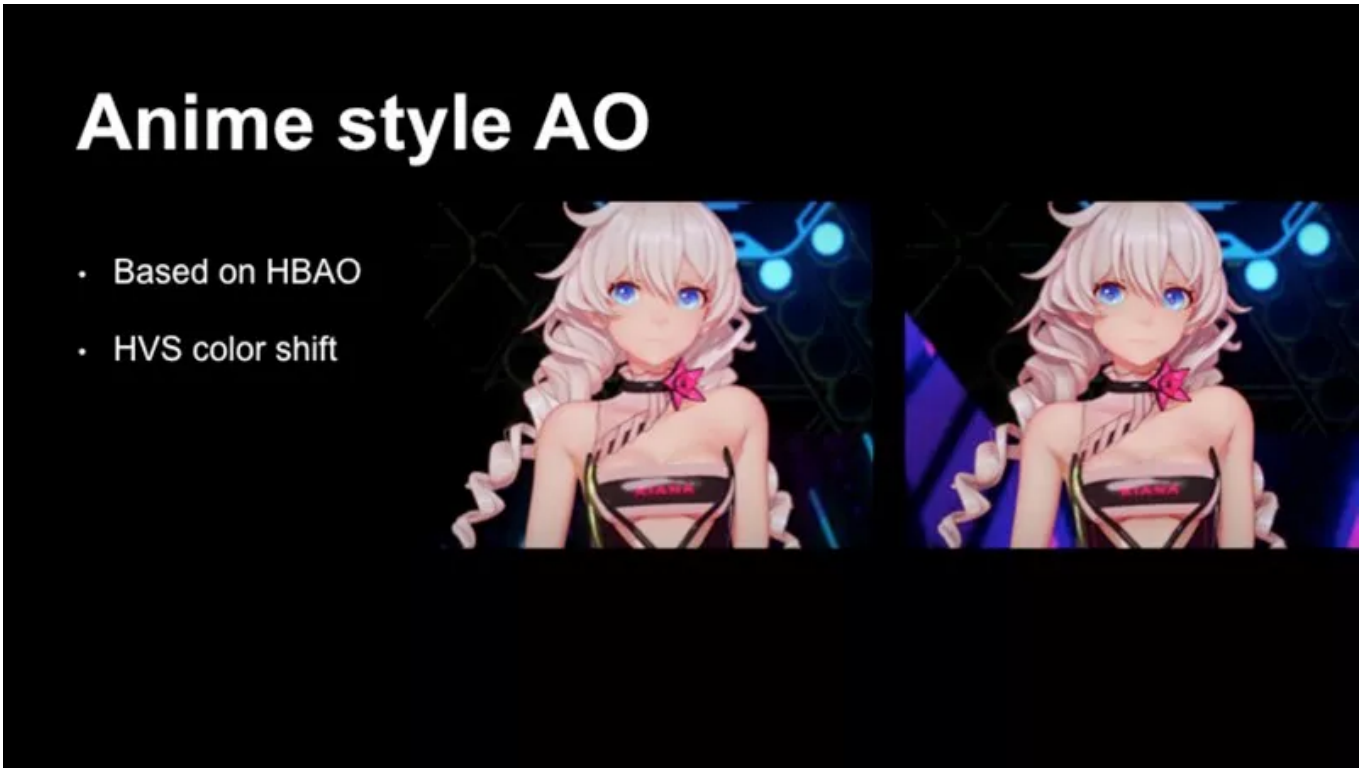


接下来我们来看一下使用Real-time GI的例子。

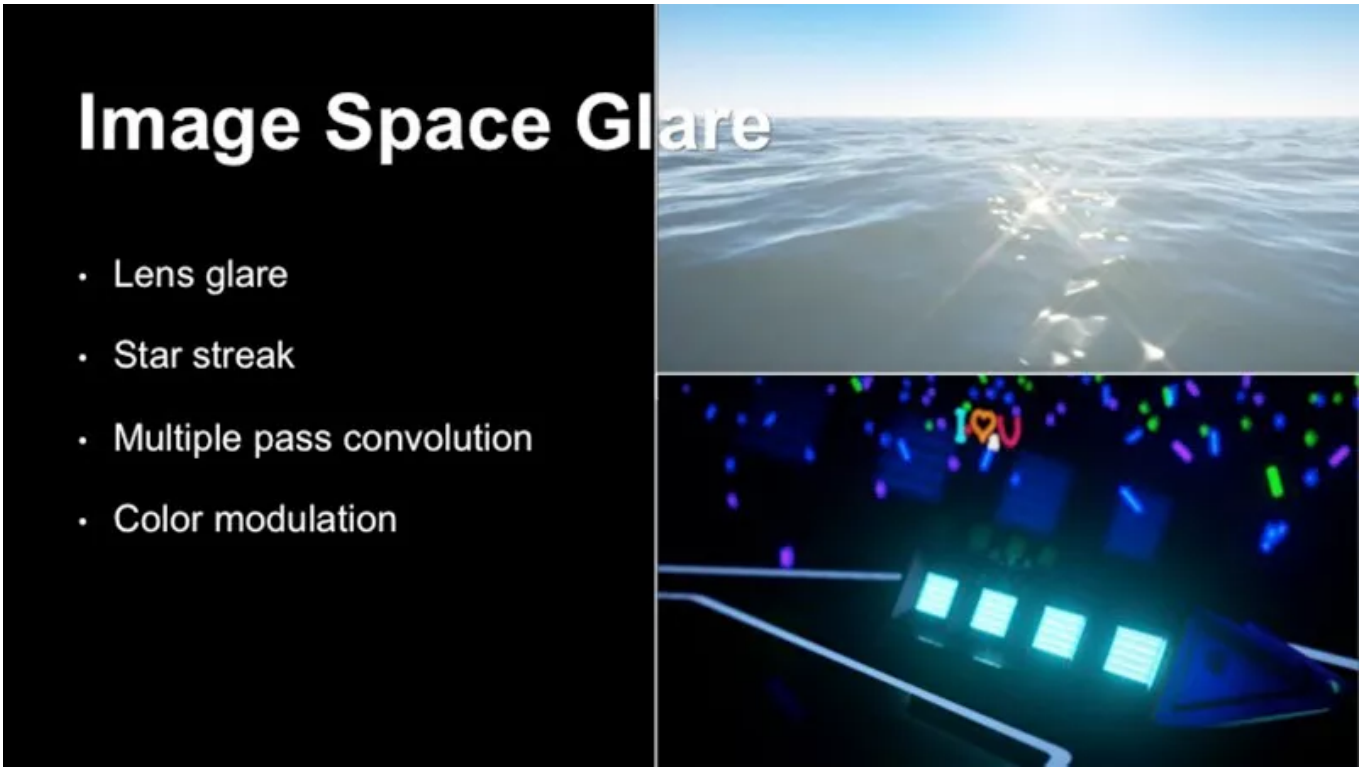
在这个简单的演示场景中，我们使用Enlighten来烘焙 Real-time GI的Lightmap，然后使用动态自发光材质和体积光作为光源，我们使用AVpro插件解码视频文件，将其设置在自发光纹理上，并设置强度值为1以上。我们就可以获得一个动态且明亮的面积光源，同时要记得更新GICache，以便在运行时刻可以动态更新光照环境，当与动态体积光一起使用时，整体的照明效果看起来令人印象深刻。



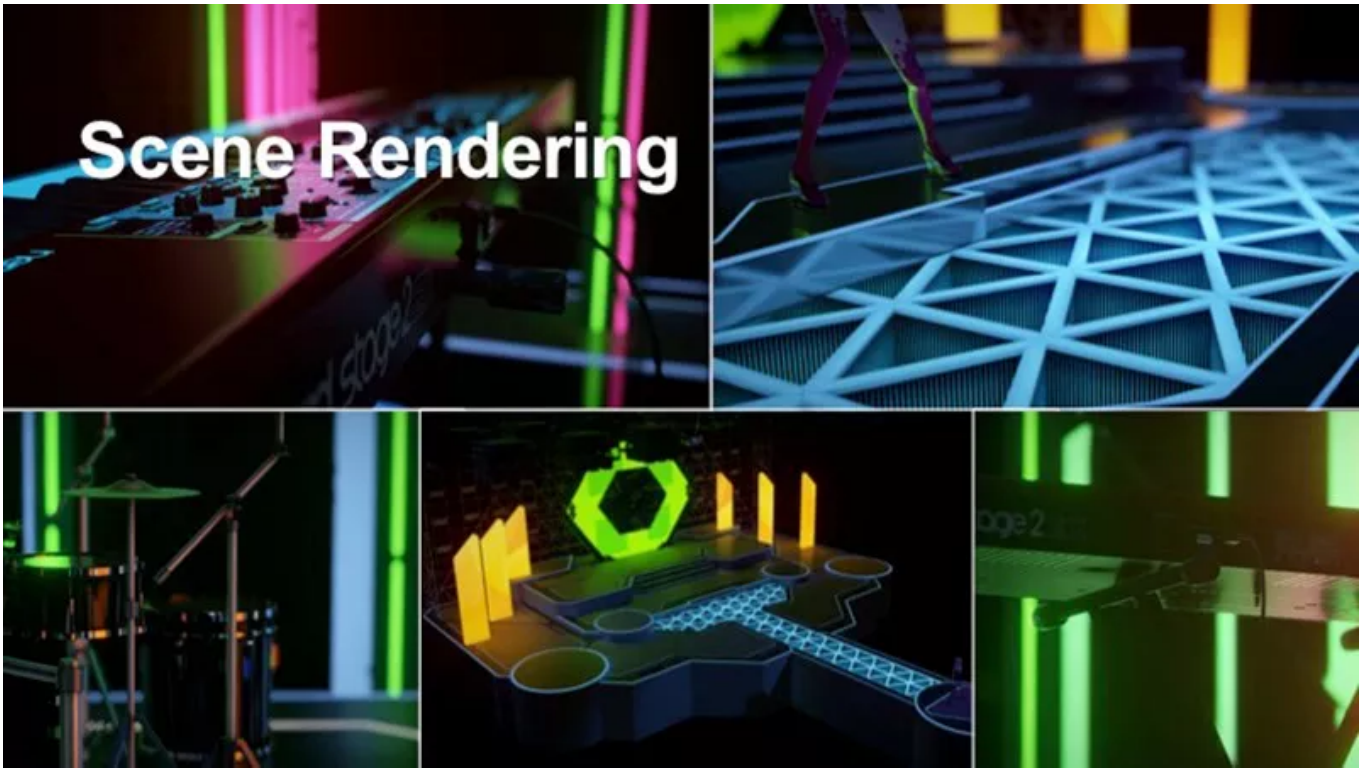
对于角色上的动态AO实现，我们使用修改过的HBAO，用于指定AO区域中颜色的饱和度和色调调整，以使加入AO后的图像颜色看起来不会变脏，通过对比图我们可以看出，在应用了AO之后，右图比左图层次感更强。



我们还重新实现了适用于卡通渲染的基于图像的眩光效果，用于模拟镜头产生的鬼影和星形散射效果。这里使用与Bloom类似的方式提取的高光区域作为输入，然后进行多次不同方向上的卷积并应用色彩调制来获得最终结果。



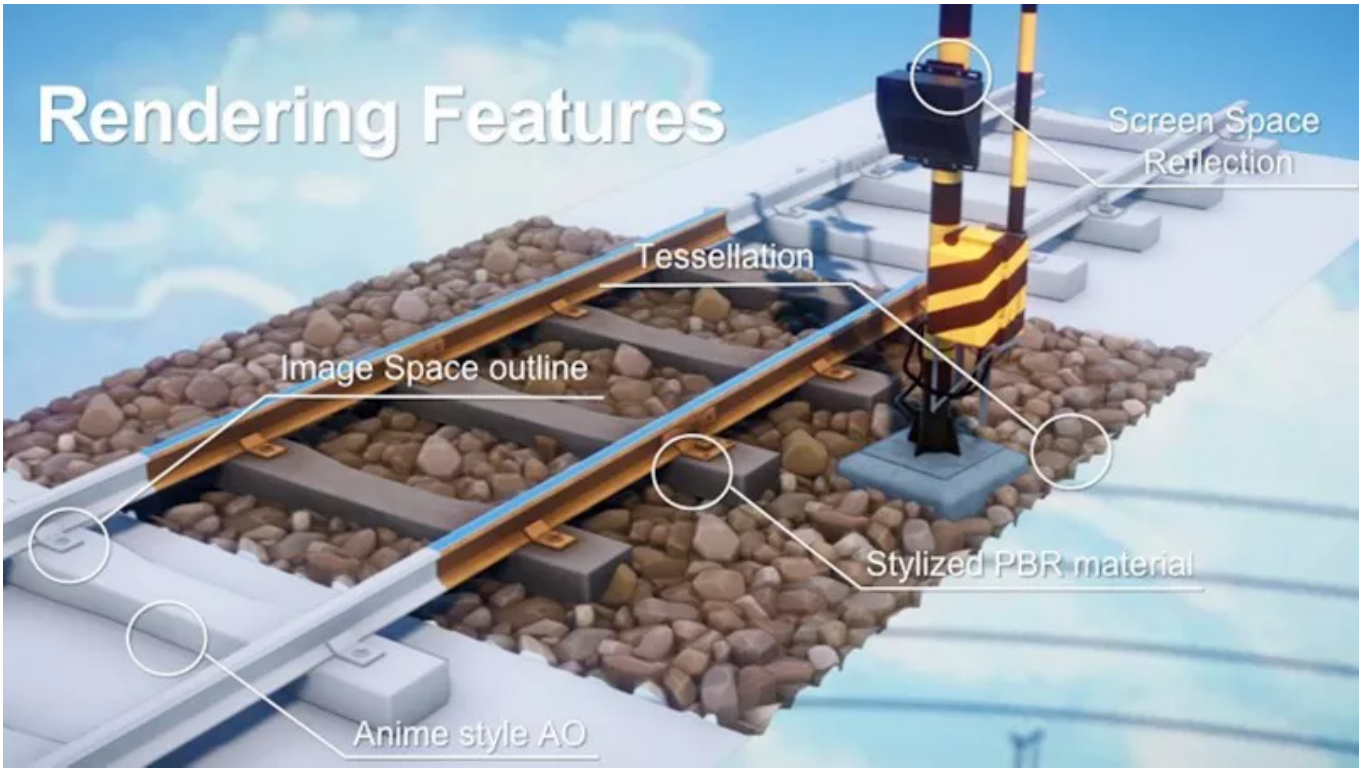
下面我们来看几张CG视频中的截图和特写。



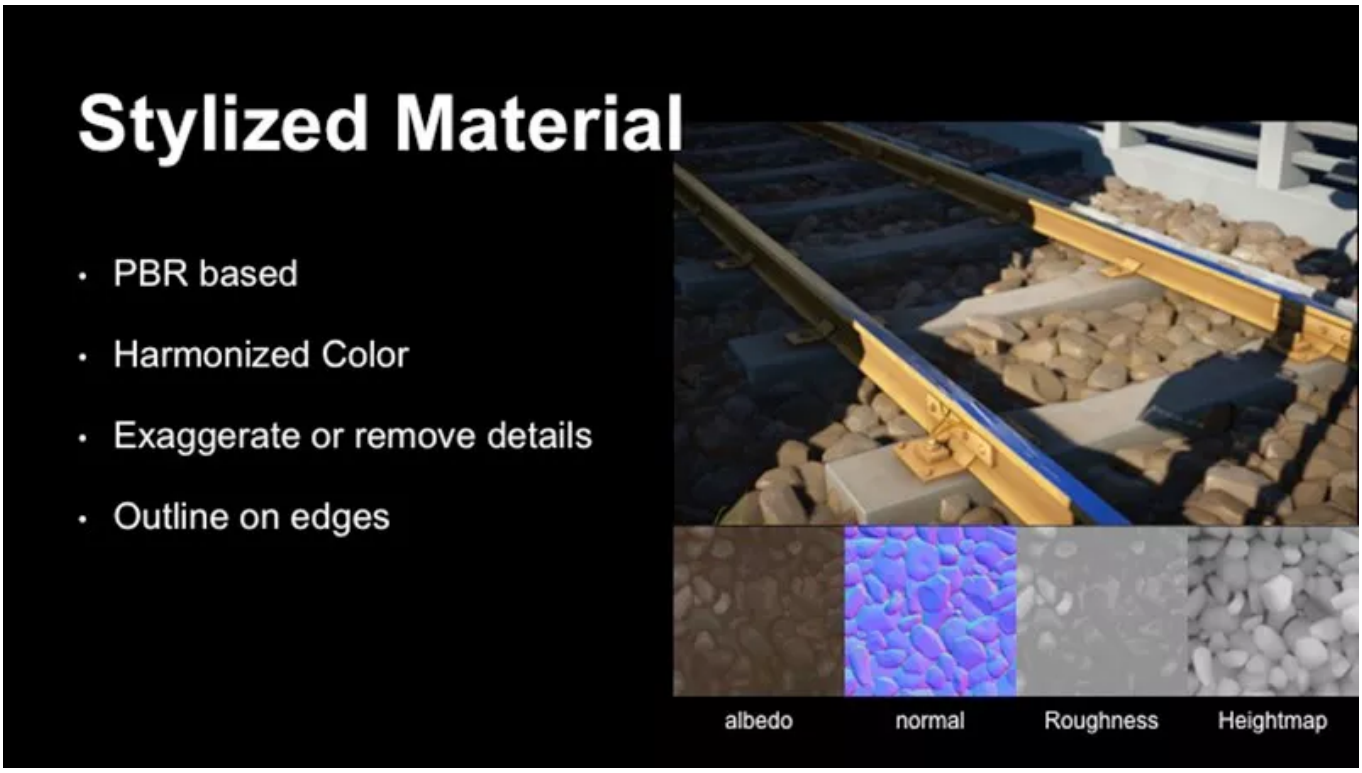
这是另一组场景截图。我们可以看到在应用了之前提到的这些渲染技术之后，整个场景可以更接近离线渲染的品质。

下面的图描述了上述场景中所应用到的主要渲染特性。从图中我们可以看到这些效果包括：风格化的PBR材质、卡通风格的AO、屏幕空间勾线、屏幕空间反射以及曲面细分等。

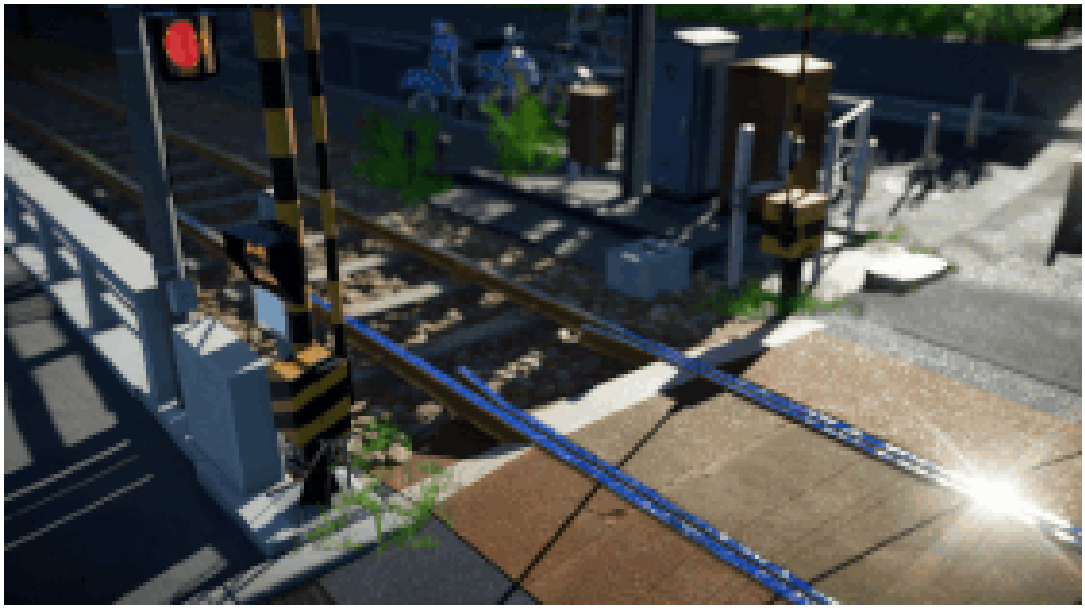
综合应用这些效果对于高品质的动画风格场景渲染起着重要的作用，我们的目标是在PBR的Shading基础上加入风格化的调整使其更具有表现力。



场景中的大部分材质都是基于物理的渲染。我们对PBR纹理集进行了一些风格化上的适应调整，例如：对于色彩的卡通化调整，以及对于物体材质细节的强调或省略。再结合使用图像空间的勾线来强调物体边缘，整体场景的表现就显得更接近动画风格。



下图展示了这些材质在不同光照角度下的光影变化。



下图展示了光影的变化。

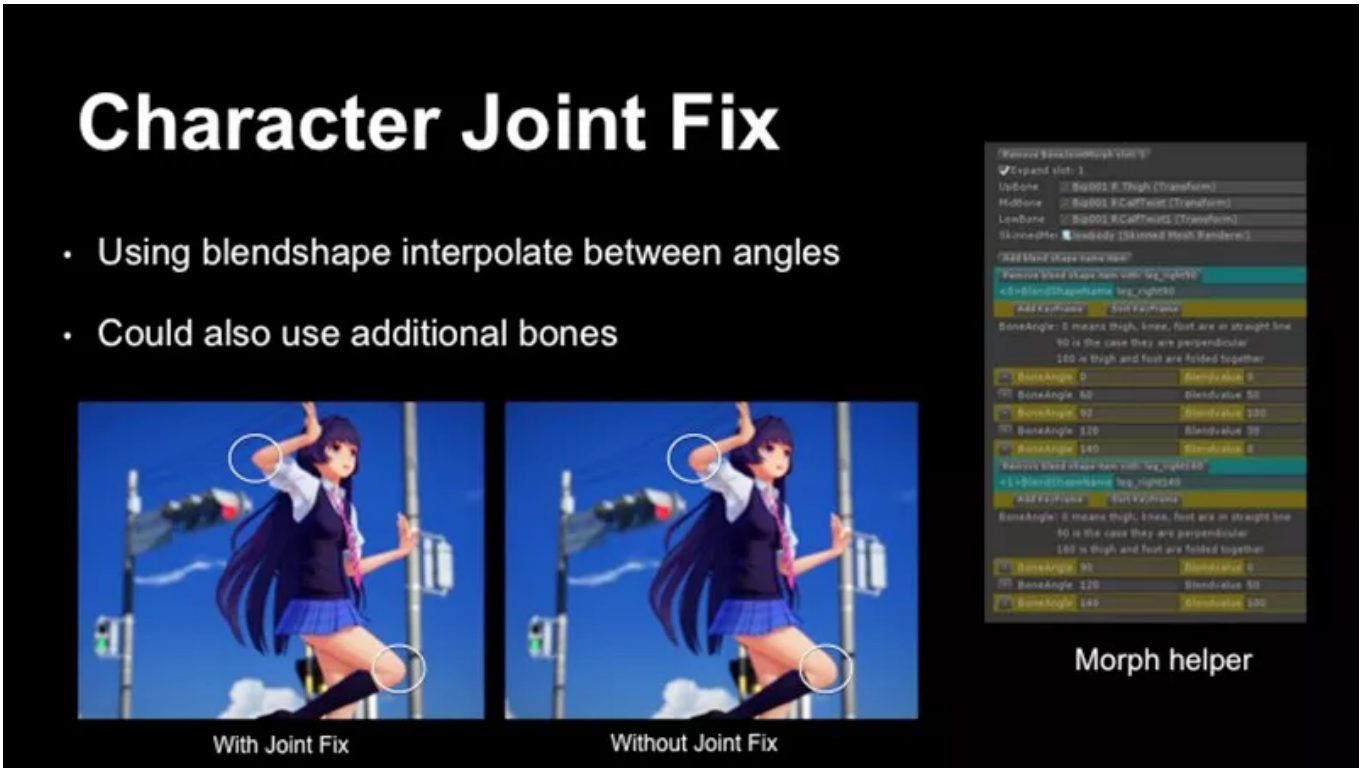
除了场景渲染之外，我们再来看看其它一些动画渲染所涉及的内容，动画表情。

我们使用Blendshape来制作面部表情。 眼睛，嘴巴和眉毛的表情独立为不同的部件单独制作，然后通过我们的自定义面部表情插件，来实现表情动画的及语音嘴型的自动映射。此外，我们还可以通过预定义不同的表情集合来在交互应用中驱动面部表情。

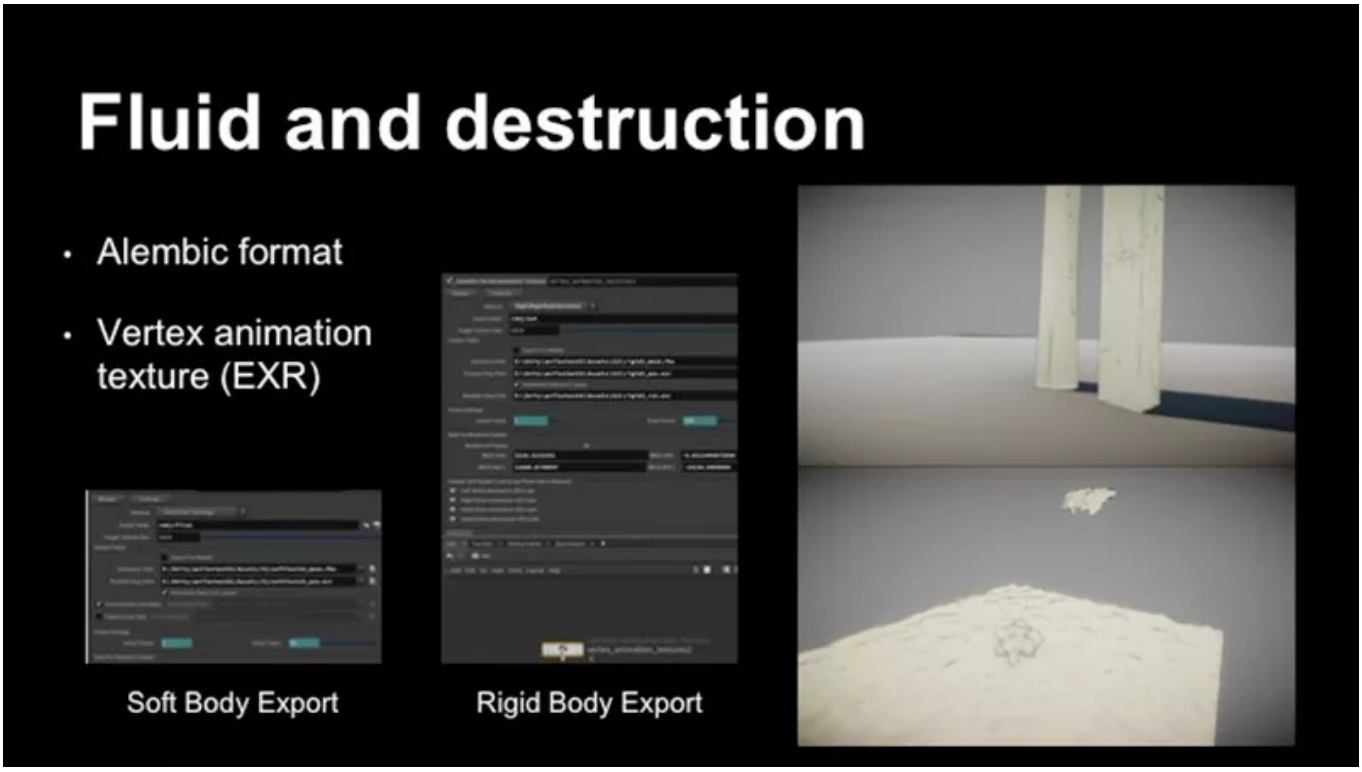
在Unity中使用Humanoid作为动画导入方式的时候，如果关节处旋转角度较大，按照动画品质的要求关节处的形状就不能令人满意。

为此我们通过在建模软件中，建立了每关节修正的Blendshape导入到Unity当中来防止关节变形。我们使用一个自动控制脚本根据关节旋转角度来差值混合形状。 为了确保更好的结果，我们为每个关节分别制作了二个Blendshape，一个用于90度，另一个用于140度以补正关节变形。

另外一种方法还可以使用额外的骨骼进行关节修正。这种方法更容易制作，但是对于结构细节的表现不如使用Blendshape。



为了可以表现更复杂的场景动态，比如流体和破碎的场景，我们可以使用Alembic格式，或者用EXR纹理作为载体从Houdini或其它DCC工具导入顶点动画资源。Houdini对EXR纹理格式导出顶点动画提供了很好的转换支持，对于Real-time的应用而言，顶点动画纹理在因为是在GPU上运行，运行效率及加载速度要快于Alembic格式。

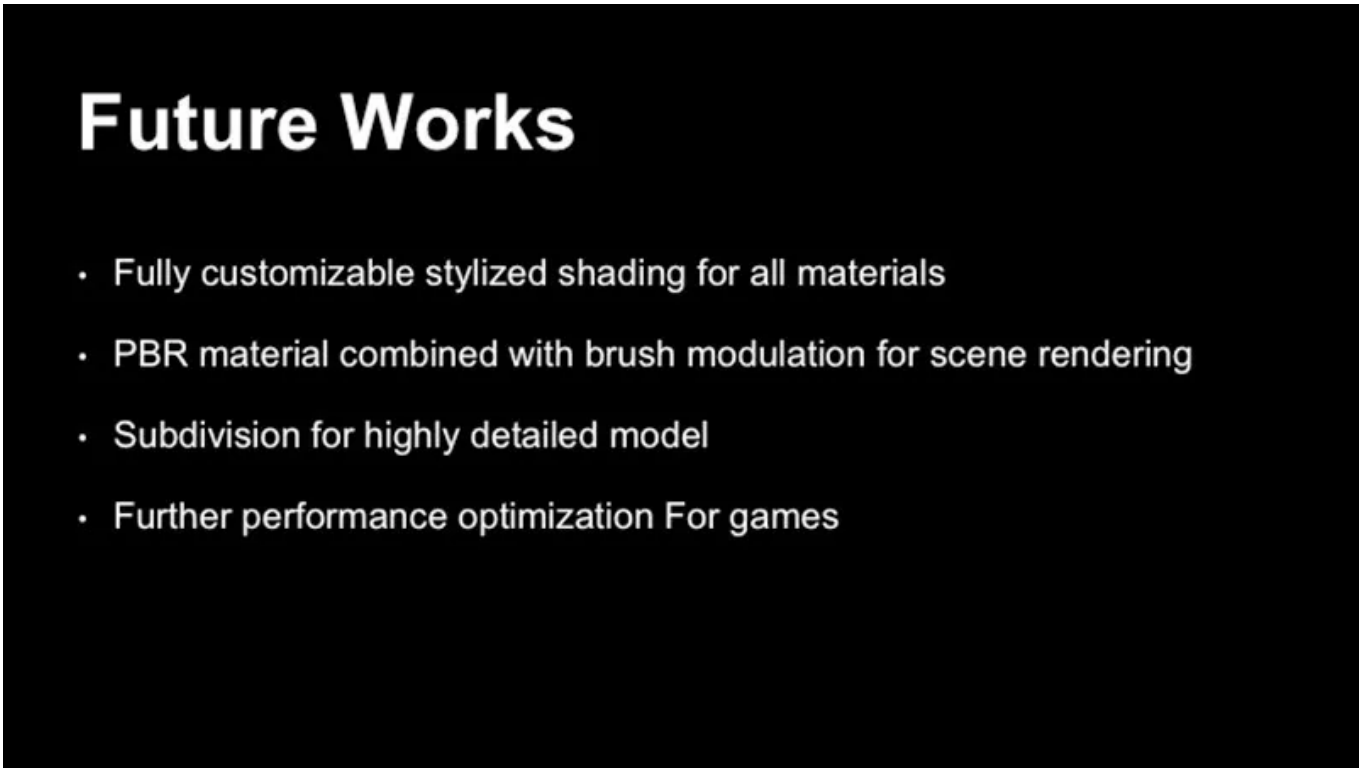


最后，我们来谈谈实时卡通渲染在今后可以继续改进和完善的地方。

首先是实现所有类型材质完全可定制的风格化渲染，目前我们初步在人物皮肤和服装渲染中的应用了笔刷以获得笔触效果。

下一步我们希望将其扩展到整个场景的渲染，比如新海诚式的场景风格，以呈现有着独特且统一的风格化动画风格渲染。另外一点是要进一步提高模型的渲染精度，我们希望可以实时呈现CG级的模型精度。可以尝试使用Geometry shader或预烘培displacement map进行动态自适应的曲面细分，相比直接导入原始高模，它可以极大减少资源导入的开销和提升运行效率。

最后是优化整套流程解决方案，使之更易于实时调整和编辑，进一步提升运行效率以适合在游戏中使用。



好的，以上就是我们今天有关于卡通渲染要分享的主要内容，谢谢大家！

推荐阅读

- 江毅冰 - 《从AAA游戏到实时渲染的动画电影》
- 育碧 - 《Eagle Flight》背后的研发
- 《旅行青蛙》小规模开发的匠心独运
- 2D游戏开发套件指南（上）
- 2D游戏开发套件指南（下）
- 《崩坏3》：在Unity中实现高品质的卡通渲染（上）

点击“阅读原文”访问Unity中文官方论坛

阅读原文