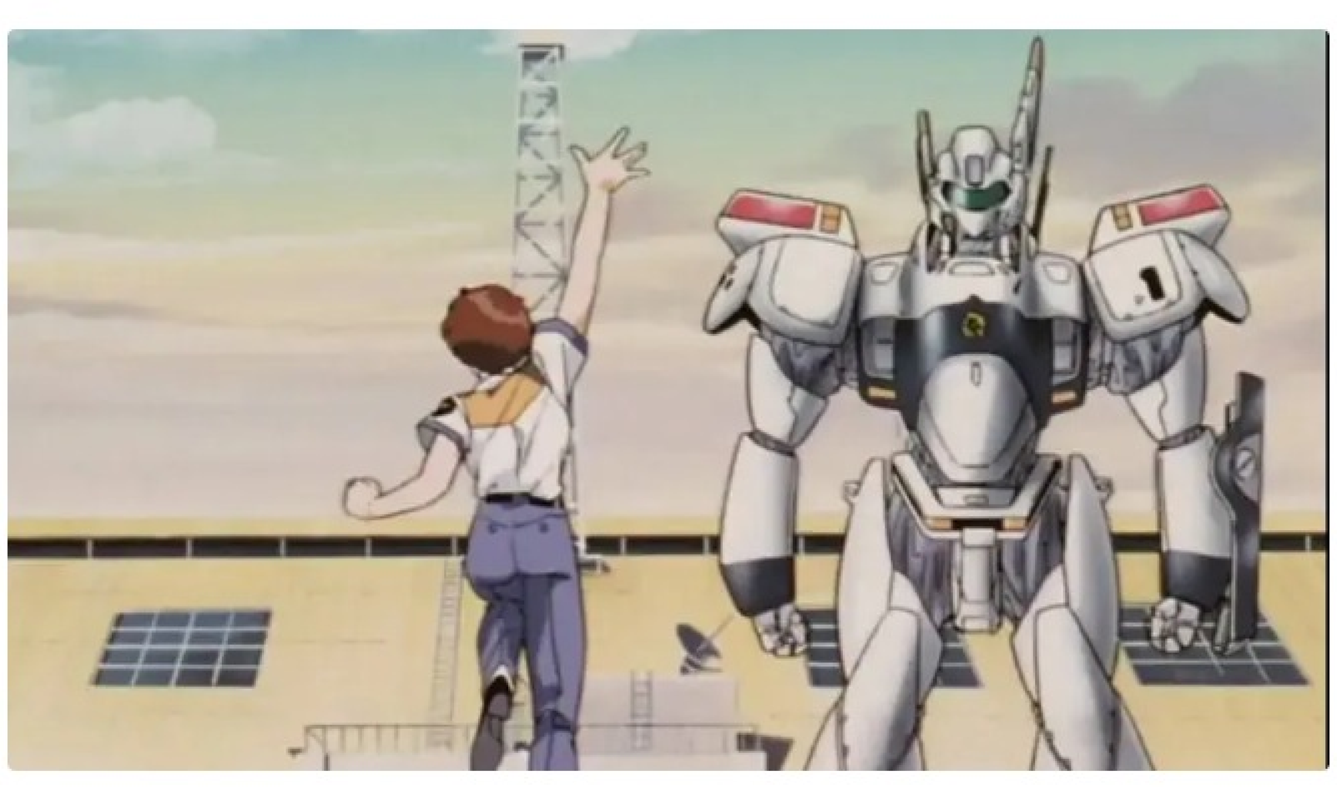
URP 入门之光照-阴影

2021年09月08日 10:04 483阅读 · 1喜欢 · 0评论
夏家因 粉丝：33 文章：2 + 关注



URP常用光照函数

GetMainLight() //获取主光源数据

half3 direction //光源方向

half3 color //光源颜色

half distanceAttenuation //距离衰减

half shadowAttenuation //阴影衰减

```
Light GetMainLight()
{
  Light light;
  light.direction = _MainLightPosition.xyz;
  light.distanceAttenuation = unity_LightData.z;
  light.shadowAttenuation = 1.0;
  light.color = _MainLightColor.rgb;
  return light;
}
```

GetMainLight() //重载1 主光源实时阴影衰减

```
Light GetMainLight(float4 shadowCoord)
{
  Light light = GetMainLight();
  light.shadowAttenuation = MainLightRealtimeShadow(shadowCoord);
  return light;
}
```

GetMainLight() //重载2

```
Light GetMainLight(float4 shadowCoord, float3 positionWS, half4 shadowMask)
{
  Light light = GetMainLight();
  light.shadowAttenuation = MainLightShadow(shadowCoord, positionWS, shadowMask, _Ma
  return light;
}
```

GetAdditionalLightCount() //返回int值 光源数量

GetAdditionalLight(index, worldPosition) //返回Light 结构体 包含光源有:

half3 direction //光源方向

half3 color //光源颜色

half distanceAttenuation //距离衰减

half shadowAttenuation //阴影衰减

示例:

```
#if _ADD_LIGHT_ON
  int addLightsCount = GetAdditionalLightsCount();
  for(int idx = 0; idx < addLightsCount; idx++)
  {
    Light addlight = GetAdditionalLight(idx, i.posWS);
    finalColor += addlight.color * addlight.distanceAttenuation * addlight.shadowAtt
  }
#endif
```

// 采用球谐光照计算环境光

```
half3 ambient = SampleSH(normalWS);
```

```
half3 ambient = _GlossyEnvironmentColor;
```

//采样烘焙贴图

```
half3 SampleLightmap(float2 lightmapUV, half3 normalWS)
{
  #ifdef UNITY_LIGHTMAP_FULL_HDR
    bool encodedLightmap = false;
  #else
    bool encodedLightmap = true;
  #endif
    half4 decodeInstructions = half4(LIGHTMAP_HDR_MULTIPLIER, LIGHTMAP_HDR_EXPONENT,
  #if defined(LIGHTMAP_ON) && defined(DIRLIGHTMAP_COMBINED)
    return SampleDirectionalLightmap(TEXTURE2D_LIGHTMAP_ARGS(LIGHTMAP_NAME, LIGHTMAP
  #elif defined(LIGHTMAP_ON)
    return SampleSingleLightmap(TEXTURE2D_LIGHTMAP_ARGS(LIGHTMAP_NAME, LIGHTMAP_SAMP
  #else
    return half3(0.0, 0.0, 0.0);
  #endif
}
```

//Light Function

Lambert

```
half3 LightingLambert(half3 lightColor, half3 lightDir, half3 normal)
{
  half NdotL = saturate(dot(normal, lightDir));
  return lightColor * NdotL;
}
```

Blinn-phong

```
half3 LightingSpecular(half3 lightColor, half3 lightDir, half3 normal, half3 viewDir
{
  float3 halfVec = SafeNormalize(float3(lightDir) + float3(viewDir));
  half NdotH = saturate(dot(normal, halfVec));
  half modifier = pow(NdotH, smoothness);
  half3 specularReflection = specular.rgb * modifier;
  return lightColor * specularReflection;
}
```

Shadow

//表明该Pass代表选择阴影渲染模式

```
Tags { "LightMode" = "ShadowCaster" }
```

//用于获取应用阴影的深度偏移后的阴影坐标

```
ApplyShadowBias(positionWS, normalWS, _LightDirection)
```

//获取阴影坐标

```
TransformWorldToShadowCoord(positionWS);
```

//一些用于接收阴影用的关键字

```
#pragma multi_compile _ _MAIN_LIGHT_SHADOWS
#pragma multi_compile _ _MAIN_LIGHT_SHADOWS_CASCADE
#pragma multi_compile _ _ADDITIONAL_LIGHTS_VERTEX _ADDITIONAL_LIGHTS
#pragma multi_compile _ _SHADOWS_SOFT
```

示例

```
Pass
{
    Name "ShadowCaster"
    Tags { "LightMode" = "ShadowCaster" }
    Cull Off
    ZWrite On
    ZTest LEqual

    HLSLPROGRAM
    #pragma vertex vert
    #pragma fragment frag

    #include "Packages/com.unity.render-pipelines.universal/Shaders/LitInput.hlsl"
    #include "Packages/com.unity.render-pipelines.universal/ShaderLibrary/Shadows.hl

    float3 _LightDirection;
    //为了让shader的SRP Batcher能够使用，所以每个pass的Cbuffer都要保持一直。（应该是这样吧
      CBUFFER_START(UnityPerMaterial)
      //...
      CBUFFER_END

      struct Attributes
      {
        float4 positionOS: POSITION;
        float3 normalOS: NORMAL;
        float2 texcoord: TEXCOORD0;
      };

    struct Varyings
    {
      float2 uv: TEXCOORD0;
      float4 positionCS: SV_POSITION;
    };

  // 获取裁剪空间下的阴影坐标
    float4 GetShadowPositionHClips(Attributes input)
    {
        float3 positionWS = TransformObjectToWorld(input.positionOS.xyz);
        float3 normalWS = TransformObjectToWorldNormal(input.normalOS);
        float4 positionCS = TransformWorldToHClip(ApplyShadowBias(positionWS, normal
        return positionCS;
    }

    Varyings vert(Attributes input)
    {
      Varyings output;
      output.uv = TRANSFORM_TEX(input.texcoord, _MainTex);
      output.positionCS = GetShadowPositionHClips(input);
      return output;
    }

    half4 frag(Varyings input): SV_TARGET
    {
      half4 albedoAlpha = SAMPLE_TEXTURE2D(_MainTex, sampler_MainTex, input.uv);
      //这里要是否需要裁剪透明通道。现在BASE是Opaque渲染模式，没有裁剪。需要用的自己改一下
      return 0;
    }
      ENDHLSL
}
```

//或者使用UsePass调用内置的pass

```
UsePass "Universal Render Pipeline/Lit/ShadowCaster"
```

//URP只支持使用单个Pass渲染 如果要开启多Pass需要按以下方式

第二个pass添加

```
Tags{ "LightMode" = "SRPDefaultUnlit" }
```

即可让这两个pass同时生效