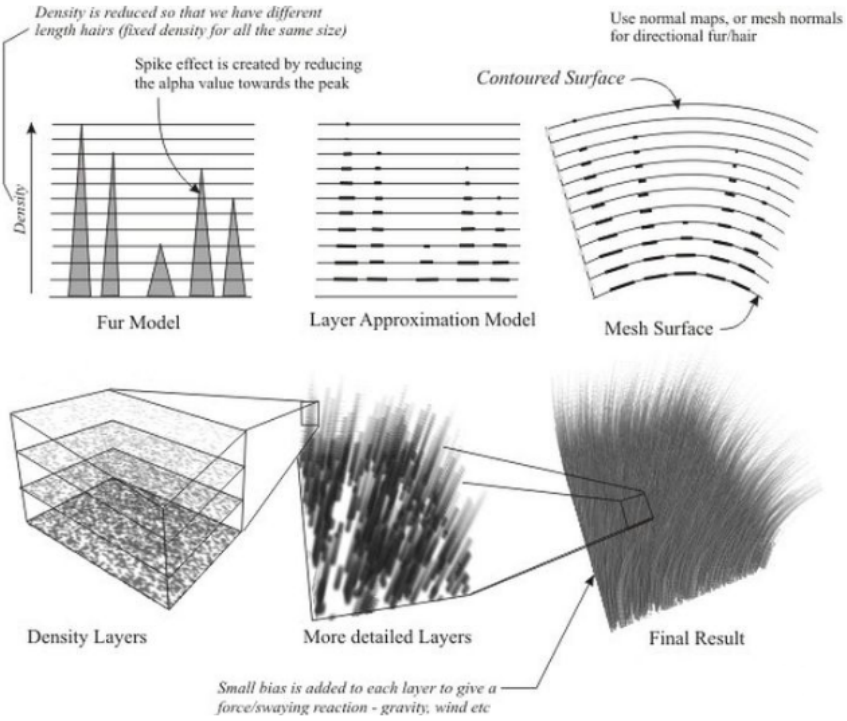
毛发shader

目标：

- 单pass下用shell实现毛发效果。
- 可调节层数，长度等信息。
- 顶点运动使毛发配合物体运动。
- PBR光照。

制作：

1. 理解shell模型： 大致就是单根毛发由多层结构组成，当层数足够多和近的时候看起来就像毛发了。 毛发向法线方向生长，用alpha值（用噪点图来代替）来对每层大小进行递减。



2. 单层Shell实现：

第一步：先实现一个眼法线方向进行顶点外扩。

```
output.positionHCS = TransformObjectToHClip(float4(input.positionOS +
input.normalOS * _SingleShellLen * layerNum ,1));
```

第二步：根据不同的shell级数来调节alpha值，越往外应该越细。第一行，为一个拟合的函数，让尖端更加圆滑。

```
baseColor.a = saturate((noise * 2 - (layerOffset * layerOffset + layerOffset
*noise* 5)) * density);
col.a *= noise;
```

先将noise map采样作为了alpha。

然后根据不同的层数来对alpha进行递减来使尖端的毛看起来更细。但因为还没有实 现多层shell，所以还看不到实际效果。这里的LayerNum代表的是当前层数的offset,之后会通 过脚本传入。最后再乘上刚才的noise map (乘上毛发末端才不会连在一起)。

不关闭深度写入， 然后使用 OneMinusSrcAlpha 混合模式。

3.多层shell实现：

这是最重要的一步，多层的shell可以直接表现出毛发的效果，但因为要在单pass中达 到效果，所以用一个脚本来进行实现。

第一步：对于shader进行简单的设置就可以在材质面板看到开启gpu instancing的选项，不过现在还不够。用GPU instancing是以免之后开启脚本后开销太大。

使用GPU instancing的话只需要消耗一个DC。

```
#pragma multi_compile_instancing
```

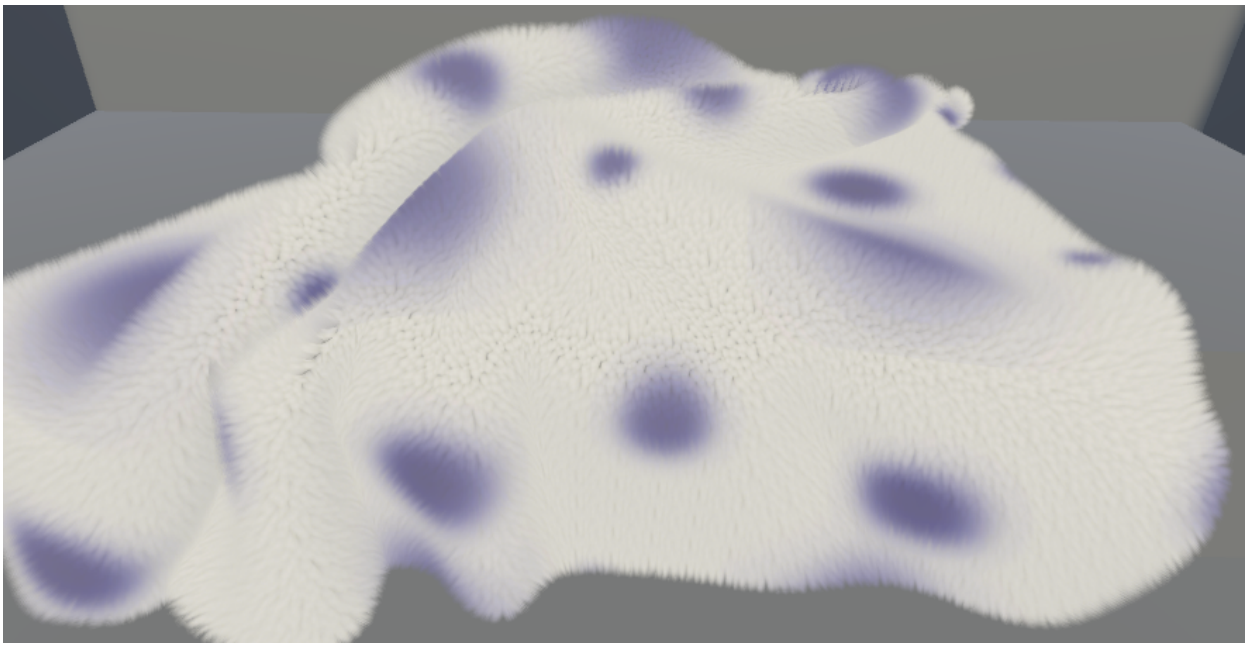
目前大致思路是用脚本生成多个object然后对于不同的object传入不同的_LayerNum 参数，以此来达到多层shell的实现。（并把生成的object变为初始object的子物体）

第二步：将模型变为prefab,用shellLevel来控制层数。生成与层数-1相同的object（prefab）。

```
public Transform prefab;
public int shellLevel = 10;
void Start()
{ for (int i = 0; i < shellLevel; i++)
{ Transform level = Instantiate(prefab);
level.localPosition = transform.position;
 level.SetParent(transform); } }
```

```
MaterialPropertyBlock propertyBlock = new MaterialPropertyBlock();
propertyBlock.SetFloat("_LayerNum", (1.0f / furShellLevels) * i);
 level.GetComponent().SetPropertyBlock(propertyBlock);
```

同时每一个新生成的object传入不同的LayerNum参数，根据i值来个的递增。效果如下：



第三步：在shader中完善gpu instancing。

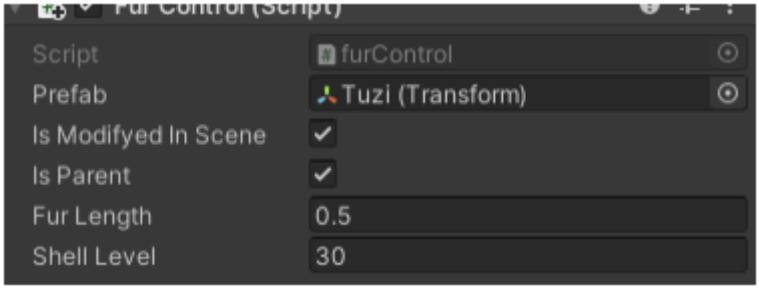
使用unity的宏将需要改变的参数先存入一个buffer。

```
UNITY_INSTANCING_BUFFER_START(InstanceProperties)
UNITY_DEFINE_INSTANCED_PROP(float, _LayerNum)
UNITY_INSTANCING_BUFFER_END(InstanceProperties)
```

并且在之后的代码中用下面这个宏来使用_LayerNum属性。

```
UNITY_ACCESS_INSTANCED_PROP(InstanceProperties,_LayerNum)
```

将之前的SingleShellLen参数也做相同的处理，方便在脚本中一起控制。 调节脚本让其可以实时修改各种参数（目前还不完善）：



第四步：完善shader中的毛发基本效果。

```
col.a = (noise*2-(step *step +step*5)) * _FurDensity;
col.a *= noise;
```

在最后一步乘上noise map之前做一个拟合函数同时用一个新的参数来控制毛发的浓 密程度。也放进脚本一起修改[3]。

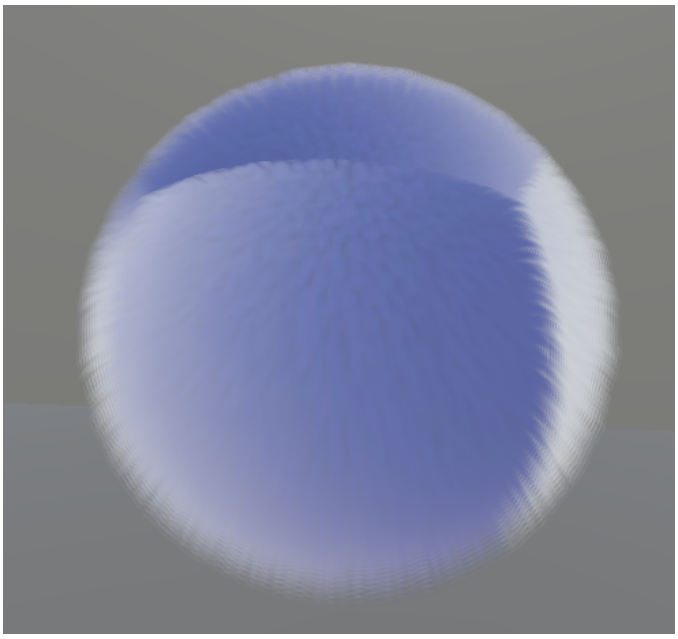
第四步：完善脚本。 之前的脚本实时更改有些问题（同时增加和减少层数会造成渲染顺序错误），于是改 成需要按键来进行生成。利用custom editor，给脚本增加两个按钮，一个用来清除原毛发，一 个用来生成新的毛发。

4.灯光：

毛发的基本效果达成了，再来为毛发配上合适的光照效果。大致分为2部分：环境光， sss效果。

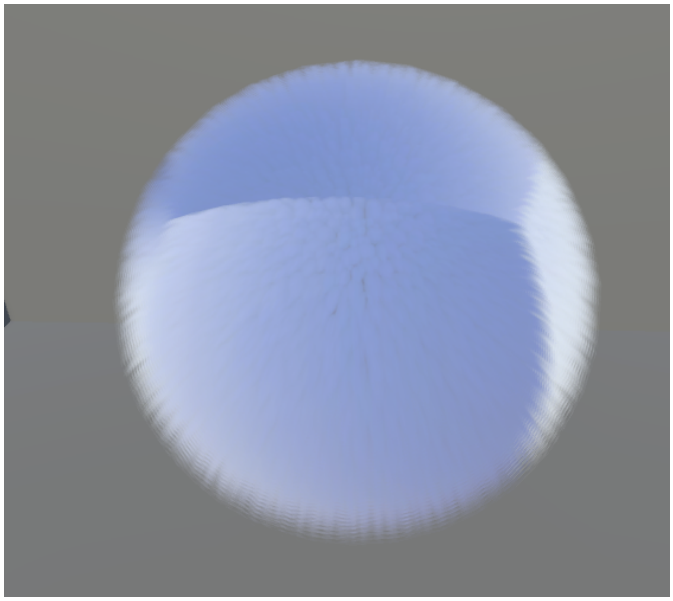
第一步：SSS效果。在背面用类似菲涅尔产生一个简单sss效果。

```
float3 backLightDir = N * _BackSSDistortion + L;
float backSSS = max(0,min(1,1-dot(V,backLightDir)));
float3 sssResult = saturate(backSSS * _SSSIntensity);
```



第二步：环境光。环境光使用SH：

```
//enviroment lighting
float3 ambient_SH = SampleSH(normalWS:float4(N, 1));
float3 iblDiffuse = max(0, 0.03 * baseColor.rgb + ambient_SH);
```

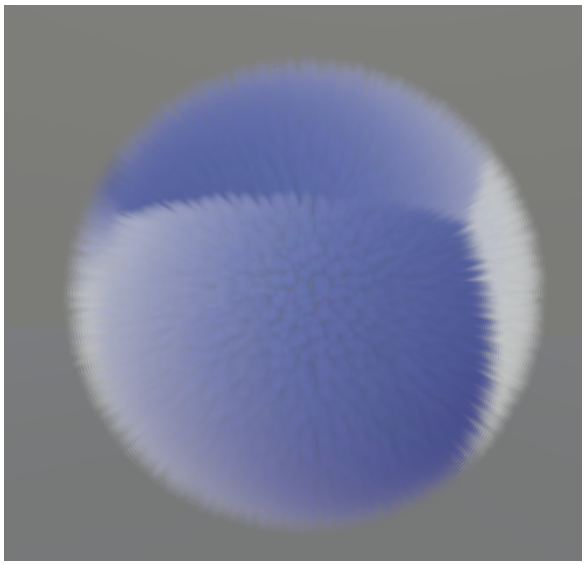


第三步：环境光遮蔽和阴影。设置一个阴影颜色，然后用layerOffset进行线性差值，使毛发根部 更暗，并加上参数设置[3]。同时对环境光进行遮蔽。对所有光照结果啊乘dot(N,L)

```
baseColor.rgb = lerp(_ShdowColor.rgb * baseColor.rgb,baseColor.rgb,layerOffset * _ShadowRange);
```

```
float3 ambient_SH = SampleSH(float4(N, 1));
half Occlusion =pow(layerOffset,2.2);
Occlusion +=0.04 ;
```

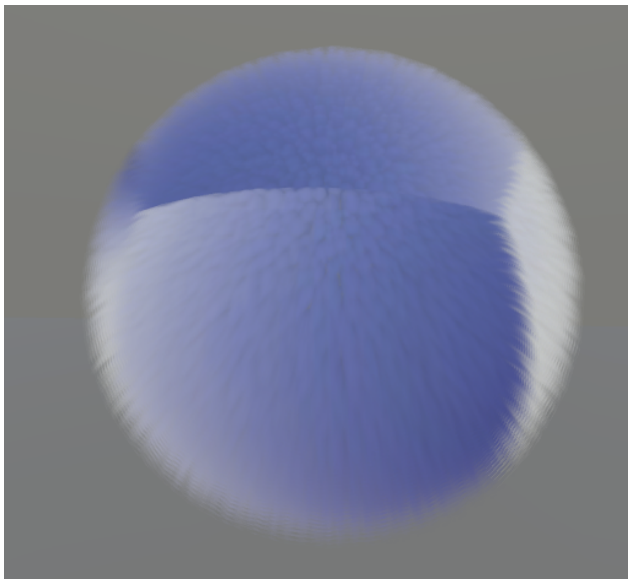
```
half3 SH_result = lerp (shadowColor*ambient_SH,ambient_SH,Occlusion * shadowRange) ;
```



5.毛发运动：

第一步：受重力影响。通过一个受layerOffset影响的曲线来对毛发进行偏移，越尖端受重力影响越大。

```
float3 GetGravity(float layerOffset) {
return pow(layerOffset,3) * mul(unity_ObjectToWorld,float3(0,-1,0)) * _GravityForce;
}
```



第二步：实现物体位移毛发也跟着位移。在shader中申明一个参数_MovementOffset，然后在脚本中实时对这个参数进行更新。大致原理就是在物体运动时，对偏尖端的顶点（pow(layerOffset,3)可实现[3]）进行一个运动反方向的偏移，来实现一个毛发的运动效果。

```
float3 GetMovement(float layerOffset) {
 return pow(layerOffset,3) * mul(unity_ObjectToWorld,_MovementOffset);
}
```

现在shader中对刚才用来实现重力效果的代码进行修改，把固定的重力方向改为由脚 本传参的_MovementOffset。

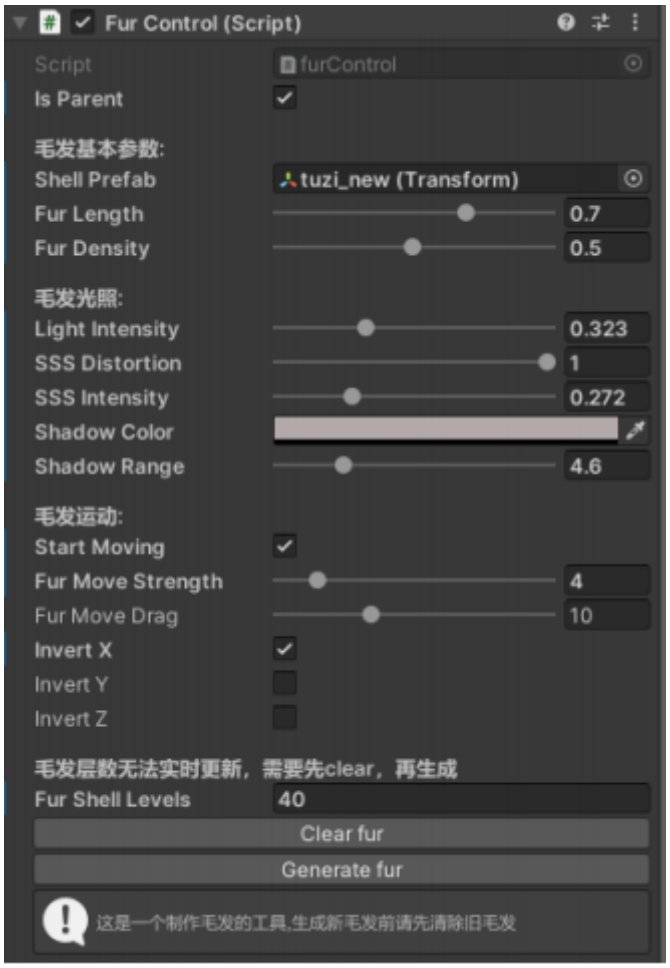
在脚本中需要获得当前物体的运动方向，然后根据运动方向来决定_MovementOffset 并实时传给材质。

给父物体添加一个刚体，通过刚体来得到运动方向。（脚本中有一个简单的运动代码， 方便展示）

```
parentRb.velocity = new Vector3(0.5f * Mathf.Sin(Time.time * 0.5f * Mathf.PI), 0.5f * Mathf.Sin(Time.time * 0.5f * Mathf.PI), 0);
furOffset = parentRb.velocity;
```

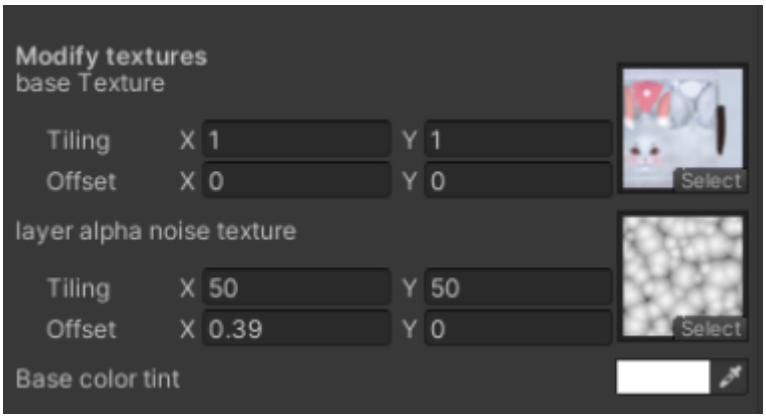
再在脚本中添加可以翻转xyz轴的选项（方便调整在运动时毛的方向）

6.完善： 将材质中的光照相关参数加入脚本，同时将除了 shell层数参数以外的参数改变成在脚 本中实时更新。 脚本截图：



可以看到大部分参数都是使用这个脚本来进行调节。

材质中只需要放好相关贴图：



7.总结

如果在脚本中实时更新毛发的层数（实时删除和生成新的子物体），会造成渲染顺序上 的一些错误，非实时限制毛发层数的修改就解决了这个问题。

参考： 1. An introduction to shell based fur technique https://gim.studio/an-introduction-to-shell-based-fur-technique/
2. Fur effects: https://www.xbdev.net/directx3dx/specialX/Fur/index.php
3.王者荣耀毛发制作：https://mp.weixin.qq.com/s/aIWMEO5Qa2gNn2yCmnHbOg
4.毛尾巴：https://zhuanlan.zhihu.com/p/59483593
5.GPU实例：https://mp.weixin.qq.com/s/qDPfrn2Vtw4qLUpiOBCa8g
6.毛皮材质：https://zhuanlan.zhihu.com/p/57897827