粒子特效作业分析文档

1. **作业说明**

**引擎版本是Unity2018.3.6f1版本，打开可执行文件后按下空格键可播放动画效果，按下空格键可重播。**

1. **实现过程**
2. 冰冻特效

这一部分，通过查阅文档和资料，发现Unity在2018年的更新中提供了可视化编程的开发工具ShaderGraph，帮助实现一些shader效果。经过一番学习后发现其简单易用，界面友善，在对图形学有一定了解的基础上使用起来非常舒适，于是决定使用它进行冰冻特效的开发。

下面我们掠过ShaderGraph的环境配置以及基础，直接进入材质球的制作过程。

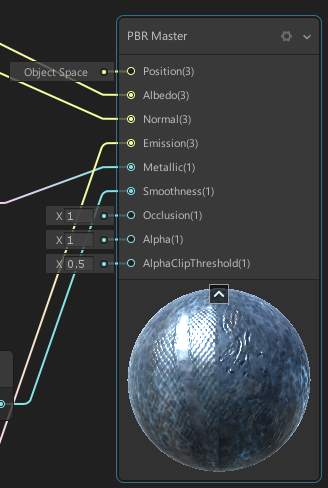


图2-ShaderGraph中的输出窗口

看到最终输出的材质球包含以下几种属性，反射率，凹凸贴图，发光贴图，金属材质贴图，平滑度。首先，最基本的是反射率贴图，其他几样都属于后期效果，所以为了实现模型逐渐冰冻，我们希望看到的是材质球的反射率贴图能够逐渐变成冰块的贴图。

进一步说，我们希望某项变量能够控制反射率贴图中原始贴图和冰块贴图的占比，因为这样一来，只需要将这个变量暴露到外部，交由脚本控制，就能够实现动画效果。

那么如何控制原始贴图和冰块贴图的占比呢，很简单，只需要获得两个贴图的彩色信息，做一个百分比加法即可。ShaderGraph提供了许多常用的数学函数，因此做百分比加分是十分简单的，只需要将这个比例数值交给外部即可。下面给出从外部变量获取材质截图的途径。

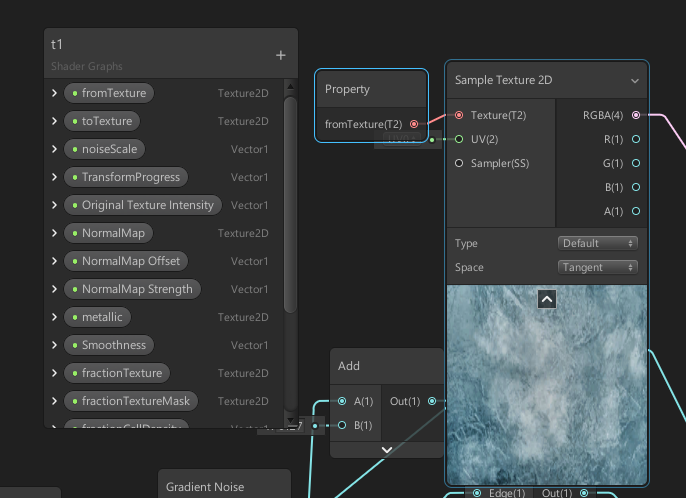


图3-ShaderGraph的变量接口和导入的材质样例

看到，t1面板中是所有我新建的外部变量，其中的fromTexture变量表示原始贴图（实际上这里fromTexture和toTexture一不小心写反了但是实际使用起来是没有关系的，由于更改变量名会导致需要修改许多细节，为了节省时间这里并未更改），将这个贴图交给一个sampleTexture2D函数，输出该材质的的rgba值。

那么是否直接获得两个材质，按指定比例相加就可以了呢？当然可以，但是这样就是简单的透明度变换，我不希望用这种效果，因此我们使用分型噪声来做一个遮罩。

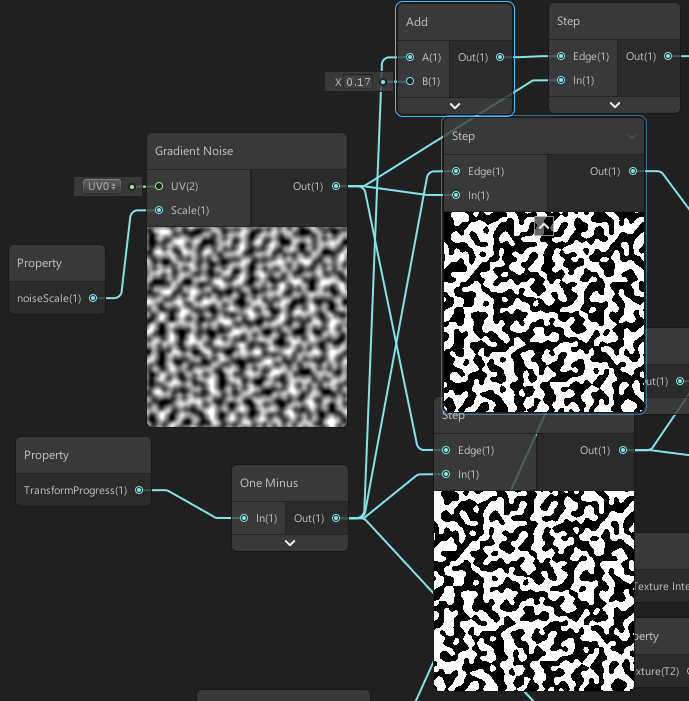


图4-生成一个噪声并使用Step函数

看起来稍微有点复杂，首先生成了一个分型噪声，并用外部变量控制它的缩放比例。左下方，使用一个外部变量TransformProgress用于控制变换的进程，也可以认为是百分比吧。至于它连接的那个OneMinus函数，是用来取对该变量取反，纯粹是为了让这个变量更符合它名字里的逻辑，不用在意。

看到途中有三个step，我们只关注下方两个展开的step，最上方的step与反射率贴图albedo无关。Step函数会对两个输入进行比较，当输入一大于输入二则输出白，反之输出黑。注意这里两个Step输入相同，只是顺序不同，因此输出的两个Step是完全相反的图。

到这里就比较明确了，接下来只需要用图A乘以其中一个Step，用图B乘以另一个Step，其结果相加，就能够得到一张贴图，在这个贴图里，图A和图B各占一部分，且互不重叠。这时候调整TransformProgress参数，就能够改变贴图占据的百分比。

大致如图所示：

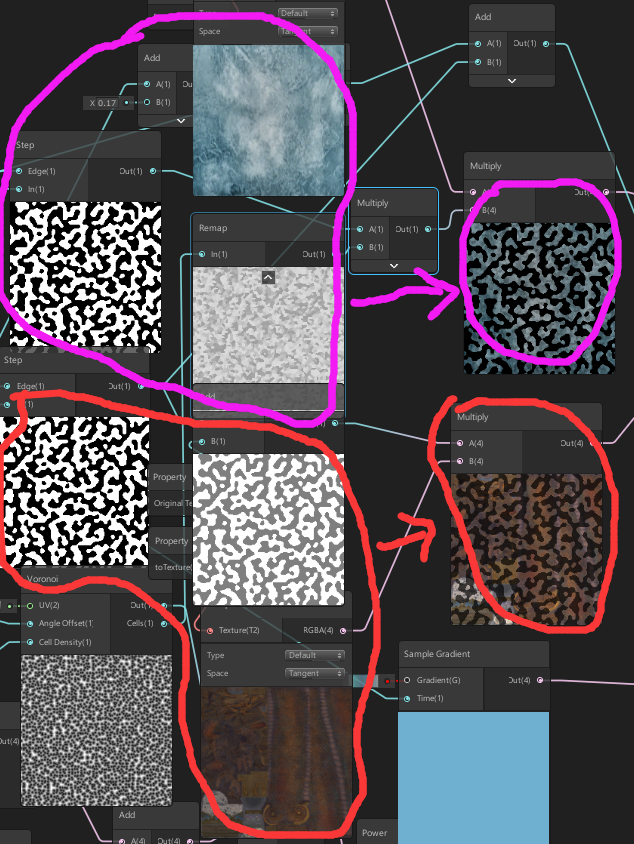
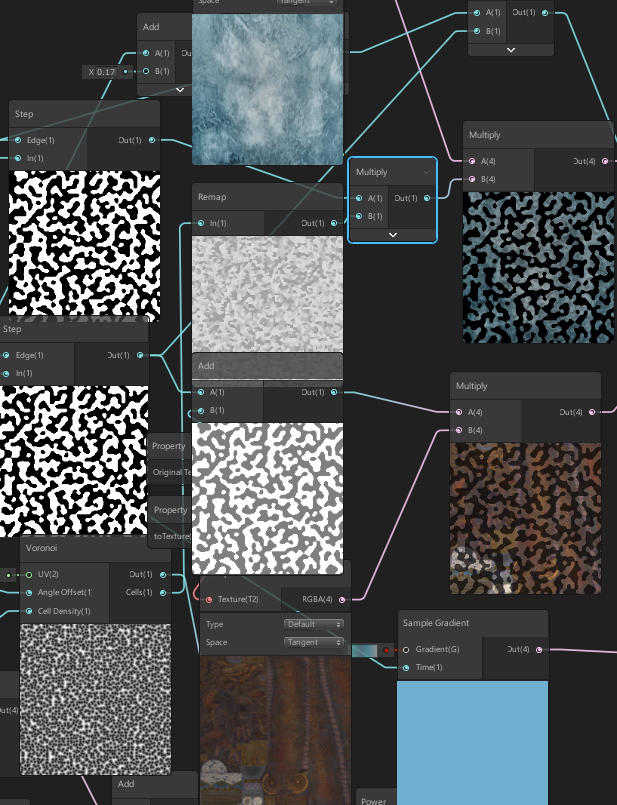


图5/6-混合两张贴图到albedo输出

紫色上方三张图相乘得到右边的贴图，这里包含了一张新的噪声，添加一点点细节。红色下方三张图中，右边的噪声是由左边的噪声增加一个常数得到，这样做的目的是，即使在本不该有原始贴图的区域，也能够得到一些原始贴图的信息，这样，在被冰冻材质包裹的范围内也能隐隐约约看到原始贴图。最后将得到的两个结果直接相加，得到的效果如下所示：

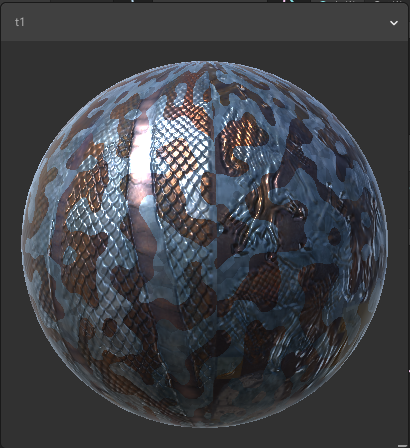
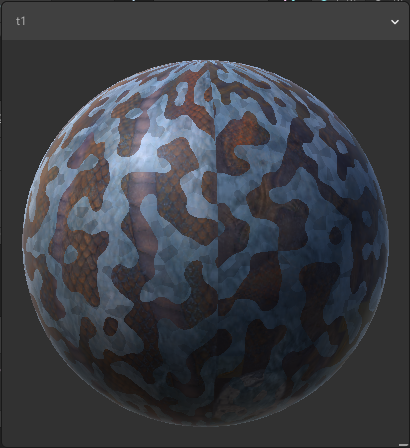


图7-直接得到的贴图 图8-加上metallic和normal贴图后

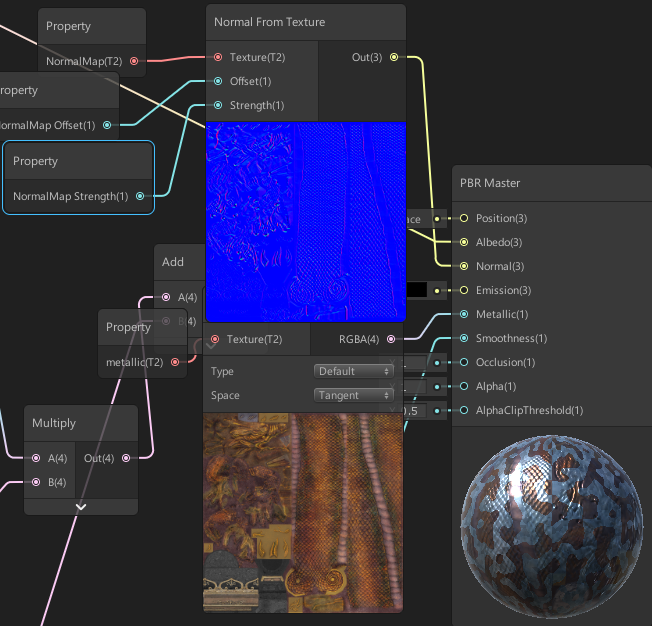


图9-凹凸贴图和金属材质贴图

这里直接加上金属材质贴图和凹凸贴图，得到一个更好的效果，现在我们的材质球就已经初步成型了。已经能够在外部调整参数变量，来得到一个冰冻效果，如下所示。

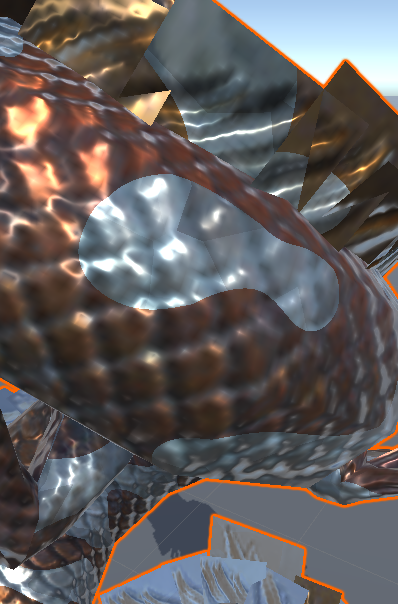
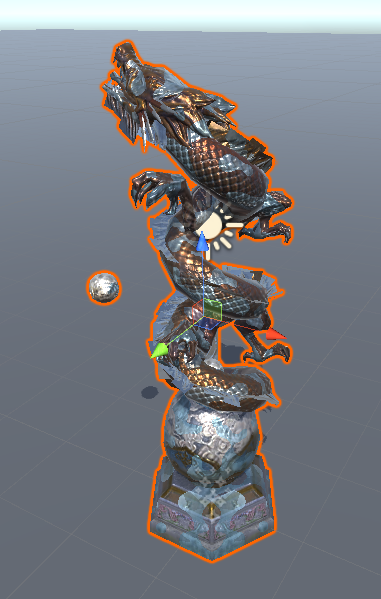


图10/11-Unity内材质表现与细节

但是看起来仍然有点奇怪，边缘过渡太过生硬了，不符合预期效果，为了缓解这种尴尬，我们希望在边缘加一点特殊效果，用来掩盖这锐利的边缘，通知也能够强调一下过渡动画，因为目前看起来这个转变相当不起眼，不知不觉就转换完毕了。也不能说这样不好，但是还是希望效果能被关注到。

因此这里就用到自发光效果了。当然了，自发光时，当然不希望整体都发光，因此需要又需要一个遮罩来罩住需要发光的部分，这个部分自然就是过渡边缘了。为了找到这个边缘，需要再用到一次之前使用的Step。

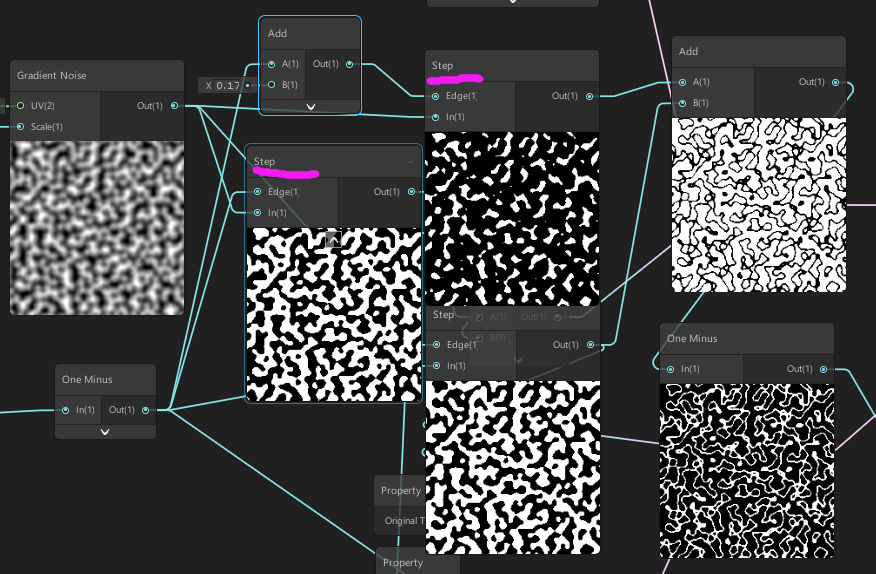


图12-获取边缘蒙版

不难看出，图中被标记的两个Step具有相同的输入，唯一不同的是，第二个Step中的TransformProgress参数被增加了一点点。这样得到的输出和最下方的Step相加，就不是完全严丝合缝的了。就会漏出一部分空间，在这部分空间内像素值为0，其他部分像素值为1，则得到一个边缘遮罩。利用这个遮罩，我们很容易得到发光的贴图：

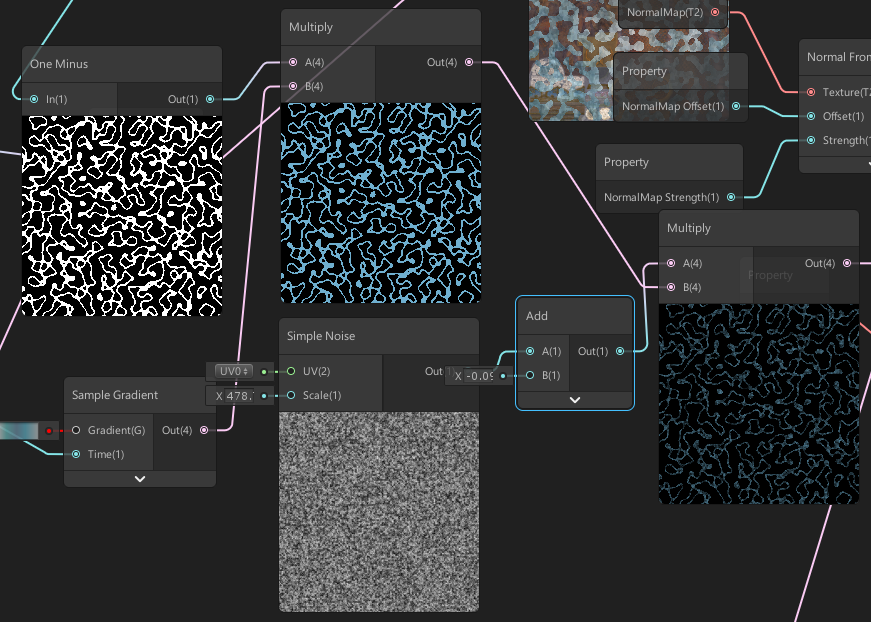


图13-发光边缘制作

这里为了增加一点细节，引入一个噪声。实际上应该可以更换为任意贴图，以产生更绚丽的效果。目前为止变换效果如图：

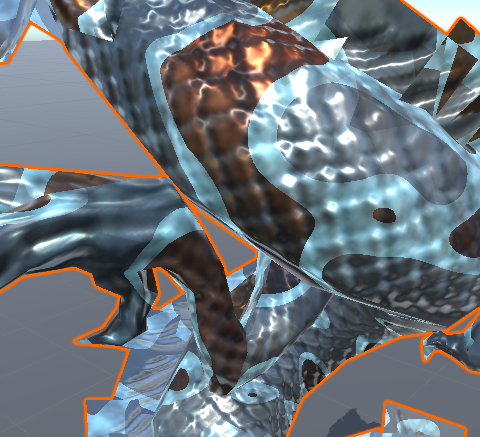
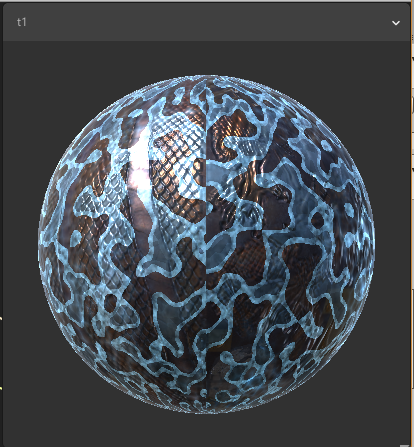


图14/15-边缘发光效果图与细节

仍然有点差强人意，其实希望边缘能够更加平滑，而不是非有既无的断层式变换，这里还需要改进。

现在冰冻特效做好了，那么下一步就是进行碎片的制作。

1. 碎裂效果

这里我的想法是制作一个冰块爆炸的粒子特效，然后在爆炸的粒子能够遮挡住模型的时候，让模型消失。让模型消失只需要一行代码就能够完成，因此制作重点放在碎裂效果上。

制作分为两部分，首先，需要碎块。其次，需要一些烟雾来增加细节。同样为了增加细节，在地面也增加一层粒子，达到霜冻的效果。

首先进入maya制作几个简单的碎块，作为粒子的模型。导入Unity后如图：

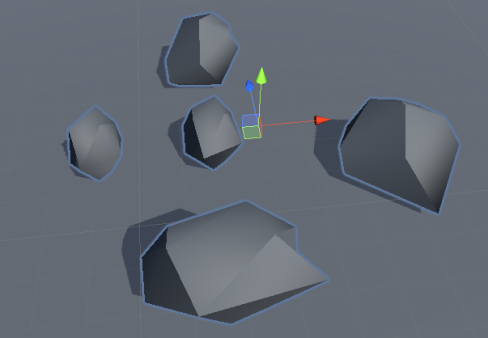


图16/17-maya导入的石块模型以及Unity的场景结构

在Unity中新建四个粒子系统。首先制作stones部分：

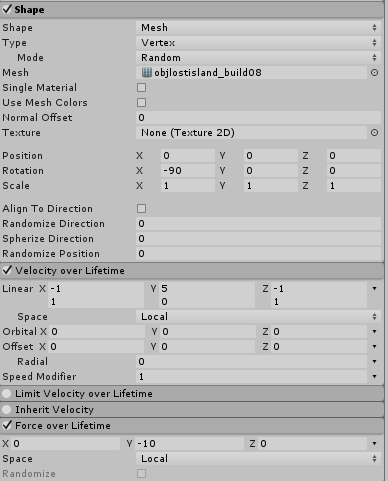
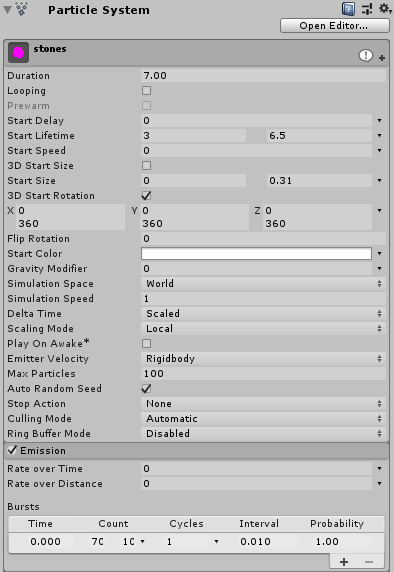


图18/19-stones粒子系统设置

更改粒子时长和生命周期，生命周期在3到6.5之间随机，设置随机大小与旋转，并更改释放模式为爆发模式，即所有粒子一次性出现。粒子发射器的形状使用模型的形状，并调整到跟模型贴合。由于是爆炸，因此给一个初速度范围，并接受一个向下的阻力，可以视为重力。

粒子模型即上述的石头模型，材质即为之前制作的冰冻材质，设置粒子接受碰撞和阴影。当然还需要设置粒子的大小曲线，使之有一个突然从零增大的过程。效果如图：



图20-石块粒子系统截图

按照同样的方式我们可以制作剩下的三个效果，当然，剩下的三个效果并没有使用模型而是使用材质贴图，这里剩下的三个效果使用的实际上是一样的贴图，只是做了不同的变换和着色。给出效果图：

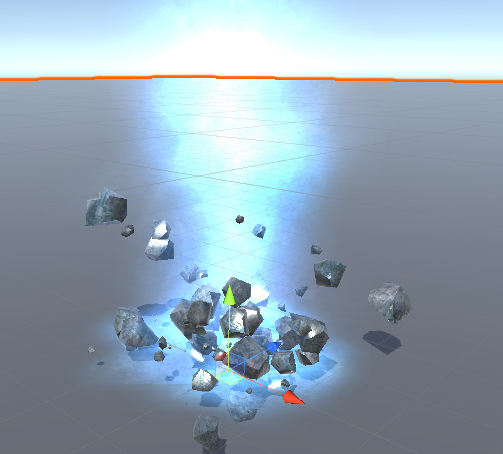


图21-多个粒子系统叠加后的光效

接下来就是使用脚本编辑动画。先附上代码截图。

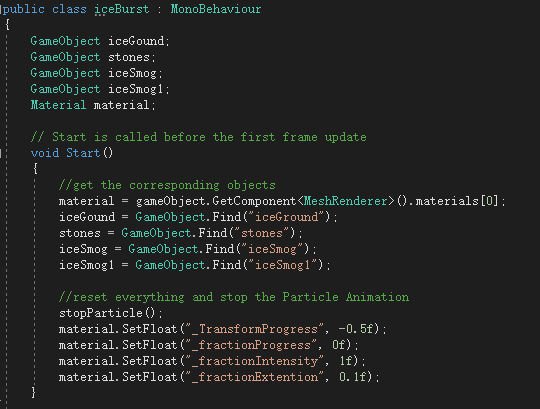


图22-C#脚本Start()部分

在类内定义几个游戏对象，通过其名字找到对应的对象，为了确保不会在游戏一开始的时候就自动播放粒子动画，这里先停止粒子动画。然后对已经做好的材质做一定的初始化，这里主要初始化TransformProgress参数，即贴图置换的进程。另外几个有关fraction的参数是后来添加的，用于初始化冰裂纹，这将会在后文阐述，这里先给出停止和播放动画的函数：

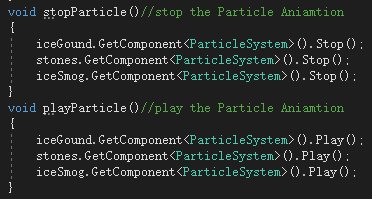


图23-C#脚本自定义函数部分

下面定义一个IEnumarator类型的函数，这里自然是作为协程的迭代器使用了。

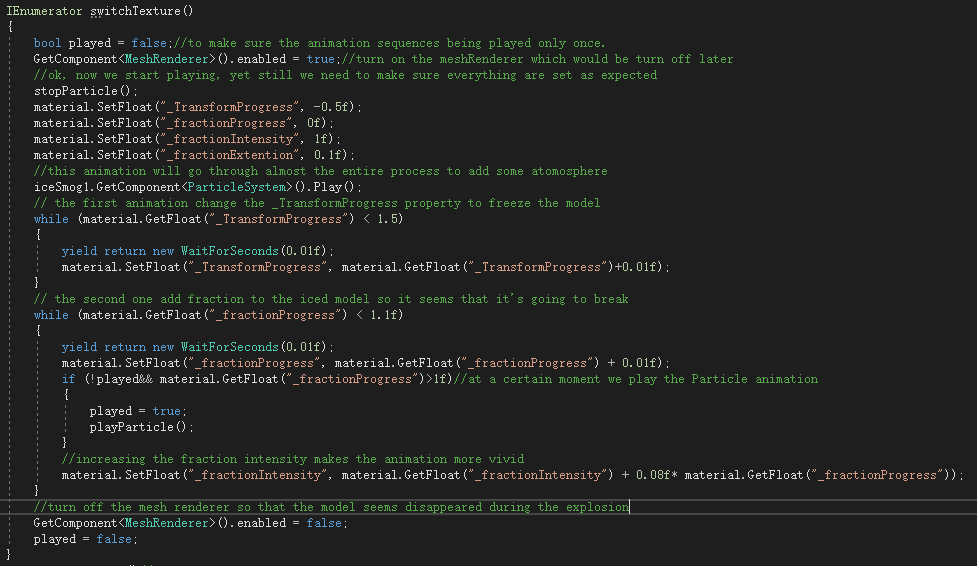


图24-C#脚本协程部分

在进入循环之前先重置一些重要参数，和Start()函数里做的事情是一样的。然后先播放一个动画，这个动画会缓慢地释放冷气，作为模型冰冻时的细节效果。

接下来进入循环，只要材质的转换还没有结束，就不断运行，每次将转换进程推进一点。然后进入第二个循环，这个循环会在模型完全变成冰块后在冰块上添加一些冰裂的效果,这个效果将会在最后介绍。

同时，在这个循环中，我们选择一个合适的时机开始播放动画，这样在循环结束meshRenderer被关闭前动画会播放一会，直到动画的粒子生成遮住模型。值得注意的是这里需要设置动画仅播放一次，以保证效果正确。

1. 添加冰裂纹

在做完上述效果后，实际上已经接近完成了，但是模型变成冰块之后立即爆裂还是有点奇怪，希望再增添一点预兆，让效果更加自然，因此考虑在材质上增加一道冰裂纹。在albedo材质里做冰裂纹觉得似乎有点奇怪，因此决定在emission贴图上做文章。

但是emission上已经有一个边缘特效了，其实也不要紧，由于希望实现的效果是当原材质完全转换成冰之后再裂开，因此冰裂和边缘发光在时间上是不冲突的，即使真的会重叠，两个效果也并不会影响到对方，而只是简单线性叠加而已。因此问题不大。

下面开始介绍冰裂纹效果制作过程，当然最后产出离裂纹的效果还有一定距离，目前个人认为是没有找到合适的噪声类型，当然也可以自行制作一份贴图，但这里为了节省时间，就暂时利用系统生成的噪声。

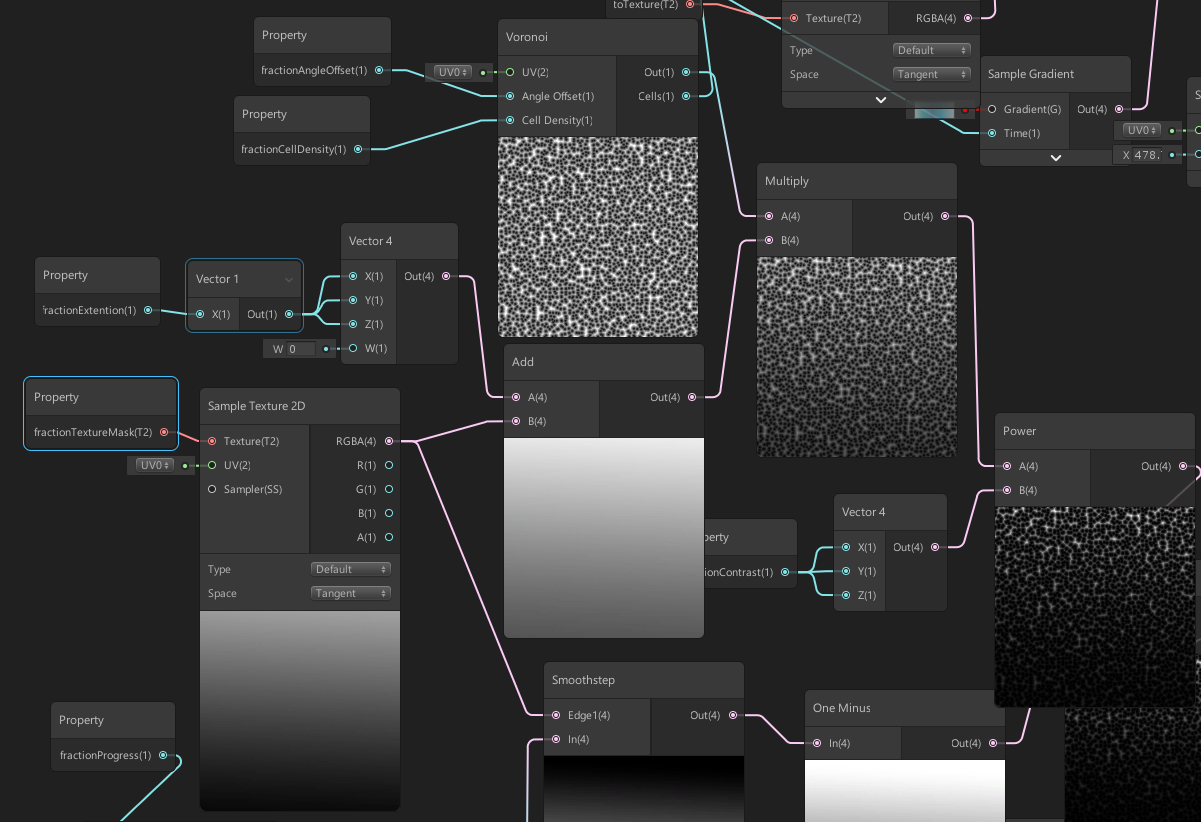


图25-制作冰裂纹蒙版

可以看到我们引入一张黑白贴图作为遮罩，并生成一张Voronoi类型的噪声，将噪声和遮罩进行叠加。当然在这之前遮罩与fractionExtention变量相加，能够对蒙版进行一定程度的扩展。然后对得到的加过蒙版的噪声进行对比度的调整，这里使用一个幂函数作为简单的对比度调节。下面使用一个参数调节蒙版变化，达到动画效果：

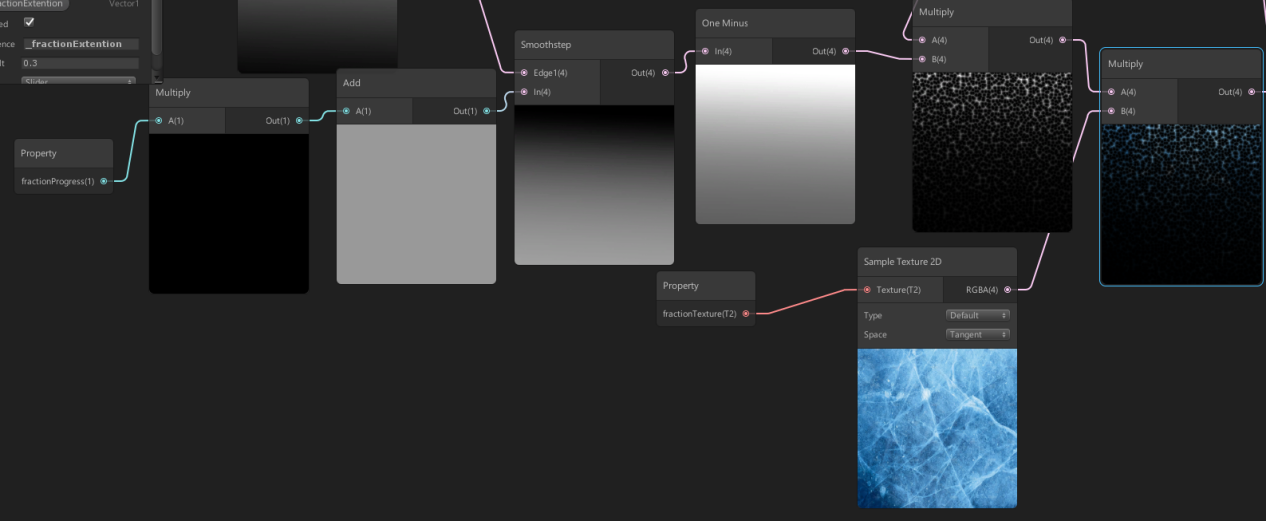


图26-制作冰裂纹

可以看到，这里类似的使用SmoothStep函数进行控制，其效果类似Step函数，但有渐变处理。这里最初的multiply和add函数是为了翻转fractionProgress变量，让它更符合语义，同样不必在意。将SmoothStep的输出翻转之后作用在上面的蒙版上，就得到了一个可变的蒙版，这时将一个新建的材质赋给它，就能够在模型上实现类似冰裂的效果。效果如下：

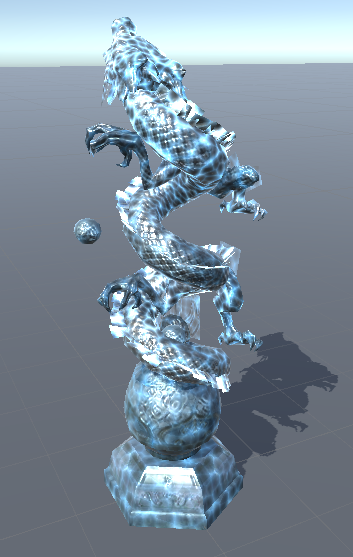


图27-材质最终效果

1. 透明通道处理

会注意到上图的透明材质贴图部分没能正确显示，龙模型身上的鳍连同面片一起显示了出来。这里解决一下这个问题。

进入ShaderGraph，将制作好的Alpha通道贴图赋给Alpha变量，然后调整AlphaClipThreshold参数。如下所示：

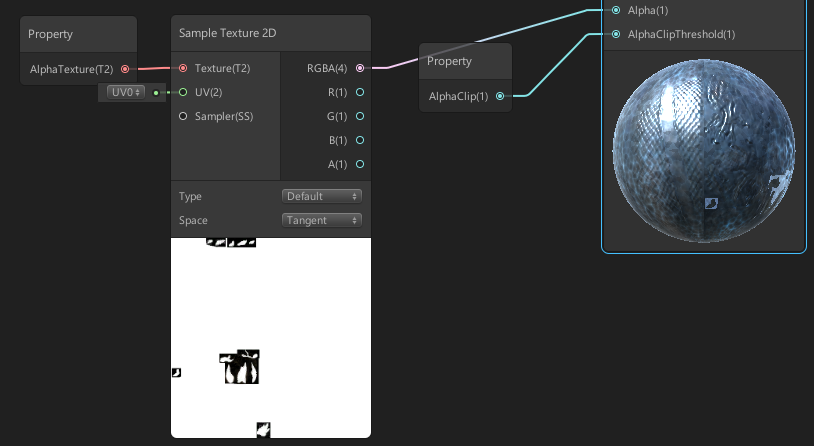


图28-透明通道调整

在Unity中预览效果如下：

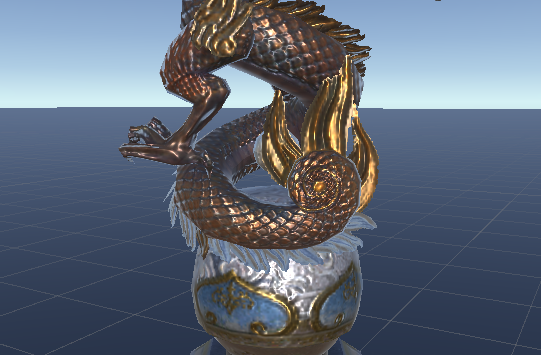


图27-透明通道处理结果

感谢看到这里，这就是最终效果了。具体动态效果需要进入项目或者是运行可执行文件查看。