



大家好

欢迎加入LightDir (光向) 研习社
欢迎大家一同探索开源共享的知识分享模式

今日内容

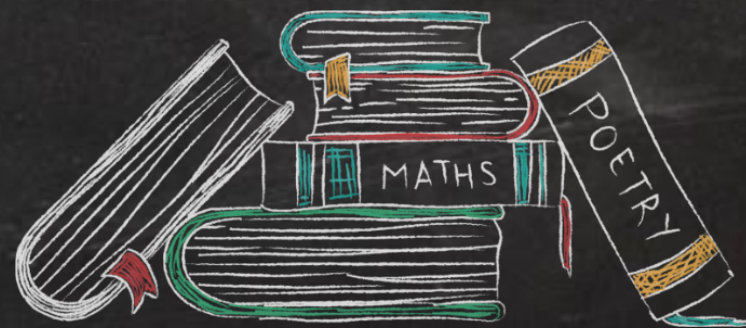
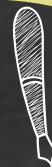


 符文·化神·赛博小人

$$\begin{array}{r} 24 \times 5 \\ \hline 33 \end{array}$$

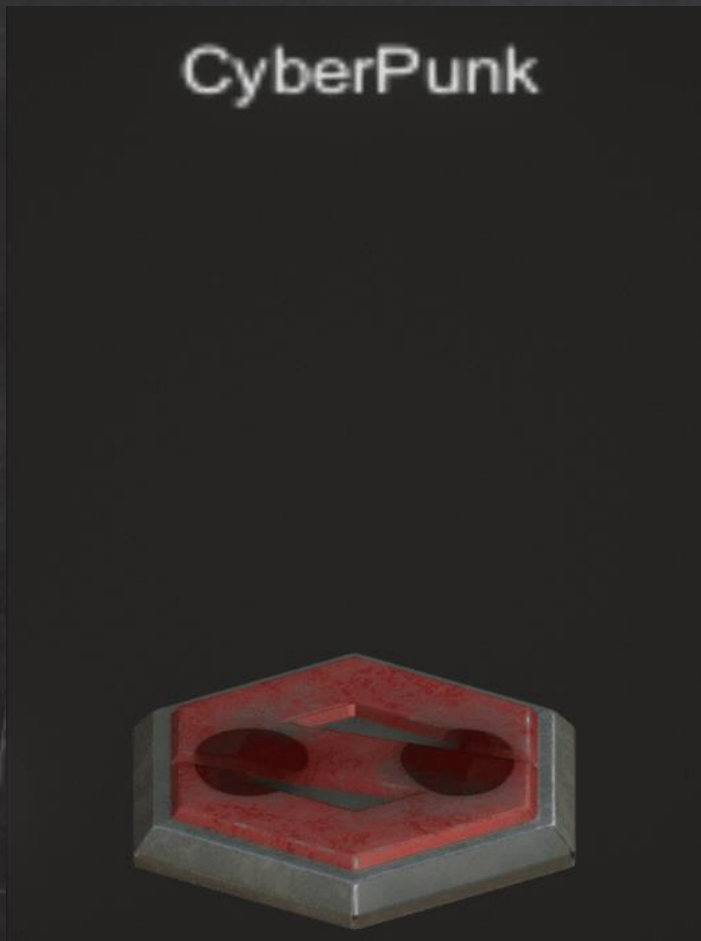
1

符文·化神·赛博小人



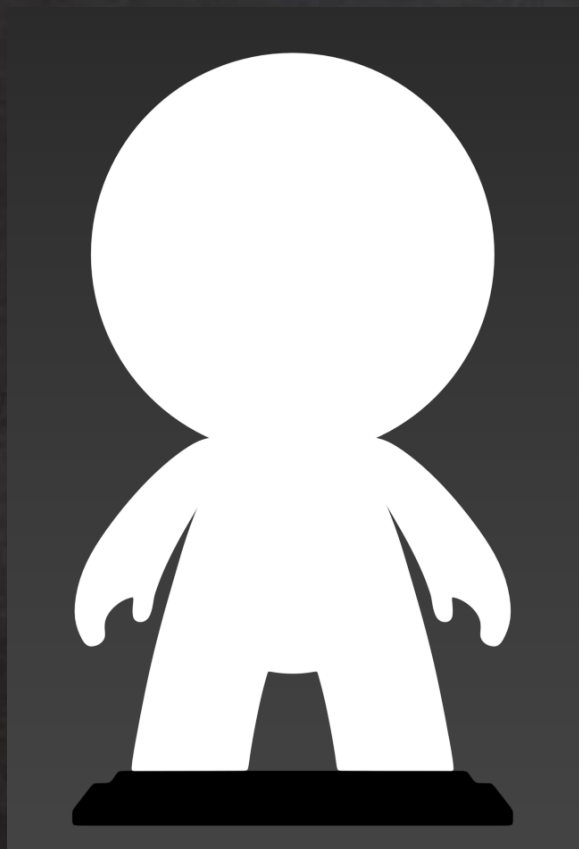
01 案例

LightDir. 光向研习社



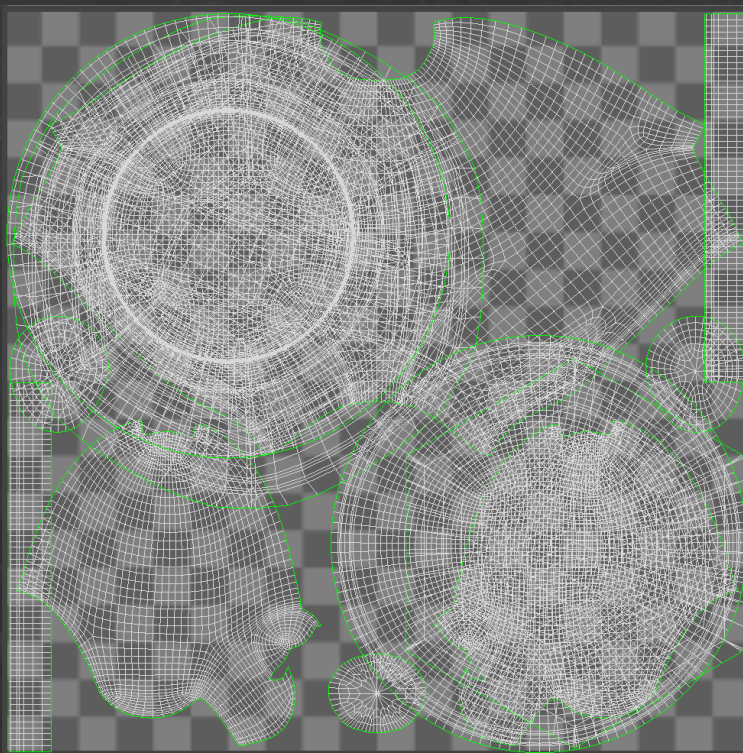
- 在OldSchoolPro的基础上，修改模贴，Shader，制作赛博小人；

02 模型准备

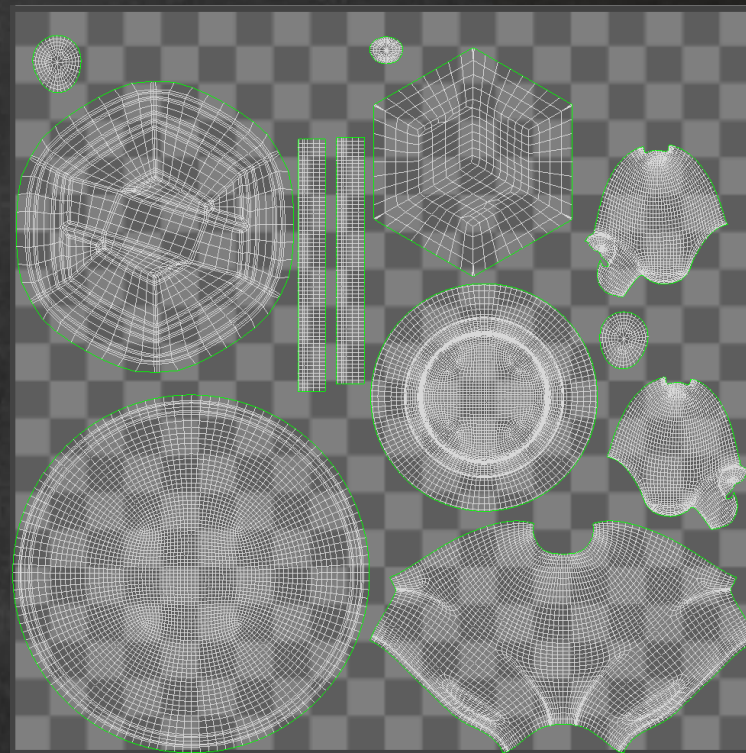


Step1: 为模型追加顶点色, 用于遮罩顶点动画;

- 小人: 白
- 基座: 黑



UV1

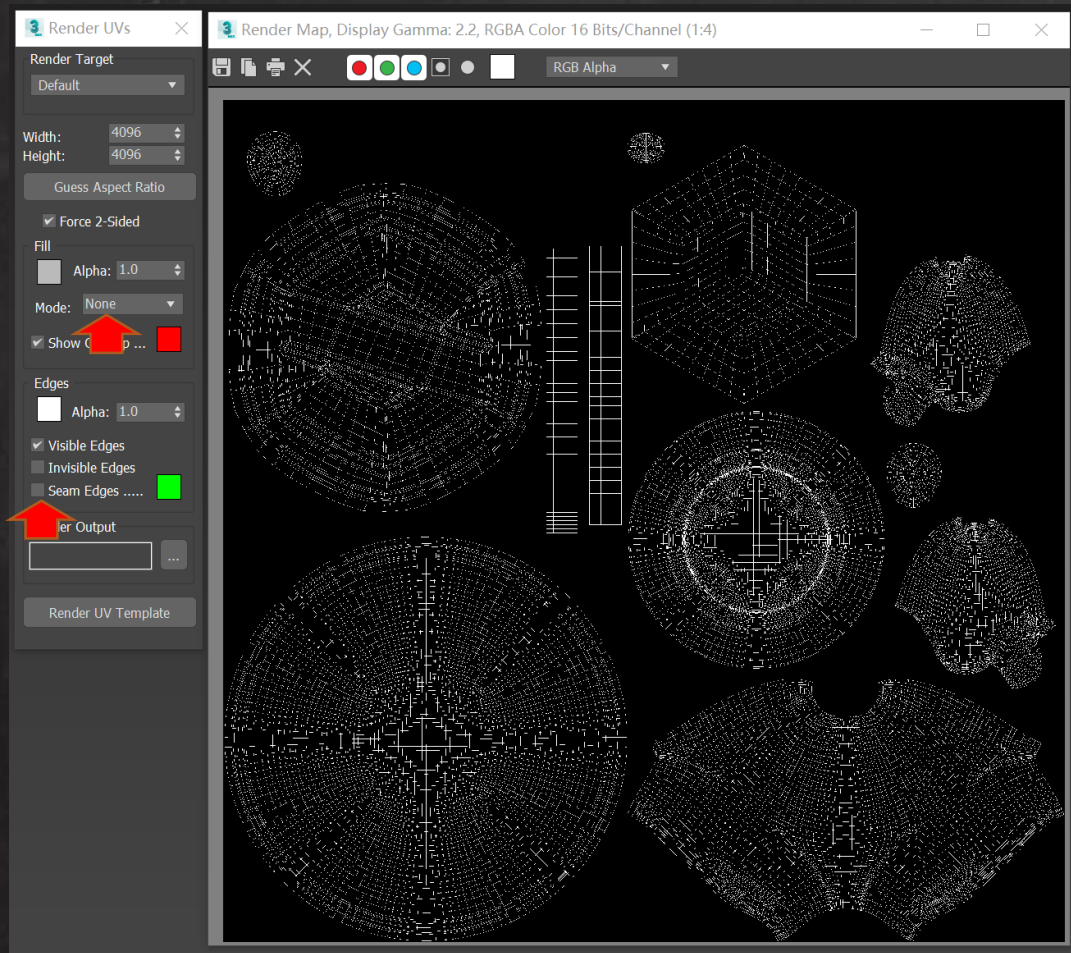


UV2

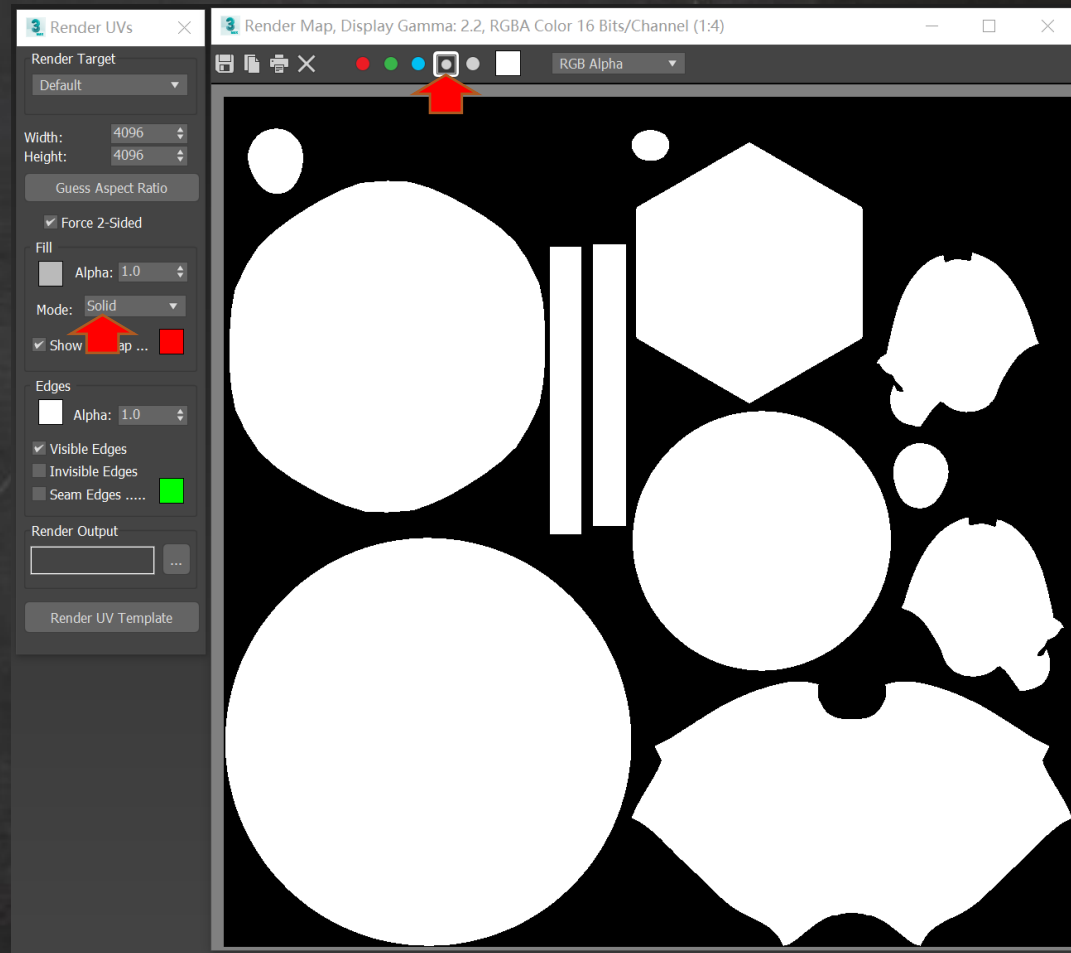
Step2: 为小人追加UV2(引擎中的UV1,Unity编号从0开始):

- 原资产UV1头, 身体, 基座分别展开在一块UV区间; UV2将三部分合并放在同一块UV区间;

03 EffectMap制作



UVMesh



UVIsland

Step1: 3DsMax用Render UVW Template渲染以下UV信息(其他DCC软件渲染UV方法请自行搜索):

- UVMesh: UV网格图;
- UVIsland: UV分块Alpha;

04 EffectMap制作

EXPLORER

L21_Cyberpunk.sbs

Effectmap

Resources

Common MeetMat2c

common_meetmat2c_position [udim]

UVMesh

UVIsland

Open

Return

Copy

Ctrl+C

Remove

Del

Rename

F2

Reload

Ctrl+R

Show in Explorer...

Re-locate...

Bake model Information

Refresh all baked map

Ambient Occlusion

Ambient Occlusion Map from Mesh

Bent Normals Map from Mesh

Color Map from Mesh

Convert UV to SVG

Curvature

Curvature Map from Mesh

Curvature Map from Mesh (deprecated)

Height Map from Mesh

Normal Map from Mesh

Opacity Mask from Mesh

Position

Position Map from Mesh

Thickness Map from Mesh

Transfer d Texture from Mesh

World Space Direction

World Space Normals

Add baker

Delete baker

EXPLORER

L21_Cyberpunk.sbs

Effectmap

Resources

Common MeetMat2c

common_meetmat2c_position [udim]

UVMesh

UVIsland

LIBRARY

3D Perlin Noise

3D Perlin Noise Fractal

3D Simplex Noise

3D Worley Noise

Anisotropic Noise

Blue Noise Fast

BnW Spots 1

BnW Spots 2

Substance Designer

Select elements to bake

Select by: Materials

Material

Faces

Color

Merged_Mesh_01_Head3

7516

Setup High Definition Meshes

Add high definition model

Remove all

Use low definition

Bakers default values

Default size

1024

1024

Default Format

Truevision Targa (*.tga *.targa)

Default Anti Aliasing

Subsampling 8x8

Default UV set

UV 1

Bakers render list

Add baker

Delete baker

Pull to top

Push down

Baker

Size

1024x1024

Anti alias

Avg. normals

UV set

Format

Position

1

tga

Baker Parameters

Mode

All Axis

Axis

1

Normalization Type

BSphere

Normalization Scale

Full Scene

Preset

Save baking setup

Start Render

Close

GPU acceleration: enabled

Bakers default values

Default size

1024

1024

Default Format

Truevision Targa (*.tga *.targa)

Default Anti Aliasing

Subsampling 8x8

Default UV set

UV 1

Position

Step2: 将模型导入SubstanceDesigner, 烘焙Position信息:

05 EffectMap制作

Step3: 制作EffMap01:

1. 资源处理:

1. 将UVMesh反相, 扩边;
2. 分离UVIsland的Alpha;

2. 模型网格Mask:

1. 处理后的UVMesh再反相;

3. 模型面随机灰度Mask:

1. UVMesh>FloodFill;
2. FloodFill>FloodFill2RandomGayscale;
3. Blend处理背景色;

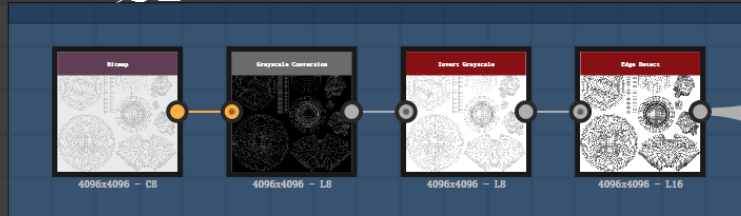
4. 模型面内坡度Mask:

1. Bevel获得坡度;
2. Blend处理背景色;

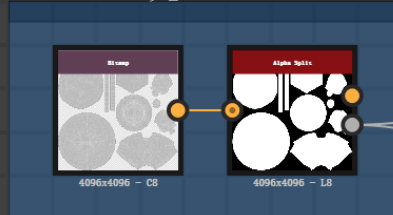
5. 合并3张Mask为一张EffMap01;

6. 高频信息, 导出大图, 教学不考虑性能导出4K; 项目中应将模型网格密度降低, 同时能降低Mask的信息密度, 以控制纹理密度;

UVMesh处理

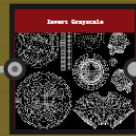


UVIsland处理



EffMap01

网格Mask



4096x4096 - L16

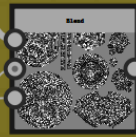
面随机灰度Mask



4096x4096 - C16



4096x4096 - L16



2048x2048 - L8

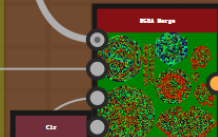
面坡度Mask



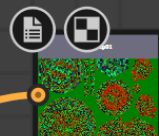
4096x4096 - L16



2048x2048 - L8



4096x4096 - C16

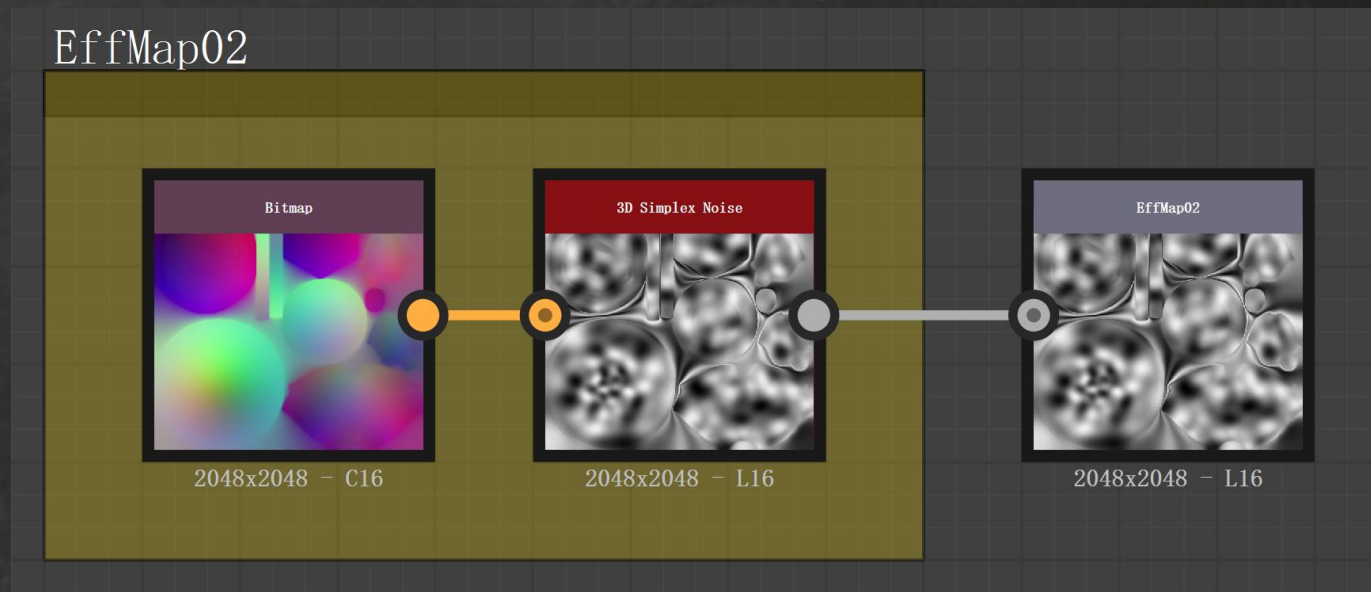


4096x4096 - C16

06 EffectMap制作

Step4: 制作EffMap02:

1. Position>3DSimplexNoise;
2. 低频信息, 导出小图, 256即可;



07 编写Shading

1. 以L11_OldSchoolPro为模板, CtrlCV;
2. 修改路径名;
3. 追加面板参数组Effect, 包含以下参数:

1. _EffMap01: 特效纹理1;
2. _EffMap02: 特效纹理2;
3. _EffCol: 光效颜色(HDR);
4. _EffParams: 特效参数;
 - X: 波密度;
 - Y: 波速度;
 - Z: 混乱度;
 - W: 消散强度;

4. 修改SubShaderTags:

- Queue = Transparent 修改渲染队列
- RenderType = Transparent 修改渲染模式

5. 修改混合方式为AB(不预乘法);

6. 修改cginc文件引用路径;

```
Shader "AP01/L21/CyberPunk" {
    Properties {
        [Header(Texture)]
        _MainTex ("RGB:基础颜色 A:环境遮罩", 2D) = "white" {}
        [Normal] _NormTex ("RGB:法线贴图", 2D) = "bump" {}
        _SpecTex ("RGB:高光颜色 A:高光次幂", 2D) = "gray" {}
        _EmitTex ("RGB:环境贴图", 2d) = "black" {}
        _Cubemap ("RGB:环境贴图", cube) = "_Skybox" {}
        [Header(Diffuse)]
        _MainCol ("基本色", Color) = (0.5, 0.5, 0.5, 1.0)
        _EnvDiffInt ("环境漫反射强度", Range(0, 1)) = 0.2
        _EnvUpCol ("环境天顶颜色", Color) = (1.0, 1.0, 1.0, 1.0)
        _EnvSideCol ("环境水平颜色", Color) = (0.5, 0.5, 0.5, 1.0)
        _EnvDownCol ("环境地表颜色", Color) = (0.0, 0.0, 0.0, 0.0)
        [Header(Specular)]
        [PowerSlider(2)] _SpecPow ("高光次幂", Range(1, 90)) = 30
        _EnvSpecInt ("环境镜面反射强度", Range(0, 5)) = 0.2
        _FresnelPow ("菲涅尔次幂", Range(0, 5)) = 1
        _CubemapMip ("环境球Mip", Range(0, 7)) = 0
        [Header(Emission)]
        [HideInInspect] _EmitInt ("自发光强度", range(1, 10)) = 1
        [Header(Effect)]
        _EffMap01 ("特效纹理1", 2D) = "gray" {}
        _EffMap02 ("特效纹理2", 2D) = "gray" {}
        _EffCol ("光效颜色", color) = (0.0, 0.0, 0.0, 0.0)
        _EffParams ("X:波密度 Y:波速度 Z:混乱度 W:消散强度", vector) = (0.03, 3.0, 0.3, 2.5)
    }
    [HDR]
    SubShader {
        Tags {
            "Queue"="Transparent" // 调整渲染顺序
            "RenderType"="Transparent" // 对应改为Cutout
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }
            Blend One OneMinusSrcAlpha // 修改混合方式One/SrcAlpha OneMinusSrcAlpha
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"
            // 追加投影相关包含文件
            #include "AutoLight.cginc"
            #include "Lighting.cginc"
            #include "../Lesson11/cginc/MyCginc.cginc" // 修改Cginc引用路径
            #pragma multi_compile_fwdbase_fullshadows
            #pragma target 3.0
        }
    }
}
```

08 编写Shading

7. 对应声明输入参数;

8. 输入结构追加:

1. uv1 : TEXCOORD1 UV2信息, 用于采样EffMaps;
2. color : COLOR 顶点色信息, 用于遮罩基座;

9. 输出结构追加:

1. uv1 : UV2信息, 用于采样EffMaps;
2. effectMask : 特效遮罩信息;

```
// 输入参数
// Texture
uniform sampler2D _MainTex; uniform float4 _MainTex_ST;
uniform sampler2D _NormTex;
uniform sampler2D _SpecTex;
uniform sampler2D _EmitTex;
uniform samplerCUBE _Cubemap;
// Diffuse
uniform float3 _MainCol;
uniform float _EnvDiffInt;
uniform float3 _EnvUpCol;
uniform float3 _EnvSideCol;
uniform float3 _EnvDownCol;
// Specular
uniform float _SpecPow;
uniform float _FresnelPow;
uniform float _EnvSpecInt;
uniform float _CubemapMip;
// Emission
uniform float _EmitInt;
// Effect
uniform sampler2D _EffMap01;
uniform sampler2D _EffMap02;
uniform float3 _EffCol;
uniform float4 _EffParams;
// 输入结构
struct VertexInput {
    float4 vertex : POSITION; // 顶点信息 Get✓
    float2 uv0 : TEXCOORD0; // UV信息 Get✓
    8-1 float2 uv1 : TEXCOORD1; // UV信息 Get✓
    float4 normal : NORMAL; // 法线信息 Get✓
    float4 tangent : TANGENT; // 切线信息 Get✓
    8-2 float4 color : COLOR; // 追加顶点色信息
};
// 输出结构
struct VertexOutput {
    float4 pos : SV_POSITION; // 屏幕顶点位置
    float2 uv0 : TEXCOORD0; // UV0
    9-1 float2 uv1 : TEXCOORD1; // UV0
    float4 posWS : TEXCOORD2; // 世界空间顶点位置
    float3 nDirWS : TEXCOORD3; // 世界空间法线方向
    float3 tDirWS : TEXCOORD4; // 世界空间切线方向
    float3 bDirWS : TEXCOORD5; // 世界空间副切线方向
    9-2 float4 effectMask : TEXCOORD6; // 追加effectMask输出
    LIGHTING_COORDS(7, 8) // 投影相关
};
```


09 编写Shading

10. 追加赛格小人顶点动画方法:

float4 CyberpunkAnim(float noise, float mask, float3 normal, inout float3 vertex)

- 输入:
 - noise: 偏移噪声;
 - mask: 基座遮罩;
 - normal: 模型法线;
 - vertex: 模型顶点位置(inout);
- 输出:
 - effectMask: float4:
 - X: 波形遮罩-小;
 - Y: 波形遮罩-中;
 - Z: 波形遮罩-大;
 - W: 基座遮罩;
- 过程:
 1. 生成锯齿波;
 2. noise扰动锯齿波;
 3. smoothstep重映射不同波形, 同其他信息打包输出;
 4. 实现顶点动画;

11. 顶点Shader修改:

1. 采样EffMap02获得noise
 - VS采样纹理方法: `tex2Dlod(sampler2D, float4);`
2. 用动画方法对模型顶点预处理, 同时返回遮罩信息;
3. 传递UV2;

```
// 动画方法 inout顶点信息 返回effect相关遮罩
10 float4 CyberpunkAnim(float noise, float mask, float3 normal, inout float3 vertex) {
    // 生成锯齿波Mask
    10-1 float baseMask = abs(frac(vertex.y * _EffParams.x - _Time.x * _EffParams.y) - 0.5) * 2.0;
    baseMask = min(1.0, baseMask * 2.0);
    // 用Noise偏移锯齿波
    10-2 baseMask += (noise - 0.5) * _EffParams.z;
    // SmoothStep出各级Mask
    float4 effectMask = float4(0.0, 0.0, 0.0, 0.0);
    10-3 effectMask.x = smoothstep(0.0, 0.9, baseMask);
    effectMask.y = smoothstep(0.2, 0.7, baseMask);
    effectMask.z = smoothstep(0.4, 0.5, baseMask);
    // 将顶点色遮罩存入EffectMask
    effectMask.w = mask;
    // 计算顶点动画
    10-4 vertex.xz += normal.xz * (1.0 - effectMask.y) * _EffParams.w * mask;
    // 返回EffectMask
    return effectMask;
}
// 输入结构>>>顶点Shader>>>输出结构
VertexOutput vert (VertexInput v) {
    // 采样纹理
    11-1 float noise = tex2Dlod(_EffMap02, float4(v.uv1, 0.0, 0.0)).r;
    // 输出结构
    VertexOutput o = (VertexOutput)0;
    // 计算顶点动画 同时获取EffectMask
    11-2 o.effectMask = CyberpunkAnim(noise, v.color.r, v.normal.xyz, v.vertex.xyz);
    o.pos = UnityObjectToClipPos(v.vertex); // 顶点位置 OS>CS
    o.uv0 = v.uv0 * _MainTex_ST.xy + _MainTex_ST.zw; // 传递UV
    11-3 o.uv1 = v.uv1;
    o.posWS = mul(unity_ObjectToWorld, v.vertex); // 顶点位置 OS>WS
    o.nDirWS = UnityObjectToWorldNormal(v.normal); // 法线方向 OS>WS
    o.tDirWS = normalize(mul(unity_ObjectToWorld, float4(v.tangent.xyz, 0.0)).xyz); // 切线方向 OS>WS
    o.bDirWS = normalize(cross(o.nDirWS, o.tDirWS) * v.tangent.w); // 副切线方向
    // 投影相关
    TRANSFER_VERTEX_TO_FRAGMENT(o)
    // 返回输出结构
    return o;
}
```

10 编写Shading

12. 像素Shader修改:

1. 光照模型部分无需改动;
2. 采样EffMap01, 获得各种遮罩:
 - meshMask: 模型网格遮罩;
 - faceRandomMask: 模型面随机灰度遮罩;
 - faceSlopeMask: 模型面坡度遮罩;
3. 获取VertexOutput信息effectMask:
 - effectMask.x: 波形遮罩·小;
 - effectMask.y: 波形遮罩·中;
 - effectMask.z: 波形遮罩·大;
 - effectMask.w: 基座遮罩;
4. 计算Opacity透明度:
 1. 计算中范围方框消散透明度;
 2. 计算大范围坡度消散透明度;
 3. 混合两种透明度;
5. 计算自发光:
 1. 基于遮罩计算自发光强度;
 2. 叠加特效自发光;
6. 混合光照;
7. 预乘输出;

```
// 输出结构>>>像素
float4 frag(VertexOutput i) : COLOR {
    // 排版考虑 省略相同代码 实际代码省略将不可运行!

12-1    .....
        // 特效部分
        // 采样EffMap02
        float3 _EffMap01_var = tex2D(_EffMap01, i.uv1).xyz;
12-2    float meshMask = _EffMap01_var.x;
        float faceRandomMask = _EffMap01_var.y;
        float faceSlopeMask = _EffMap01_var.z;
        // 获取EffectMask
        float smallMask = i.effectMask.x;
12-3    float midMask = i.effectMask.y;
        float bigMask = i.effectMask.z;
        float baseMask = i.effectMask.w;
        // 计算Opacity
12-4    float midOpacity = saturate(floor(min(faceRandomMask, 0.999999) + midMask));
        float bigOpacity = saturate(floor(min(faceSlopeMask, 0.999999) + bigMask));
        float opacity = lerp(1.0, min(bigOpacity, midOpacity), baseMask);
        // 叠加自发光
12-5    float meshEmitInt = (bigMask - smallMask) * meshMask;
        meshEmitInt = meshEmitInt * meshEmitInt;
        emission += _EffCol * meshEmitInt * baseMask;
        // 返回结果
12-6    float3 finalRGB = dirLighting + envLighting + emission;
12-7    return float4(finalRGB * opacity, opacity);
    }
    ENDCG
}

FallBack "Diffuse"
}
```

11 知识点

1. smoothstep重映射波形;
2. tex2Dlod顶点Shader中采样纹理;
 - 也可采样Mipmap;

Parameters

samp

Sampler to lookup.

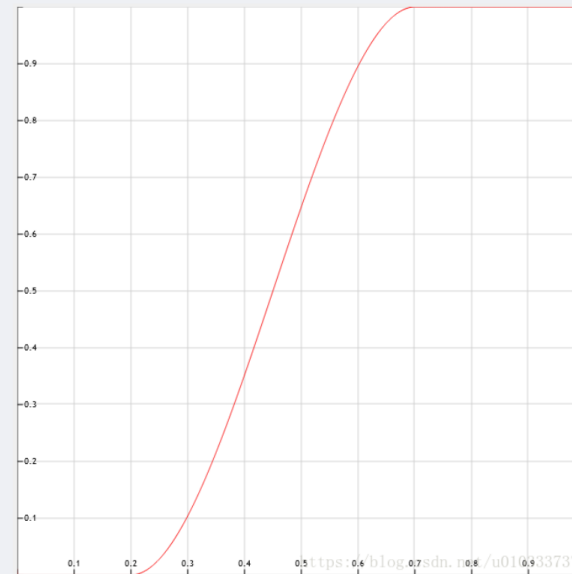
s.xy

Coordinates to perform the lookup.

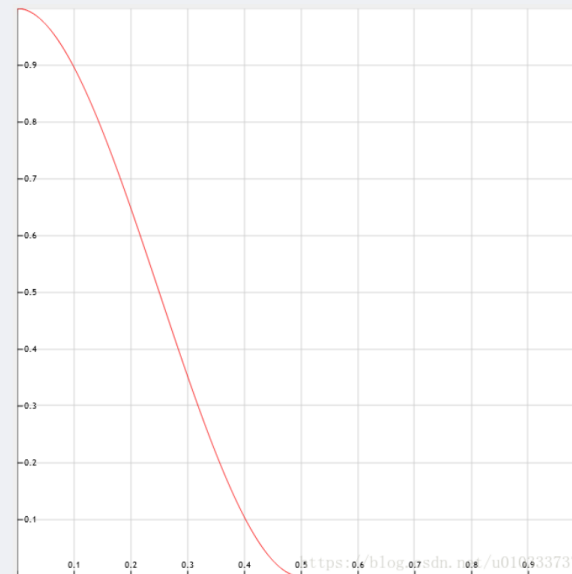
s.w

Level of detail.

smoothstep(0.2, 0.7, x)



smoothstep(0.5, 0, x)





Thanks