

【翻译】第八章节：轮廓加强（关于转换法线向量）

2014-11-29 08:35:00

1831 人阅读

Unity3D

cg

轮廓加强

A⁻

A⁺

文章内容	例子源码	网友评论	最后编辑：2014-12-21 18:09:12
本文永久地址： http://www.omuying.com/article/96.aspx ，【 文章转载请注明出处！ 】			

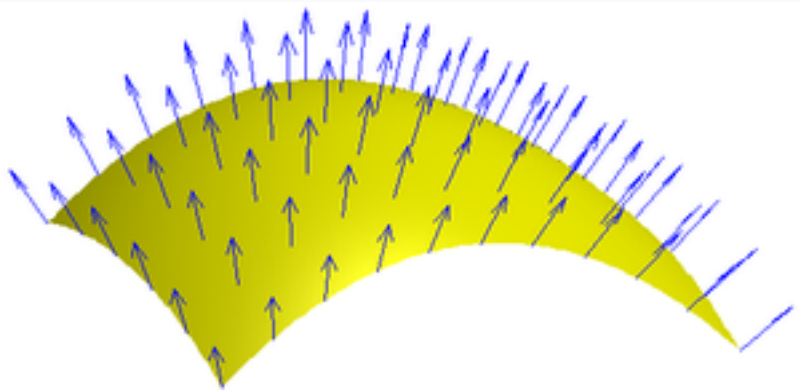
原文链接：http://en.wikibooks.org/wiki/Cg_Programming/Unity/Silhouette_Enhancement

本教程介绍表面法线向量的转换，本篇文章假定你熟悉《透明度》和《世界空间中的着色器》章节。

半透明物体的轮廓要比这个物体的其他部分更加不透明，即使没有光照也同样如此，我们可以在 Unity 中通过转换法线来达到这样的效果。

光滑表面的轮廓

在表面光滑的情况下，点在表面轮廓上的特征是法线向量平行于视图平面并且与视图方向正交。在下面的图中：



图顶部的蓝色法线向量是平行于视图平面的，而其他的法线则更多的是指向视图（摄像机），通过计算并检测视图方向与法线向量是否正交，我们可以判断一个点是否（几乎）在轮廓上。

更具体的，如果用 V 表示规范化的视图方向，用 N 表示规范化的表面法线向量，那么我们可以通过两者的点积是否等于0（即 $V \cdot N = 0$ ）来判断两个向量是否正交，但在实践中，这可能行不通（因为很少会等于0），然而我们可以通过点积（ $V \cdot N$ ）的值是否接近于0来判断一个点是否（接近）在轮廓上。

在轮廓上增加不透明度

如果点积（ $V \cdot N$ ）接近于0，我们应该增加不透明度 α 的值，当视图方向与法线向量点积值比较小时，我们有各种方法来增加不透明度的值，其中之一便是通常我们可以根据材质的常规不透明度 α 来计算增加的不透明度值 α' ：

$$\alpha' = \min \left(1, \frac{\alpha}{|V \cdot N|} \right)$$

这个公式会自动检查极端情况，假设考虑轮廓附近点的值接近于0（ $V \cdot N \approx 0$ ）。在这种情况下，通常不透明度 α 将被除以一个很小的正数（注意，GPU 可以解决除0的情况，所以我们不用担心），但是，无论 $V \cdot N$ 的值比 α 小还是比 α 大， \min 函数都会使 α' 的值不会大于1。

另一方面，对于远离轮廓的点，我们有 $V \cdot N \approx 1$ ，在这种情况下， $\alpha' \approx \min(1, \alpha) \approx \alpha$ ，即不透明的点的变化不会太大，这正是我们需要的，因此，这个公式是非常合理的。

在着色器中实现这个公式

在一个着色器中实现这个公式，第一个问题是我们应该在顶点着色器中还是在片段着色器中来实现？在某些情况下，答案非常清楚，因为我们只能在片段着色器中才可以执行纹理映射（texture mapping），但是在更多情况下这没有统一的答案，在顶点着色器中更快（通常顶点数少于片段数），但是有较低的画质（因为



比基尼除毛



游戏开发学习



unity3d摄像机



伪装微型摄像机



微型摄像机



无线监控摄像机



室内设计师




unity3d移动

广告

最新文章




【原创】C# 基础之 Lambda表达式 - 907 次阅读



【原创】C#基础之 IEnumerable和 IEnumerator - 792 次阅读



【原创】C#基础之事件 - 886 次阅读



【原创】C#基础之委托 - 912 次阅读



【原创】C#基础之委托的使用 - 856 次阅读



游戏开发学习



针孔摄像头



比基尼除毛



伪装微型摄像机



unity3d摄像机



微型摄像机



unity3d移动



室内设计师

广告

随机阅读

法线向量与其他顶点的属性可能在顶点之间被突然改变），因此，如果你更关心性能问题，在顶点着色器中实现可能会比较好，但是如果你更关心画质，在片段着色器中实现可能是比较好的选择，每顶点照明《[Specular Highlights](#)》与每片段照明《[Smooth Specular Highlights](#)》也有类似的权衡选择。

接下来的问题是，我们应该选择哪个坐标系统，同样，这也没有统一的答案，然而，在世界坐标系中执行会是一个不错的选择，因为在 Unity 中很多 uniform 变量的值都是在世界坐标系中指定（在其他的环境中，使用视图坐标也很常见）。

最后一个是，我们从哪里获得公式中需要的参数？通常不透明度 α 由着色器的属性指定（RGBA 颜色内），法线向量 gl_Normal 是一个标准的顶点输入参数《[在着色器中调试](#)》，视图的方向可以在顶点着色器中通过 Unity 提供的顶点位置向量 _WorldSpaceCameraPos（世界空间中的顶点位置到摄像机的位置）计算。

因此我们只需要在使用公式计算之前把顶点位置和法线向量转到世界空间中，在《世界空间中的着色器》章节，我们讨论了对对象转到世界空间的 _Object2World 矩阵和从世界转到对象空间的 _World2Object 矩阵，在《[Applying Matrix Transformations](#)》章节中详细描述了点和法线向量的矩阵变换，因此点和方向的变换可以只通过乘以他们自己的变换矩阵就可以了，例如用 modelMatrix 代替 _Object2World：

```
1 output.viewDir = normalize(_WorldSpaceCameraPos - mul(modelMatrix, input.vertex).xyz);
```

另一方面，法线向量通过法线乘以对象的转置逆变换矩阵来转换，由于 Unity 为我们提供了逆变换矩阵（_World2Object * unity_Scale.w 除了右下角元素），其实更好的替代方法是法线向量左乘逆变换矩阵（mul 时法线向量在左边），这等价于右乘转置逆变换矩阵（mul 时法线向量在右边），详情可查看《[Applying Matrix Transformations](#)》：

```
1 output.normal = normalize(mul(float4(input.normal, 0.0), modelMatrixInverse).xyz);
```

注意，错误的右下角矩阵是没有问题的，因为它总是乘以 0，此外也不需要与 unity_Scale.w 相乘，因为在这里缩放是无关紧要的，因为我们已经把向量规范化。

着色器代码

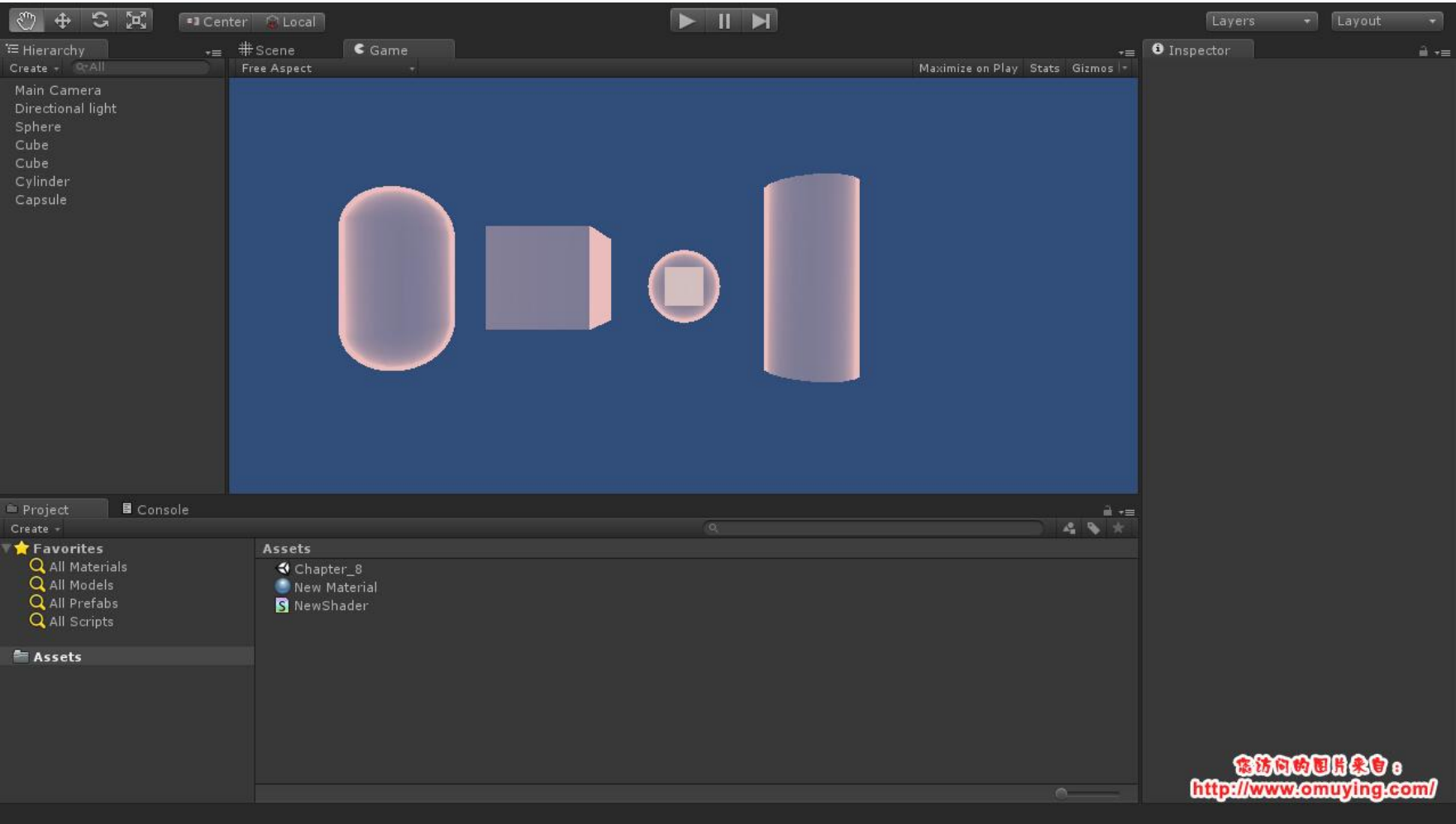
现在我们可以编写一个着色器了：

```
01 Shader "Cg silhouette enhancement"
02 {
03     Properties
04     {
05         _Color ("Color", Color) = (1, 1, 1, 0.5)
06         // user-specified RGBA color including opacity
07     }
08     SubShader
09     {
10         Tags { "Queue" = "Transparent" }
11         // draw after all opaque geometry has been drawn
12         Pass
13         {
14             ZWrite Off // don't occlude other objects
15             Blend SrcAlpha OneMinusSrcAlpha // standard alpha blending
16
17             CGPROGRAM
18
19             #pragma vertex vert
20             #pragma fragment frag
21
22             #include "UnityCG.cginc"
23
24             uniform float4 _Color; // define shader property for shaders
25
26             struct vertexInput
27             {
28                 float4 vertex : POSITION;
29                 float3 normal : NORMAL;
30             };
31             struct vertexOutput
32             {
33                 float4 pos : SV_POSITION;
34                 float3 normal : TEXCOORD;
35                 float3 viewDir : TEXCOORD1;
36             };
37
```

 暂无图片	【翻译】第二十四章节：表面反射（关于反射贴图） - 1441 次阅读
 暂无图片	【翻译】第十三章节：双面平滑表面（关于双面每像素光照） - 1101 次阅读
 暂无图片	【原创】Shader 内置 Shader 之 Specular 学习 - 2024 次阅读
 暂无图片	【翻译】第一章节：最简单的着色器（关于着色器，材质和游戏对象） - 2549 次阅读
 暂无图片	【原创】Shader 内置 Shader 之 Bumped Specular 学习 - 1785 次阅读


```
38     vertexOutput vert(vertexInput input)
39     {
40         vertexOutput output;
41
42         float4x4 modelMatrix = _Object2World;
43         float4x4 modelMatrixInverse = _World2Object;
44         // multiplication with unity_Scale.w is unnecessary
45         // because we normalize transformed vectors
46
47         output.normal = normalize(mul(float4(input.normal, 0.0),
modelMatrixInverse).xyz);
48         output.viewDir = normalize(_WorldSpaceCameraPos - mul(modelMatrix,
input.vertex).xyz);
49
50         output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
51         return output;
52     }
53
54     float4 frag(vertexOutput input) : COLOR
55     {
56         float3 normalDirection = normalize(input.normal);
57         float3 viewDirection = normalize(input.viewDir);
58
59         float newOpacity = min(1.0, _Color.a / abs(dot(viewDirection,
normalDirection)));
60         return float4(_Color.rgb, newOpacity);
61     }
62
63     ENDCG
64 }
65 }
66 }
```

这个着色器的效果如图：



着色器中 newOpacity 的值可以直接通过公式求得：

$$\alpha' = \min(1, \alpha / |\mathbf{V} \cdot \mathbf{N}|)$$

注意我们的顶点输出参数 output.normal 和 output.viewDir 在顶点着色器（因为方向的插值数据可能会超出规范化的范围）和片段着色器（因为插值数据可能不在规范化的范围内）中都规范化了，然而，在许多情况下，output.normal 在顶点着色器中规范化不是必要的，同样的，output.viewDir 在片段着色器中规范化通常也没有必要。

添加艺术控制

尽管增强轮廓是基于物体模型的，但它缺乏艺术控制，即一个艺术家不能轻松的创建比物体模型更薄或者更厚的轮廓。为了可以提供更多的控制，你可以在着色器中添加一个（正）浮点属性，并在使用公式之前计算点积 $|\mathbf{V} \cdot \mathbf{N}|$ 与这个浮点属性的 power 值，这样，CG 艺术家根据颜色的不透明度可以单独来调整更薄或者厚的轮廓。

下一个着色器，我们添加了一个用户可以控制的属性，代码如下：

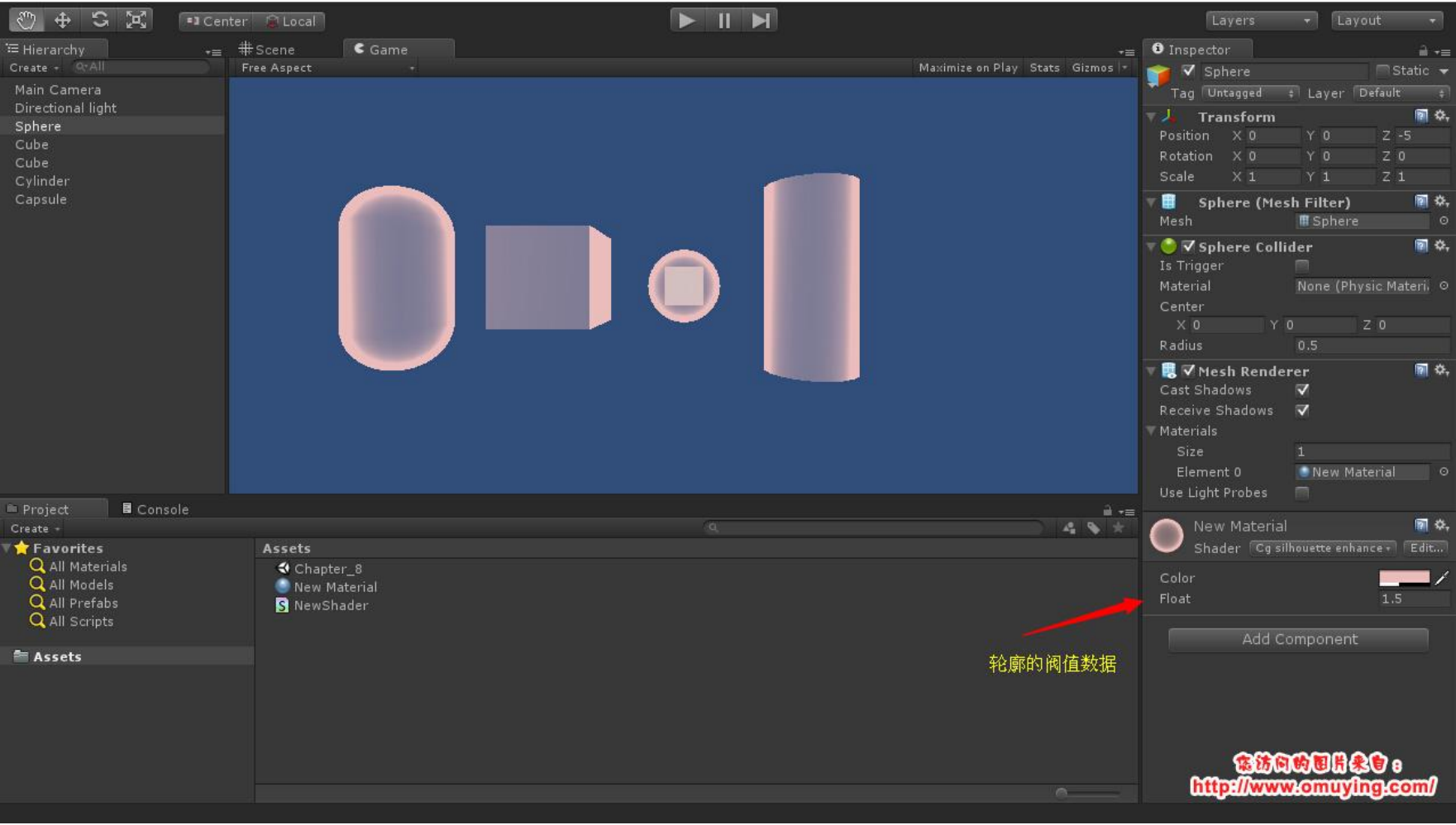
```
01 Shader "Cg silhouette enhancement"
02 {
03     Properties
04     {
```

```

05     _Color ("Color", Color) = (1, 1, 1, 0.5)
06     _Threshold("Float", float) = 0.5
07     // user-specified RGBA color including opacity
08 }
09 SubShader
10 {
11     Tags { "Queue" = "Transparent" }
12     // draw after all opaque geometry has been drawn
13     Pass
14     {
15         ZWrite Off // don't occlude other objects
16         Blend SrcAlpha OneMinusSrcAlpha // standard alpha blending
17
18         CGPROGRAM
19
20         #pragma vertex vert
21         #pragma fragment frag
22
23         #include "UnityCG.cginc"
24
25         uniform float4 _Color; // define shader property for shaders
26         uniform float _Threshold;
27
28         struct vertexInput
29         {
30             float4 vertex : POSITION;
31             float3 normal : NORMAL;
32         };
33         struct vertexOutput
34         {
35             float4 pos : SV_POSITION;
36             float3 normal : TEXCOORD;
37             float3 viewDir : TEXCOORD1;
38         };
39
40         vertexOutput vert(vertexInput input)
41         {
42             vertexOutput output;
43
44             float4x4 modelMatrix = _Object2World;
45             float4x4 modelMatrixInverse = _World2Object;
46             // multiplication with unity_Scale.w is unnecessary
47             // because we normalize transformed vectors
48
49             output.normal = normalize(mul(float4(input.normal, 0.0),
modelMatrixInverse).xyz);
50             output.viewDir = normalize(_WorldSpaceCameraPos - mul(modelMatrix,
input.vertex).xyz);
51
52             output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
53             return output;
54         }
55
56         float4 frag(vertexOutput input) : COLOR
57         {
58             float3 normalDirection = input.normal;
59             float3 viewDirection = input.viewDir;
60
61             float newOpacity = min(1.0, _Color.a / abs(pow(dot(viewDirection,
normalDirection), _Threshold)));
62             return float4(_Color.rgb, newOpacity);
63         }
64     }
65 }
66 }
67 }

```

通过调节阈值，我们就可以调节着色器的轮廓效果，如图：



恭喜你，本章节中你应该了解：

- 1、如何找到光滑表面的轮廓（法线向量与视图方向的点积）。
- 2、如何在轮廓上增加不透明度。
- 3、如何在着色器中实现增加不透明度的方程。
- 4、如何转换把点与法线向量从对象坐标系转到世界坐标系（对法线向量使用转置逆变化矩阵）。
- 5、如何计算视图方向（摄像机位置到顶点位置的差）。
- 6、如何规范化插值方向（在顶点与片面着色器中使用两次规范化函数）。
- 7、如何通过增加属性来自由控制对象轮廓的厚度。

资源下载地址：[点击下载](#)，共下载 24 次。

前一篇：[第七章：顺序无关的透明度（关于顺序无关的混合）](#)

后一篇：[第九章：漫反射（关于每顶点漫反射和多光源漫反射）](#)



赞

9 人



打酱油

0 人



呵呵

0 人



鄙视

0 人



正能量

0 人



1 条评论

最新 最早 最热



海风

很好，谢谢楼主

2015年12月11日

← 回复

♥ 顶

→ 转发

社交帐号登录:

微信

微博

QQ

人人

[更多»](#)



说点什么吧...



发布

最终幻想正在使用多说

1 条评论

最新 最早 最热



海风
很好，谢谢楼主

2015年12月11日 [← 回复](#) [♥ 顶](#) [→ 转发](#)

社交帐号登录: [微信](#) [微博](#) [QQ](#) [人人](#) [更多»](#)



说点什么吧...



发布

最终幻想正在使用多说