# 1.创建一个Unlit材质，使用Unlit显示模型

|  |
| --- |
|  |

# 2.Unlit材质的主要参数详解

|  |
| --- |
|  |
|  |

# 3.演练

## 1.新建一个项目，起名classFifteen-UnlitShader,然后在assets文件夹里面新建3个文件夹：AssetsPackage，Scenes和Scripts，然后把主场景作为main保存到Scenes文件夹中

|  |
| --- |
|  |

## 2.在AssetsPackage里面新建一个Models文件夹，把上一节课的河豚模型放进来

|  |
| --- |
|  |

## 3.拖拽模型到场景创建一个物体

|  |
| --- |
|  |

## 4.在Model文件夹里面创建一个hetun材质，把右边材质的显示模式改为球体

|  |
| --- |
|  |

## 5.在Effect一栏旋转builtin-unlit，也就是不受光照影响的材质类型

|  |
| --- |
|  |

## 6.因为他是不受光照影响的，我们需要给他勾选USE TEXTURE选项，然后把河豚图片驼峰到MainTexture一栏

|  |
| --- |
|  |

## 7.为了方便观察效果，我们添加叶子hetun1图片，然后用它来修改河豚材质的贴图

|  |
| --- |
|  |

## 8.然后点击场景中河豚物体的body节点，用我们的hetun材质来替换它原来的材质，效果如下

|  |
| --- |
|  |

## 9.因为这个材质不受光照影响，所以我们把光照关闭了，对他是没有影响的。

|  |
| --- |
|  |

## 10.点击河豚材质，可以看到右边的属性检查器里面有很多属性，其中mainTexture就是主纹理图片，tilingOffset就是这个纹理从什么位置开始贴，这两个属性只有你勾选了USE TEXTURE复选框才能可见。如果你不勾选使用纹理，那么它就是使用MainColor来绘制物体，然后面板上有USE VERTEX COLOR选项，勾选了它，创建中的物体会变黑色。因为这里物体的顶点颜色就是黑色的，此时引擎会把黑色叠加到物体上面，所以我们看到的是黑色。下面有mainColor选项，你可以选择一个颜色，比如紫色，然后在ColorScale里面修改每一个轴向的值，效果如下

|  |
| --- |
|  |

### 这个颜色会叠加到纹理的颜色，Technique表示把物体放在哪个渲染队列里面，USE BATCH表示是否开启合批，USE INSTANCE表示是否开启GPU合批，Pass表示绘制物体的一个完整流程，可以有多个，Pass0，Pass1，Pass2等等，Effect表示关联Shader的代码。还有一个USE ALPHA TEST复选框，勾选他就会有2给选项，一个是颜色通道 rgba中的一个，默认是a，可以根据你的选择来修改物体的颜色，比如我们选择r通道，效果如下

|  |
| --- |
|  |

### 因为是蓝色的。我们选择绿色，会看不到任何东西

|  |
| --- |
|  |

### 如果选择蓝色，效果如下

|  |
| --- |
|  |

## 11如果你需要看一下unlit Shader的代码，可以在资源管理器的搜索框里面输入unlit，就会有显示如图

|  |
| --- |
|  |

## 12.在builtin-unlit上面点击右键-》在资源管理器里中显示，就可以打开一个资源管理器窗口并且里面有builtin-unlit.effect文件

|  |
| --- |
|  |

## 13.然后用一个文本编辑器把它打开，这个代码我们目前的水平还是比较难看懂，不过后面我们会学习到

|  |
| --- |
| // Copyright (c) 2017-2020 Xiamen Yaji Software Co., Ltd.  CCEffect %{  techniques:  - name: opaque  passes:  - vert: unlit-vs:vert  frag: unlit-fs:frag  properties: &props  mainTexture: { value: grey }  tilingOffset: { value: [1, 1, 0, 0] }  mainColor: { value: [1, 1, 1, 1], editor: { type: color } }  colorScale: { value: [1, 1, 1], target: colorScaleAndCutoff.xyz }  alphaThreshold: { value: 0.5, target: colorScaleAndCutoff.w, editor: { parent: USE\_ALPHA\_TEST } }  color: { target: mainColor, editor: { visible: false } } # backward compability  migrations: &migs  properties:  mainColor: { formerlySerializedAs: color }  - name: transparent  passes:  - vert: unlit-vs:vert  frag: unlit-fs:frag  depthStencilState: &d1  depthTest: true  depthWrite: false  blendState:  targets:  - blend: true  blendSrc: src\_alpha  blendDst: one\_minus\_src\_alpha  blendDstAlpha: one\_minus\_src\_alpha  properties: \*props  migrations: \*migs  - name: add  passes:  - vert: unlit-vs:vert  frag: unlit-fs:frag  rasterizerState: &r1 { cullMode: none }  depthStencilState: \*d1  blendState:  targets:  - blend: true  blendSrc: src\_alpha  blendDst: one  blendSrcAlpha: src\_alpha  blendDstAlpha: one  properties: \*props  migrations: \*migs  - name: alpha-blend  passes:  - vert: unlit-vs:vert  frag: unlit-fs:frag  rasterizerState: \*r1  depthStencilState: \*d1  blendState:  targets:  - blend: true  blendSrc: src\_alpha  blendDst: one\_minus\_src\_alpha  blendSrcAlpha: src\_alpha  blendDstAlpha: one\_minus\_src\_alpha  properties: \*props  migrations: \*migs  }%  CCProgram unlit-vs %{  precision highp float;  #include <input>  #include <cc-global>  #include <cc-local-batch>  #include <input>  #include <cc-fog>  #if USE\_VERTEX\_COLOR  in lowp vec4 a\_color;  out lowp vec4 v\_color;  #endif  #if USE\_TEXTURE  out vec2 v\_uv;  uniform TexCoords {  vec4 tilingOffset;  };  #endif  out float factor\_fog;  vec4 vert () {  vec4 position;  CCVertInput(position);  mat4 matWorld;  CCGetWorldMatrix(matWorld);  #if USE\_TEXTURE  v\_uv = a\_texCoord \* tilingOffset.xy + tilingOffset.zw;  #endif  #if USE\_VERTEX\_COLOR  v\_color = a\_color;  #endif  factor\_fog = CC\_TRANSFER\_FOG(matWorld \* position);  return cc\_matProj \* (cc\_matView \* matWorld) \* position;  }  }%  CCProgram unlit-fs %{  precision highp float;  #include <output>  #include <cc-fog>  #if USE\_ALPHA\_TEST  #pragma define ALPHA\_TEST\_CHANNEL options([a, r, g, b])  #endif  #if USE\_TEXTURE  in vec2 v\_uv;  uniform sampler2D mainTexture;  #endif  uniform Constant {  vec4 mainColor;  vec4 colorScaleAndCutoff;  };  #if USE\_VERTEX\_COLOR  in lowp vec4 v\_color;  #endif  in float factor\_fog;  vec4 frag () {  vec4 o = mainColor;  o.rgb \*= colorScaleAndCutoff.xyz;  #if USE\_VERTEX\_COLOR  o \*= v\_color;  #endif  #if USE\_TEXTURE  o \*= texture(mainTexture, v\_uv);  #endif  #if USE\_ALPHA\_TEST  if (o.ALPHA\_TEST\_CHANNEL < colorScaleAndCutoff.w) discard;  #endif  o = CC\_APPLY\_FOG(o, factor\_fog);  return CCFragOutput(o);  }  }% |

## 14.可以看到方式属性检查器上面有的属性，在Shader文件里面都有变量来保存他的值，复选框就是#if--#endif里面的语句（其实，着色器程序是一种类似于c语言的程序）

|  |
| --- |
|  |
|  |