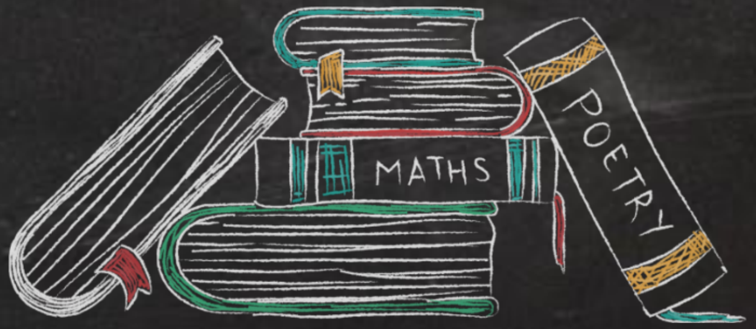
大家好

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

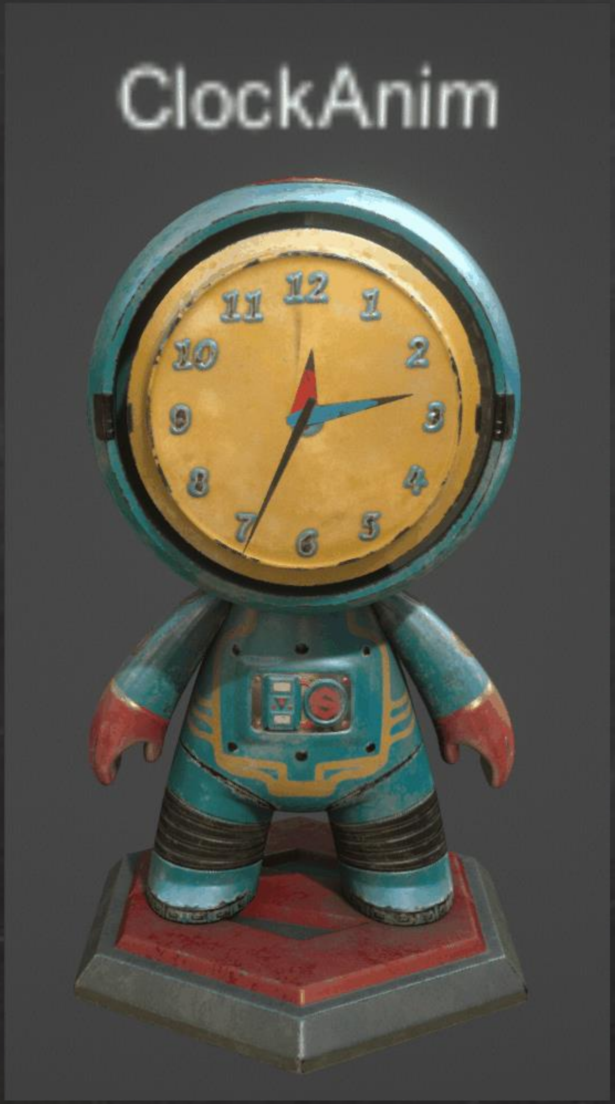
今日内容

- 符文·化神·钟表小人
- 符文·筑基·时间同步

1 符文·化神·钟表小人

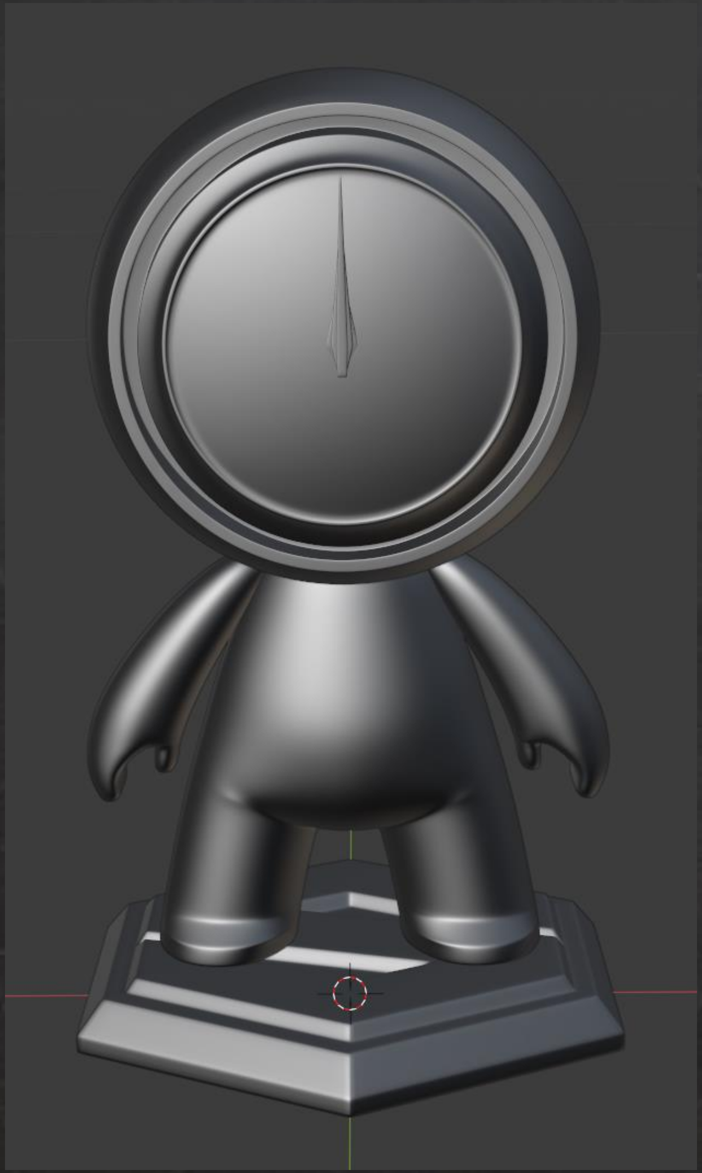


01 案例



- 在OldSchoolPro的基础上，修改模贴，Shader，制作钟表头小人；

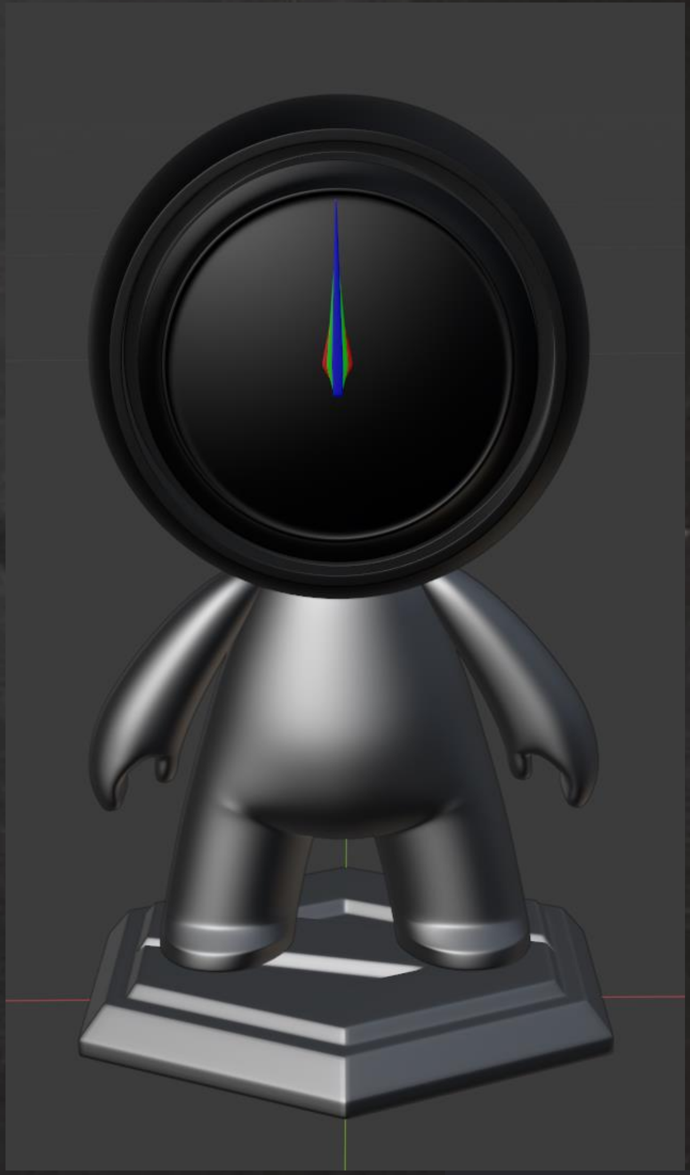
02 模型准备



Step1：为模型追加指针；

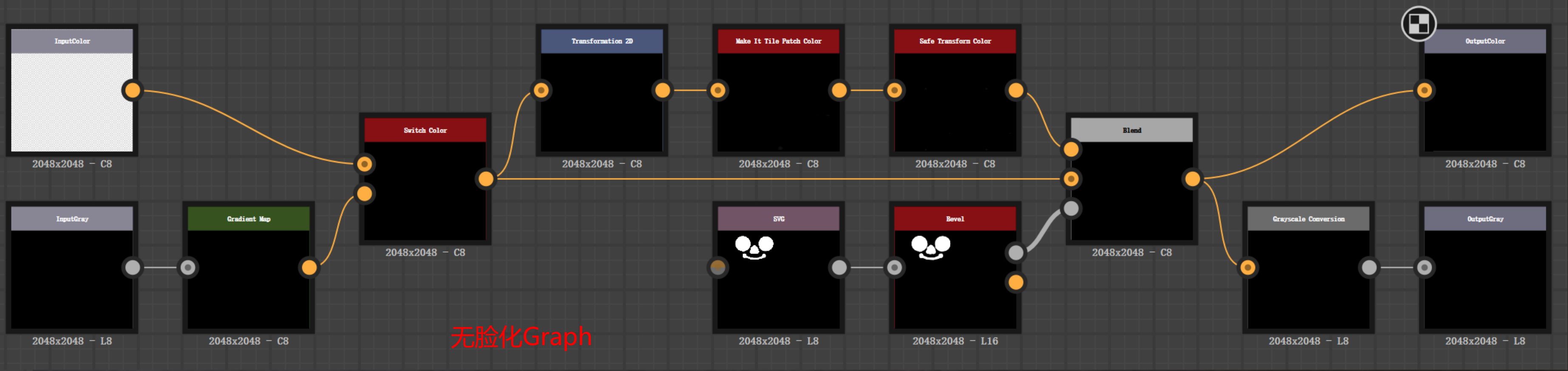
Step2：为小人头部绘制顶点色；

- 时针：红；
- 分钟：绿；
- 秒针：蓝；
- 其余：黑。

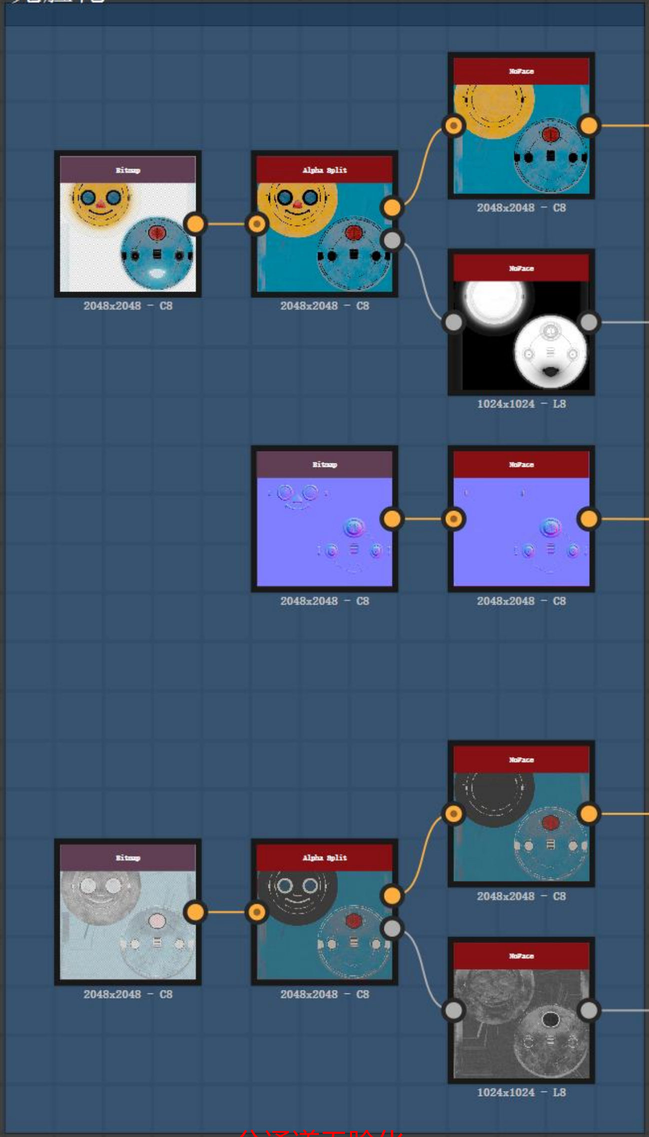


03 SD处理贴图

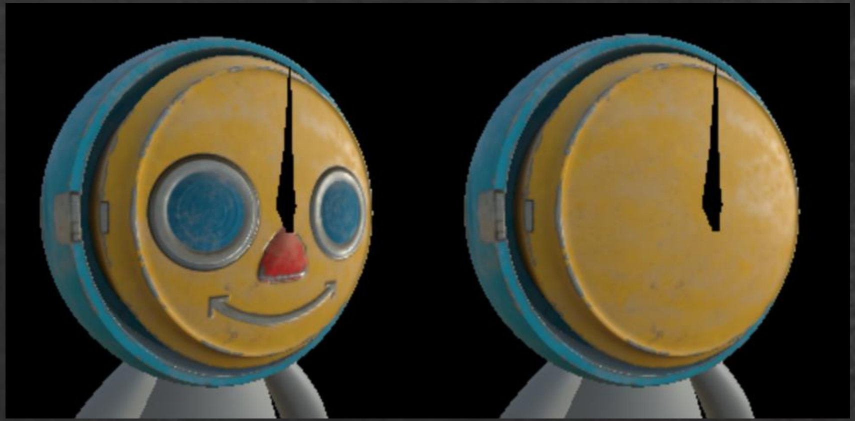
无脸化



无脸化Graph



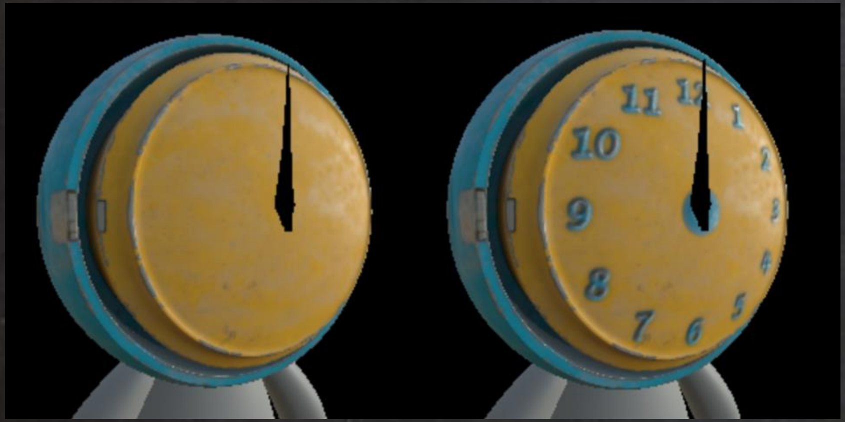
分通道无脸化



Step1：小人无脸化；

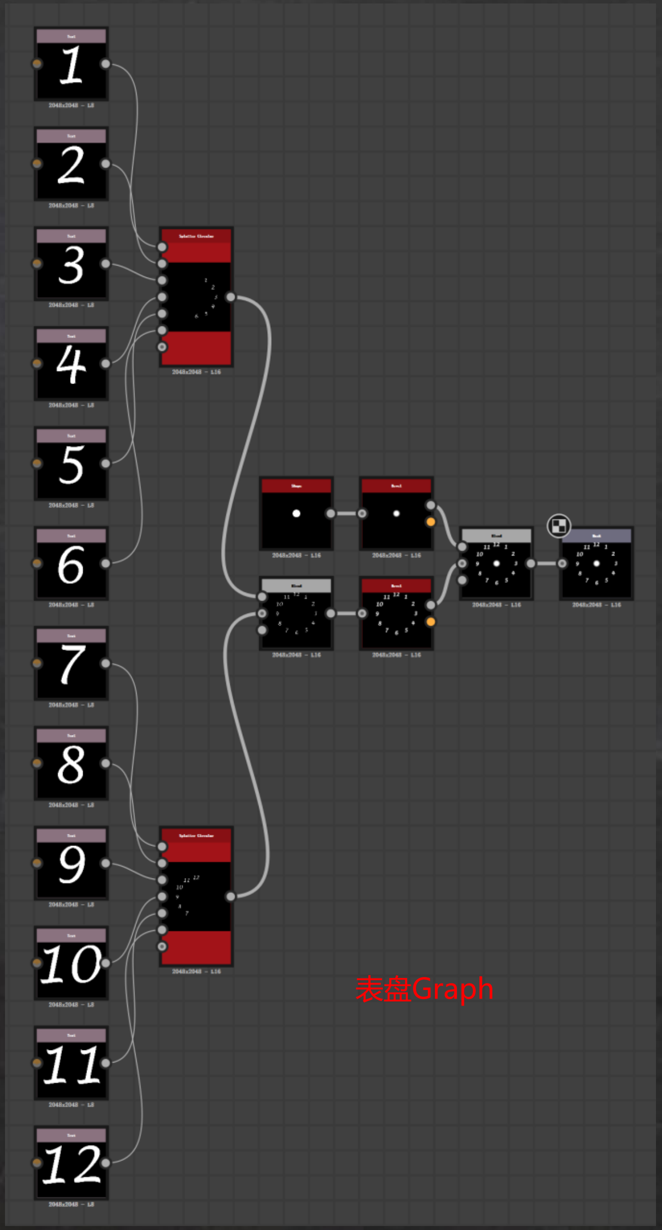
- 制作无脸化Graph；
- 分通道对各纹理无脸化；

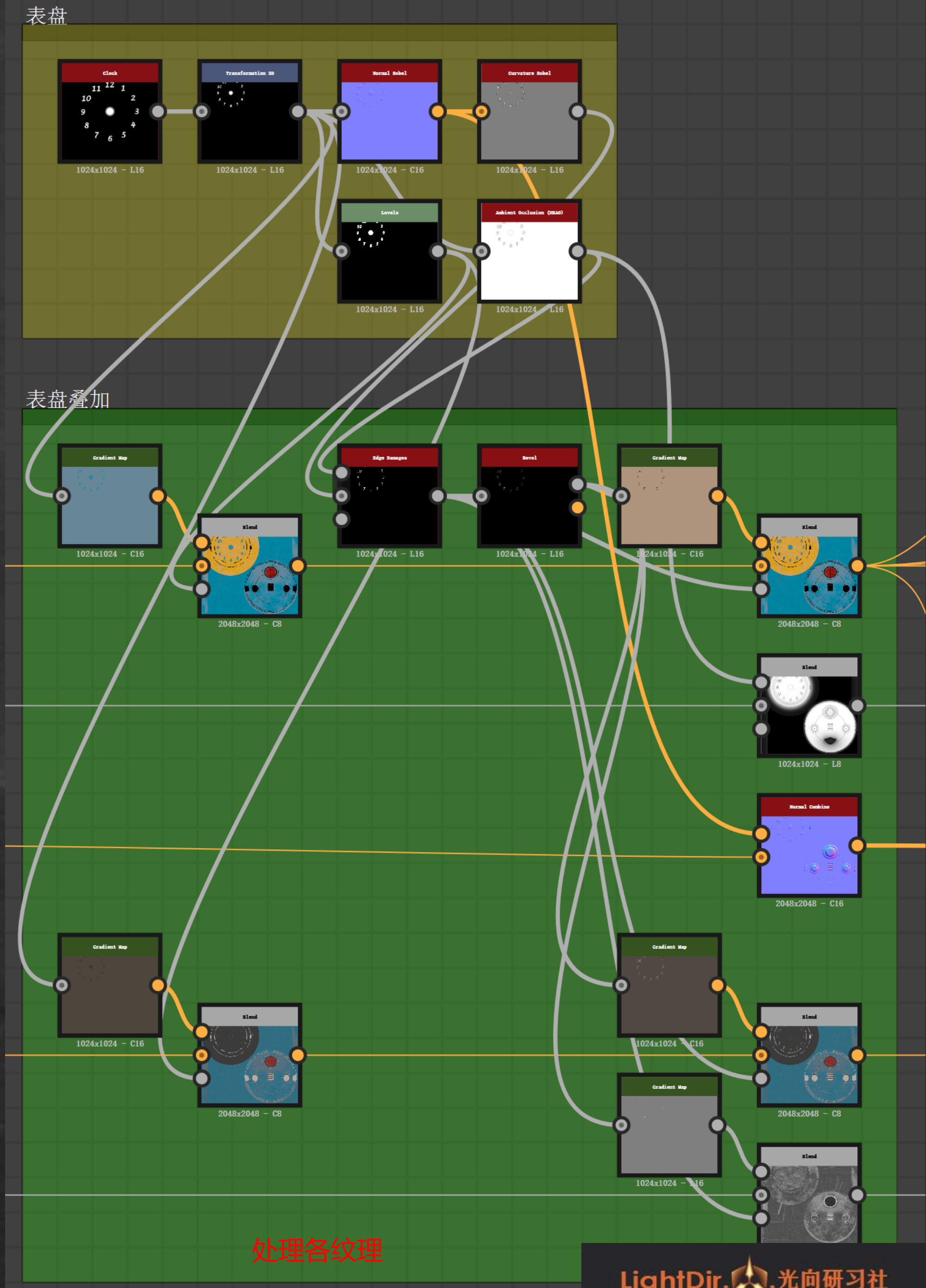
04 SD处理贴图



Step2：处理表盘；

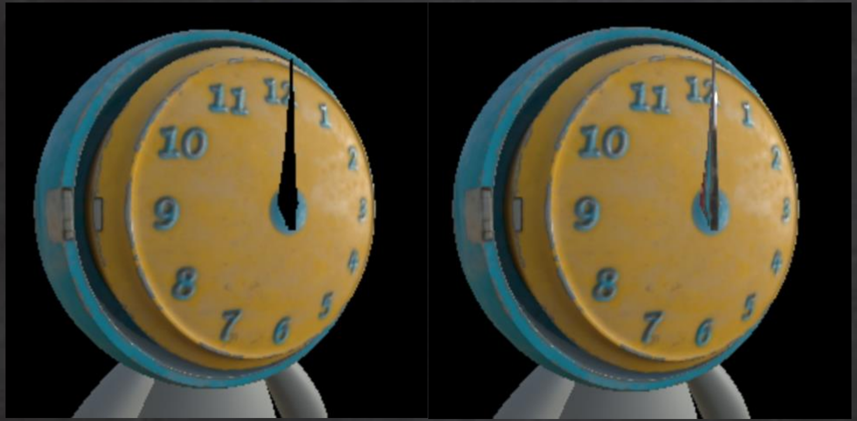
- 制作表盘Graph；
- 利用表盘Mask，处理各纹理；





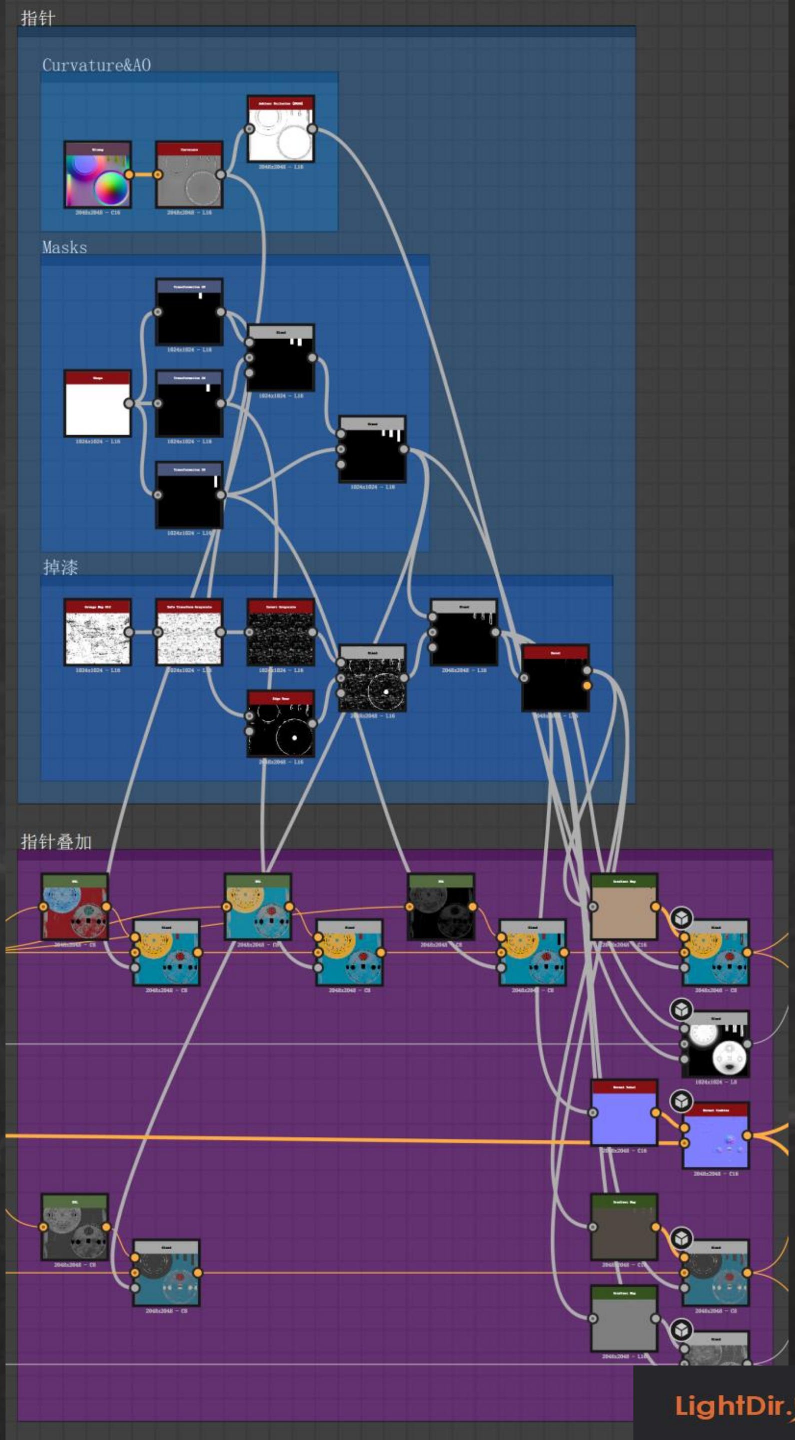
LightDir.光向研习社

05 SD处理贴图



Step3：处理指针；

- 分别抠出时针，分针，秒针Mask；
- Bake模型WSNormal，转Curvaure，再转制各种Mask；
- 利用以上Mask，处理各层纹理；



LightDir.光向研习社

06 代码实现

1. 以L11_OldSchoolPro为模板，CtrlCV；
2. 修改路径名；
3. 追加面板参数组Clock，包含以下参数：
   1. _HourHandAngle：时针转角；
   2. _MinuteHandAngle：时针转角；
   3. _SecondHandAngle：时针转角；
   4. _RotateOffset：旋转中心偏移；

2

```
Shader "AP01/L20/OldSchoolAnim" {
    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)]
            _EmitInt            ("自发光强度", range(1, 10))        = 1
        [Header(Clock)]
```

3

```
            _HourHandAngle      ("时针角度", range(0.0, 360.0))     = 0.0
            _MinuteHandAngle    ("分针角度", range(0.0, 360.0))     = 0.0
            _SecondHandAngle    ("秒针角度", range(0.0, 360.0))     = 0.0
            _RotateOffset       ("旋转偏移", range(0.0, 5.0))       = 0.0
```

07 代码实现

4. SubShaderTags，PassName，PassTags，混合模式，等：不用修改；

5. 修改cginc文件引用路径；

6. 对应声明输入参数；

```
SubShader {
    Tags {
        "RenderType"="Opaque"
    }
    Pass {
        Name "FORWARD"
        Tags {
            "LightMode"="ForwardBase"
        }
        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"
```

5

```
        #include "../../Lesson11/cginc/MyCginc.cginc"

        #pragma multi_compile_fwdbase_fullshadows
        #pragma target 3.0
        // 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;
        // Clock
        uniform float _HourHandAngle;
```

6

```
        uniform float _MinuteHandAngle;
        uniform float _SecondHandAngle;
        uniform float _RotateOffset;
```

08 代码实现

7. 顶点输入结构VertexInput追加顶点色，用于区分时钟指针；
8. 追加顶点旋转方法：void RotateZWithOffset(...)
    - angle: 旋转角度；
    - offset：旋转中心偏移距离；
    - mask：旋转遮罩；
    - vertex：inout顶点位置信息；
9. 追加时钟动画方法：void ClockAnim(...)
    - color：顶点色，用于区分指针；
    - vertex：inout顶点位置信息；
10. 顶点Shader：
    - 前面追加顶点动画预处理方法；
    - 其余不变；
11. 剩余代码：保持不变；

7

```
// 输入结构
struct VertexInput {
    float4 vertex   : POSITION;   // 顶点信息 Get✔
    float2 uv0      : TEXCOORD0;  // UV信息 Get✔
    float4 normal   : NORMAL;     // 法线信息 Get✔
    float4 tangent  : TANGENT;    // 切线信息 Get✔
    float4 color    : COLOR;
};
// 输出结构
struct VertexOutput {
    float4 pos    : SV_POSITION;  // 屏幕顶点位置
    float2 uv0      : TEXCOORD0;  // UV0
    float4 posWS    : TEXCOORD1;  // 世界空间顶点位置
    float3 nDirWS   : TEXCOORD2;  // 世界空间法线方向
    float3 tDirWS   : TEXCOORD3;  // 世界空间切线方向
    float3 bDirWS   : TEXCOORD4;  // 世界空间副切线方向
    LIGHTING_COORDS(5,6)          // 投影相关
};
```

8

```
// 沿偏移中心旋转顶点方法
void RotateZWithOffset(float angle, float offset, float mask, inout float3 vertex) {
    vertex.y -= offset * mask;
    float radZ = radians(angle * mask);
    float sinZ, cosZ = 0.0;
    sincos(radZ, sinZ, cosZ);
    vertex.xy = float2(
        vertex.x * cosZ - vertex.y * sinZ,
        vertex.x * sinZ + vertex.y * cosZ
    );
    vertex.y += offset * mask;
}
```

9

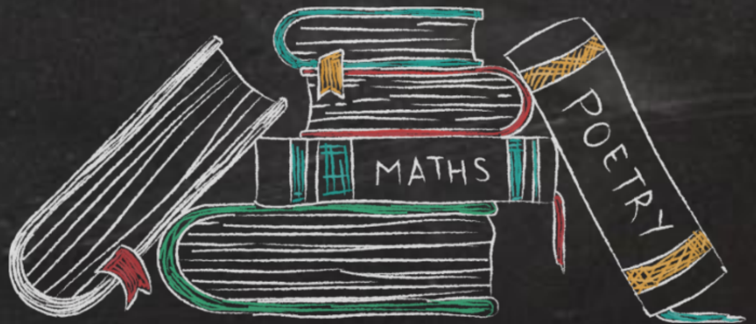
```
// 时钟动画方法
void ClockAnim(float3 color, inout float3 vertex) {
    RotateZWithOffset(_HourHandAngle, _RotateOffset, color.r, vertex);
    RotateZWithOffset(_MinuteHandAngle, _RotateOffset, color.g, vertex);
    RotateZWithOffset(_SecondHandAngle, _RotateOffset, color.b, vertex);
}
```

10

```
// 输入结构>>>顶点Shader>>>输出结构
VertexOutput vert (VertexInput v) {
    VertexOutput o = (VertexOutput)0;                   // 新建输出结构
        ClockAnim(v.color.rgb, 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
        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
        TRANSFER_VERTEX_TO_FRAGMENT(o)                  // 投影相关
    return o;                                           // 返回输出结构
}
```

LightDir.光向研习社

2 符文·筑基·时间同步



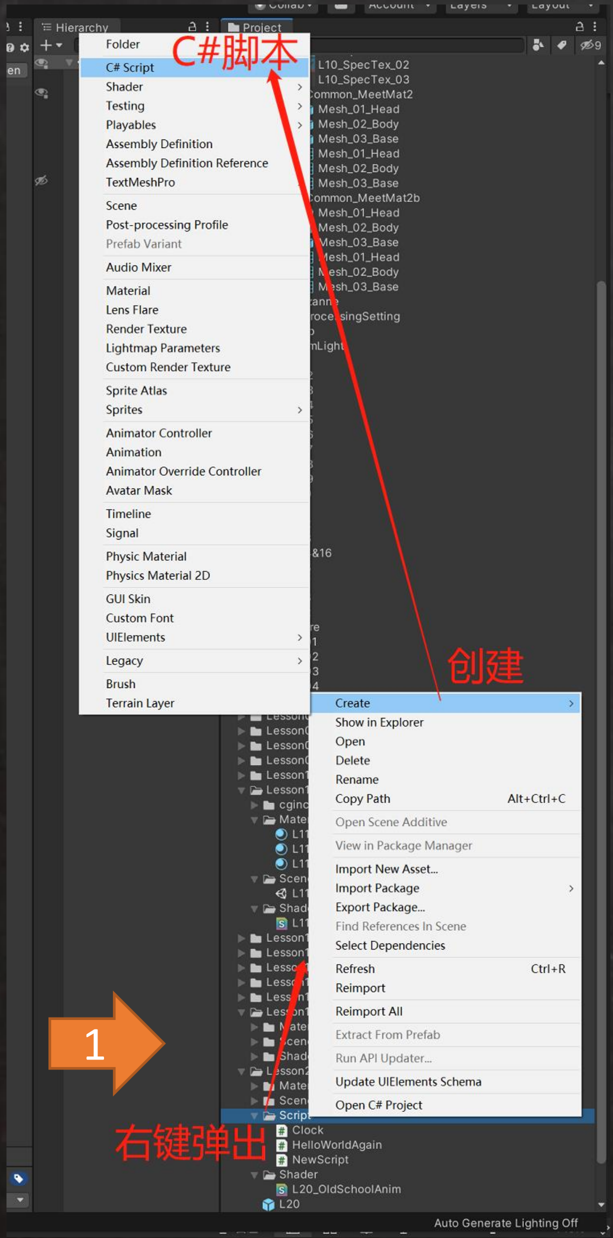
01 案例

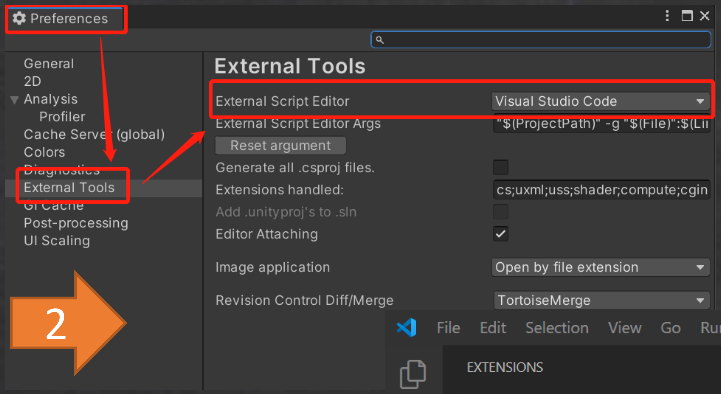
15:02

2020/11/1

- 在上节基础上，使钟表小人时间显示与系统时间一致；

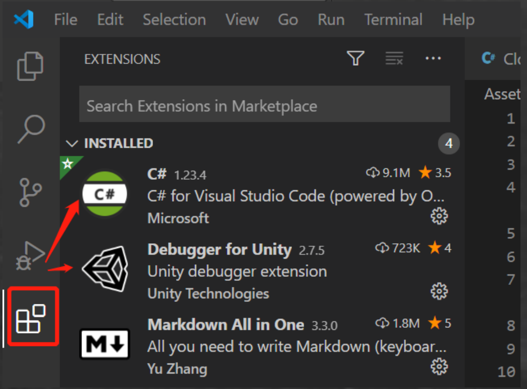
02 HelloWorld~Again~





External Tools

External Script Editor: Visual Studio Code
External Script Editor Args: "$(ProjectPath)" -g "$(File)":$(Li
Reset argument
Generate all .csproj files.
Extensions handled: cs;uxml;uss;shader;compute;cgin
Add .unityproj's to .sln
Editor Attaching ✓
Image application: Open by file extension
Revision Control Diff/Merge: TortoiseMerge



C# 1.23.4 9.1M ★3.5
C# for Visual Studio Code (powered by O...
Microsoft

Debugger for Unity 2.7.5 723K ★4
Unity debugger extension
Unity Technologies

Markdown All in One 3.3.0 1.8M ★5
All you need to write Markdown (keyboar...
Yu Zhang

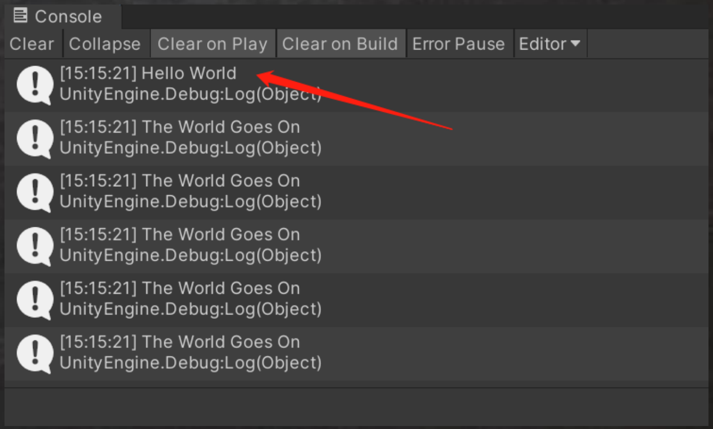
4

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HelloWorldAgain : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        Debug.Log("Hello World");
    }

    // Update is called once per frame
    void Update()
    {
        Debug.Log("The World Goes On");
    }
}
```

5



```
[15:15:21] Hello World
UnityEngine.Debug:Log(Object)
[15:15:21] The World Goes On
UnityEngine.Debug:Log(Object)
[15:15:21] The World Goes On
UnityEngine.Debug:Log(Object)
[15:15:21] The World Goes On
UnityEngine.Debug:Log(Object)
[15:15:21] The World Goes On
UnityEngine.Debug:Log(Object)
[15:15:21] The World Goes On
UnityEngine.Debug:Log(Object)
```

1. Editor中右键创建脚本，并命名；
2. 确认Editor与VSCode或者其他IDE已有关联；
3. 确认VSCode安装有相应插件（其他IDE请忽略）；
4. 双击打开C#脚本，在Start和Update中插入以下代码：
   - Start()中：Debug.Log(”Hello World”);
   - Update()中：Debug.Log(“The World Goes On”);
5. 回到Editor，运行场景，观察Console；

LightDir.光向研习社

03 代码实现

1. 创建新C#脚本，并命名为Clock；
2. 引用命名空间：
   - System：用于获取系统时间；
   - UnityEngine：使用引擎定义的类及类方法；
3. 声明Public变量：
   - Material clockMat：修改目标材质；
4. 声明Private变量：
   - bool valid：有效性标识；
   - int hourAnglePropID：时针材质属性ID；
   - int minuteAnglePropID：分针材质属性ID；
   - int secondAnglePropID：秒针材质属性ID；

```
   // 引用命名空间
2  using System;         // 主命名空间 包含所有.net基础类型和通用类型 这里用于获取系统时间
   using UnityEngine;    // Unity引擎命名空间 这里需使用UnityEngine定义的相关类和类方法

   // Clock类 继承自MonoBehaviour
1  public class Clock : MonoBehaviour
   {
       // ------ Public ------
3      public Material clockMat;             // 目标材质
       // ------ Private ------
       private bool valid;                   // 有效性标识
4      private int hourAnglePropID;          // 材质属性ID
       private int minuteAnglePropID;        // 同上
       private int secondAnglePropID;        // 同上

       // Start is called before the first frame update
       // 脚本开始运行时调用
       void Start()
       {
           ……
       }

       // Update is called once per frame
       // 脚本逐帧调用
       void Update()
       {
           ……
       }
   }
```

04 代码实现

```
    // Start is called before the first frame update
    // 脚本开始运行时调用
    void Start()
    {
        // 如果目标材质空 则跳过初始化
5-1     if(clockMat != null)
        {
            // 缓冲材质属性ID
            hourAnglePropID = Shader.PropertyToID("_HourHandAngle");
5-2         minuteAnglePropID = Shader.PropertyToID("_MinuteHandAngle");
            secondAnglePropID = Shader.PropertyToID("_SecondHandAngle");
            // 如Get到所有材质属性 标识有效性为True
5-3         if(clockMat.HasProperty(hourAnglePropID) && clockMat.HasProperty(minuteAnglePropID) && clockMat.HasProperty(secondAnglePropID))
                valid = true;
        }
    }
```

5. 实现Start()方法：

1. 目标材质判空，目标为空时终止初始化操作；
2. 缓冲Shader属性ID；
3. 属性ID判空，目标材质不包含必要属性时，不激活脚本；

05 代码实现

6. 实现Update()方法：

1. 脚本有效性判定，未激活则跳过更新操作；
2. 处理Material秒针参数；
   - 获取系统时间：秒；
   - 换算为秒针角度；
   - 将秒针角度赋给Material对应参数；
3. 同上处理分针；
4. 同上处理时针

6-1

6-2

6-3

6-4

```
// Update is called once per frame
// 脚本逐帧调用
void Update()
{
    // 判断有效性 无效则跳过Update逻辑
    if(!valid) return;
    // 处理秒针
    int second = DateTime.Now.Second;                              // 获取系统时间：秒
    float secondAngle = second / 60.0f * 360.0f;                   // 换算为指针角度
    clockMat.SetFloat(secondAnglePropID, secondAngle);             // 更新材质相关属性
    // 处理分针
    int minute = DateTime.Now.Minute;
    float minuteAngle = minute / 60.0f * 360.0f;
    clockMat.SetFloat(minuteAnglePropID, minuteAngle);
    // 处理时针
    int hour = DateTime.Now.Hour;
    float hourAngle = (hour % 12) / 12.0f * 360.0f + minuteAngle / 360.0f * 30.0f;
    clockMat.SetFloat(hourAnglePropID, hourAngle);
}
```

Thanks