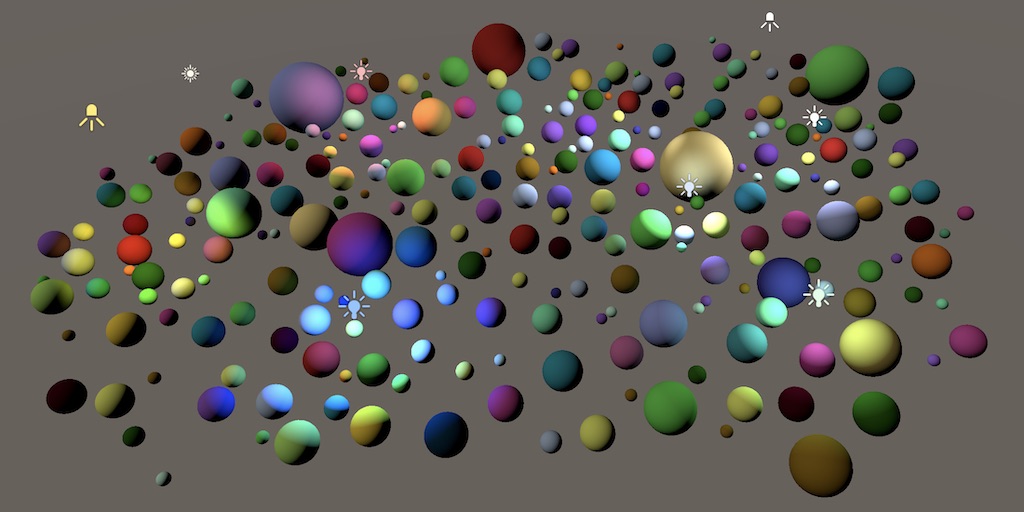
**光照-单pass前向渲染**

* **漫反射着色**
* **支持平行光、点光源、聚光灯**
* **最多支持16个可见的光源**
* **每个物体最多支持4个逐像素计算光源和4个逐顶点计算光源**

这是Unity可编程渲染管线系列教程的第三章。这章我们将添加对漫反射光照的支持，实现在一个draw call中使每个物体着色最多支持八个光源。

本教程使用Unity 2018.3.0f2完成。



**一、着色器中添加光照**

为了支持光照，我们必须为我们的管线添加一个光照着色器。照明复杂性可以从非常简单的 仅包括漫射光到非常复杂的基于物理的阴影光照。同时支持非真实感渲染，比如卡通渲染。我们将从最简单的光照着色器开始，该着色器仅计算漫反射方向光照，没有阴影。

**1.1光照着色器**

复制"Unlit.hlsl"文件并将其重命名为"Lit.hlsl"，将新文件中所有的"unlit"替换为"lit"，特别是包含定义以及顶点和片元函数名。

#ifndef MYRP\_LIT\_INCLUDED

#define MYRP\_LIT\_INCLUDED

…

VertexOutput LitPassVertex (VertexInput input) {

…

}

float4 LitPassFragment (VertexOutput input) : SV\_TARGET {

…

}

#endif // MYRP\_LIT\_INCLUDED

然后复制"Unlit.shader"文件并将其重命名为"Lit.shader"，再次将新文件中所有的"unlit"替换为"lit"。

Shader "My Pipeline/Lit" {

Properties {

\_Color ("Color", Color) = (1, 1, 1, 1)

}

SubShader {

Pass {

HLSLPROGRAM

#pragma target 3.5

#pragma multi\_compile\_instancing

#pragma instancing\_options assumeuniformscaling

#pragma vertex LitPassVertex

#pragma fragment LitPassFragment

#include "../ShaderLibrary/Lit.hlsl"

ENDHLSL

}

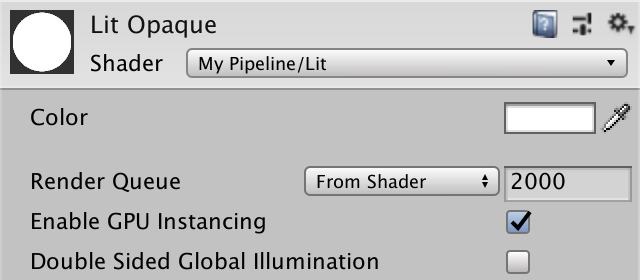
}

}

我们不应该显式使用lit pass吗？

因为我们的管线依然非常的基础，我们还不用为专门使用pass而操心。

现在我们要为新的光照着色器创建一个非透明体材质球，即使它仍然和非光照的着色器没什么区别。



**1.2 法向量**

为了计算平行光的影响，我们需要知道表面法线。所以我们必须在顶点输入和输出结构中添加法向量。有关如何计算光照的详细说明，请参见 [Rendering 4, The First Light](https://catlikecoding.com/unity/tutorials/rendering/part-4/)。

struct VertexInput {

float4 pos : POSITION;

float3 normal : NORMAL;

UNITY\_VERTEX\_INPUT\_INSTANCE\_ID

};

struct VertexOutput {

float4 clipPos : SV\_POSITION;

float3 normal : TEXCOORD0;

UNITY\_VERTEX\_INPUT\_INSTANCE\_ID

};

在LitPassVertex中将法线从对象空间转换到世界空间。因为我们假设只使用均匀缩放，因此可以简单地使用模型矩阵3x3的部分，接着在LitPassFragment的每个片元中归一化。对支持非均匀缩放要求我们使用世界到对象矩阵的转置矩阵。

VertexOutput LitPassVertex (VertexInput input) {

…

output.normal = mul((float3x3)UNITY\_MATRIX\_M, input.normal);

return output;

}

float4 LitPassFragment (VertexOutput input) : SV\_TARGET {

UNITY\_SETUP\_INSTANCE\_ID(input);

input.normal = normalize(input.normal);

…

}

为了验证我们最终是正确的法向量，将它们用到最后的颜色上。但还是保持追踪材质的颜色，我们之后将用于反照度。

float4 LitPassFragment (VertexOutput input) : SV\_TARGET {

UNITY\_SETUP\_INSTANCE\_ID(input);

input.normal = normalize(input.normal);

float3 albedo = UNITY\_ACCESS\_INSTANCED\_PROP(PerInstance, \_Color).rgb;

float3 color = input.normal;

return float4(color, 1);

}

图片

显示原生世界空间法向量。