
使用Unity开发
Vive Focus游戏内容
HTC VIVE

目 录


程序

01 环境配置

配置Android开发环境

下载并安装Java 8 SDK

添加安装目录的bin文件夹到PATH环境变量

配置Android SDK

https://developer.android.com/studio#downloads

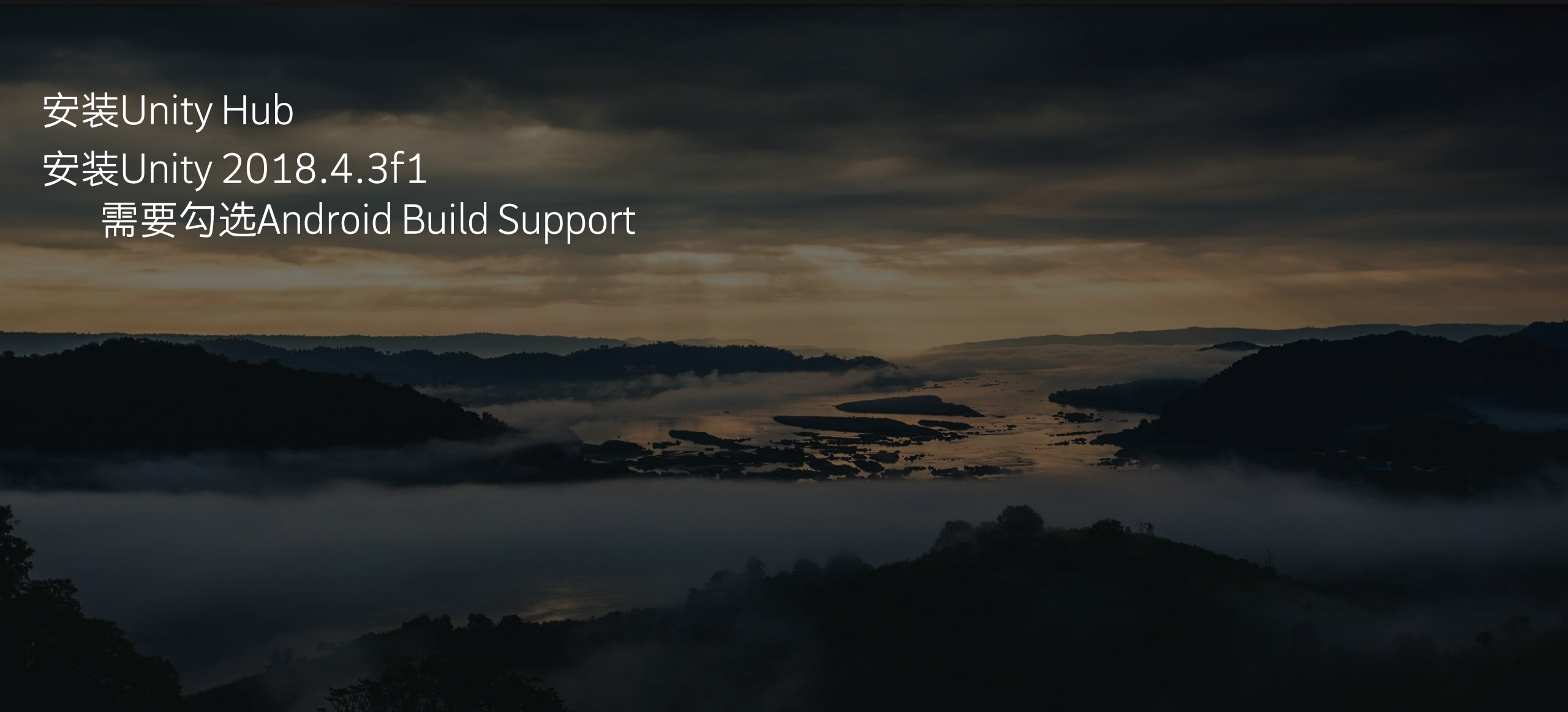
选择Command line tools only， 下载并解压缩Android SDK

打开cmd (Windows)/Terminal(Mac)， 进入解压缩之后的目录

运行.\tools\bin\sdkmanager "build-tools;28.0.3" "platforms;android-25" tools platform-tools

添加Android SDK目录的platform-tools文件夹到PATH环境变量

安装Unity



安装Unity Hub

安装Unity 2018.4.3f1

需要勾选Android Build Support

配置Unity工程

打开Unity 2018/Unity Hub，新建工程

配置Unity

- Edit – Preferences，选择External Tools标签，选择Android SDK路径

切换到Android平台

- File – Build Settings，选择Android，点击Switch Platform

点击Player Settings修改工程的设置

- Company Name
- Other Settings
  - Package Name: Android App ID，每个应用都要不一样
  - Target Architectures: 取消勾选x86

配置Vive Focus开发环境

导入WaveVR 3.0.2 Unity插件 (wvr_unity_sdk.unitypackage)

导入WaveVR 3.0.2样例 (wvr_unity_samples.unitypackage)

修改设置

- Ignore最后一个设置 (Enable VR and single-pass Support)
- 点击Accept All接受其他所有WaveVR推荐设置

确认如下Player Settings设置

- Other Settings: Scripting Define Symbols不包含WAVEVR_SINGLEPASS_ENABLED
- XRSettings: 不勾选Virtual Reality Supported

配置Vive Focus开发环境(2)

打开并带上Focus

设置-更多设置-开发者选项菜单(滑动到最下方)

如果没有开发者选项菜单

　　进入关于设备菜单，点击版本号10次，返回上一级菜单

进入开发者选项菜单，勾选开启和USB调试

把Focus连接到PC上，在对话框上勾选信任计算机，点击确定

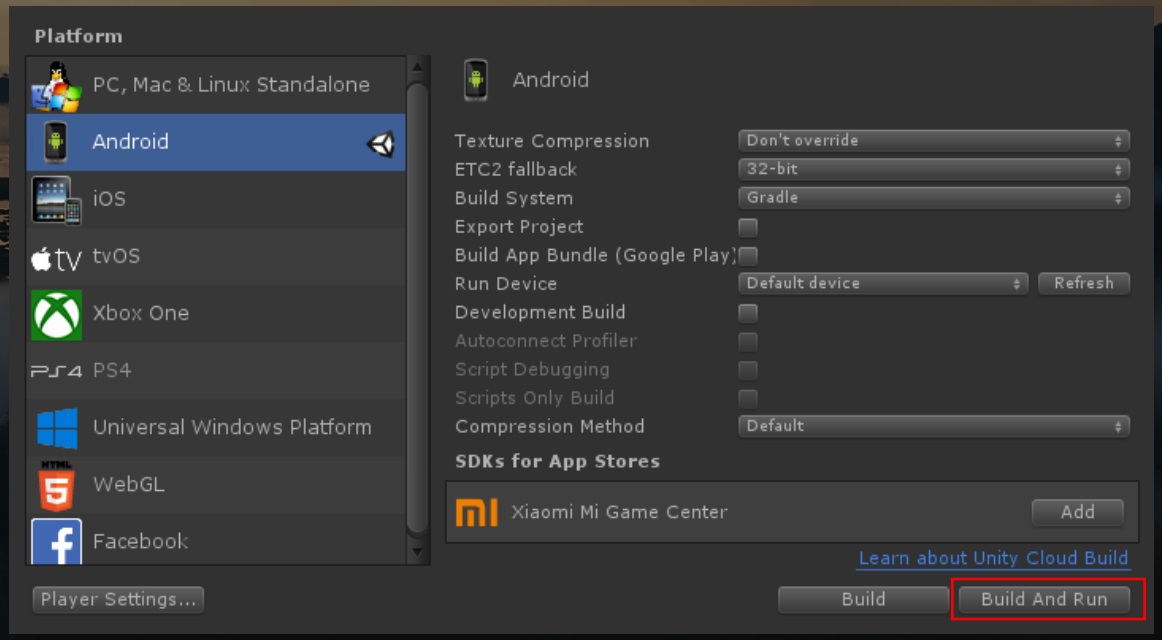
在命令行上运行adb devices命令

```
C:\>adb devices
List of devices attached
FA7CMJJ00570    device
```

打包测试

每次打包时，确保Focus已连接到电脑

双击打开Samples/HelloVR/Scenes/HelloVR场景
打开菜单File – Build Settings，点击build and run
编译结束后带上Focus即可体验



02 VIVE Input Utility (VIU)介绍

VIVE Input Utility介绍

简化VR交互设计

提供不同手柄硬件适配

支持广泛的硬件

- VIVE/VIVE Pro
- Oculus Rift / Go
- Daydream
- VIVE Wave (包括VIVE Focus 及其他一体机)
- 微软MR

为什么选择VIVE Input Utility

交互是VR的重要部分

统一的头盔定位接口
统一的手柄交互接口
PC端模拟6DoF头盔加双6DoF手柄
提供UI交互，传送，物体抓取等VR基本操作

导入VIVE Input Utility

导入VIVE Input Utility插件 (ViveInputUtility_v1.10.4.unitypackage)

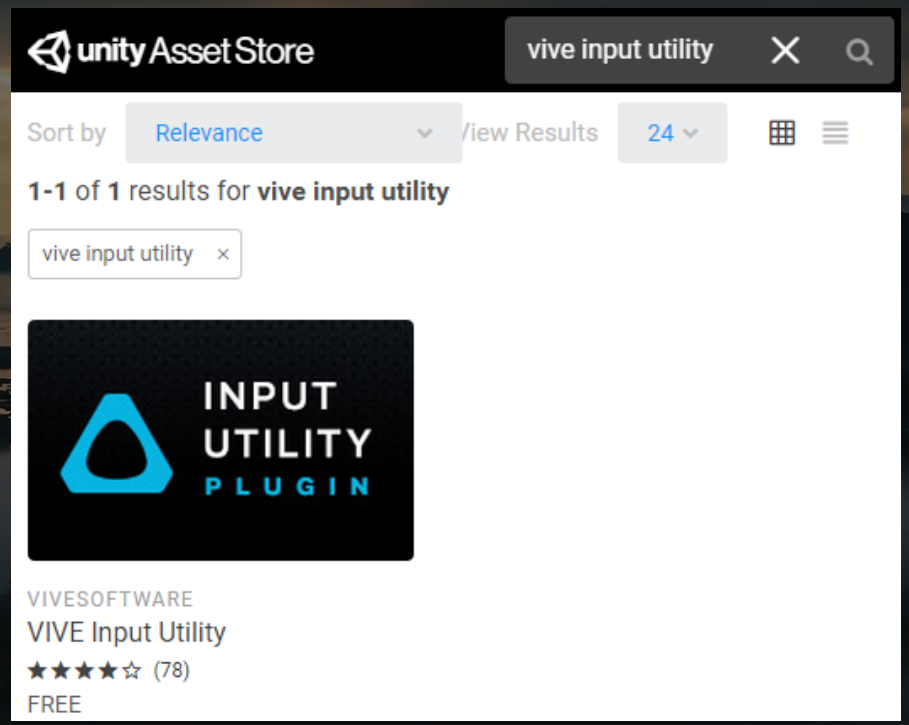
导入VIVE Input Utility

导入VIVE Input Utility插件 (ViveInputUtility_v1.10.4.unitypackage)

打开Unity Asset Store

菜单Windows – Asset Store

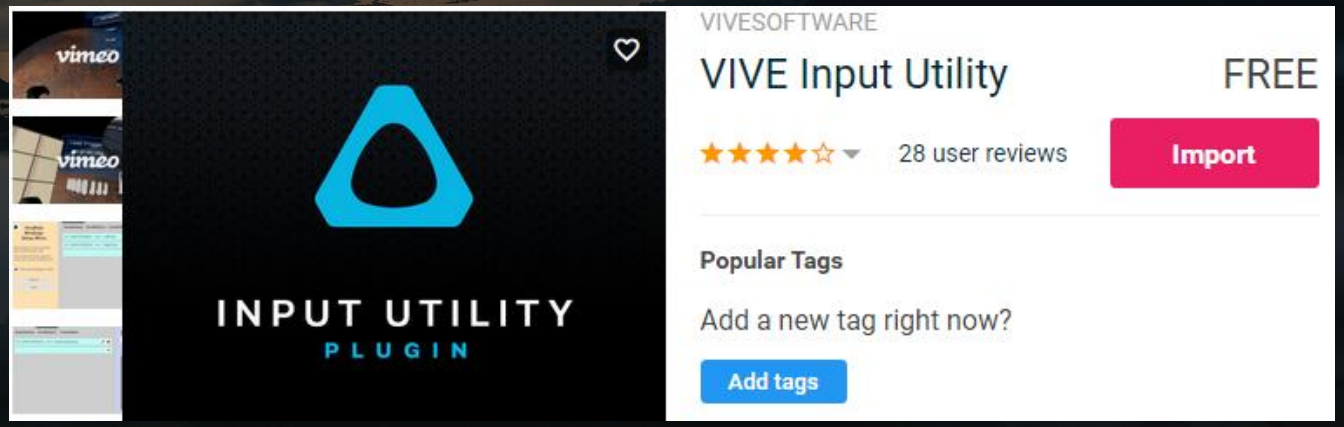
搜索VIVE Input Utility



导入VIVE Input Utility

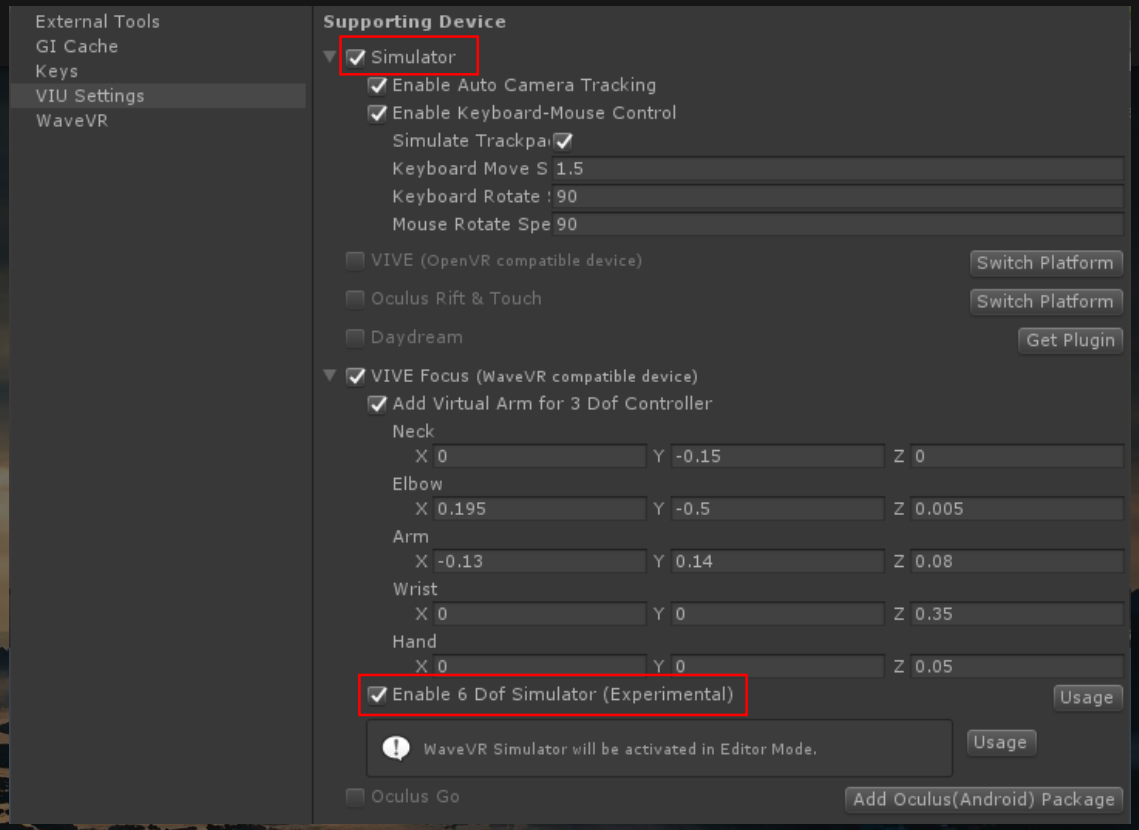
导入VIVE Input Utility插件 (ViveInputUtility_v1.10.4.unitypackage)

打开Unity Asset Store
　　菜单Windows – Asset Store
搜索VIVE Input Utility
点开之后点击Download
下载完成之后点击Import



配置VIVE Input Utility

- 修改设置： Edit – Preferences – VIU Settings



- 双击打开HTC.UnityPlugin/ViveInputUtility/Examples/0.Tutorial/Tutorial场景
- 打包运行

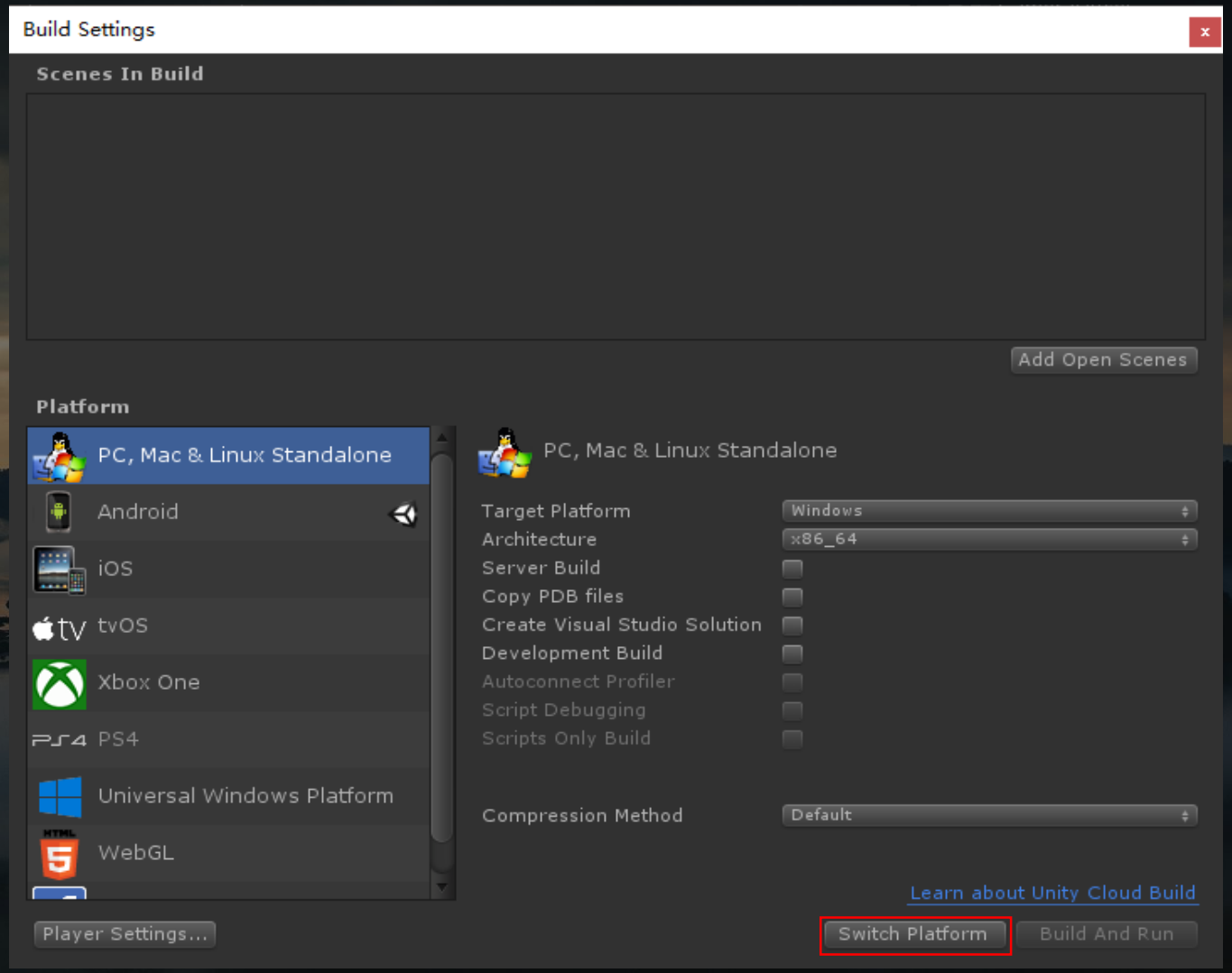
使用VIU模拟器

在Unity Editor中运行/测试

使用VIU模拟器

在Unity Editor中运行/测试

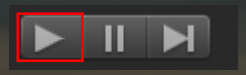
切换到Windows平台



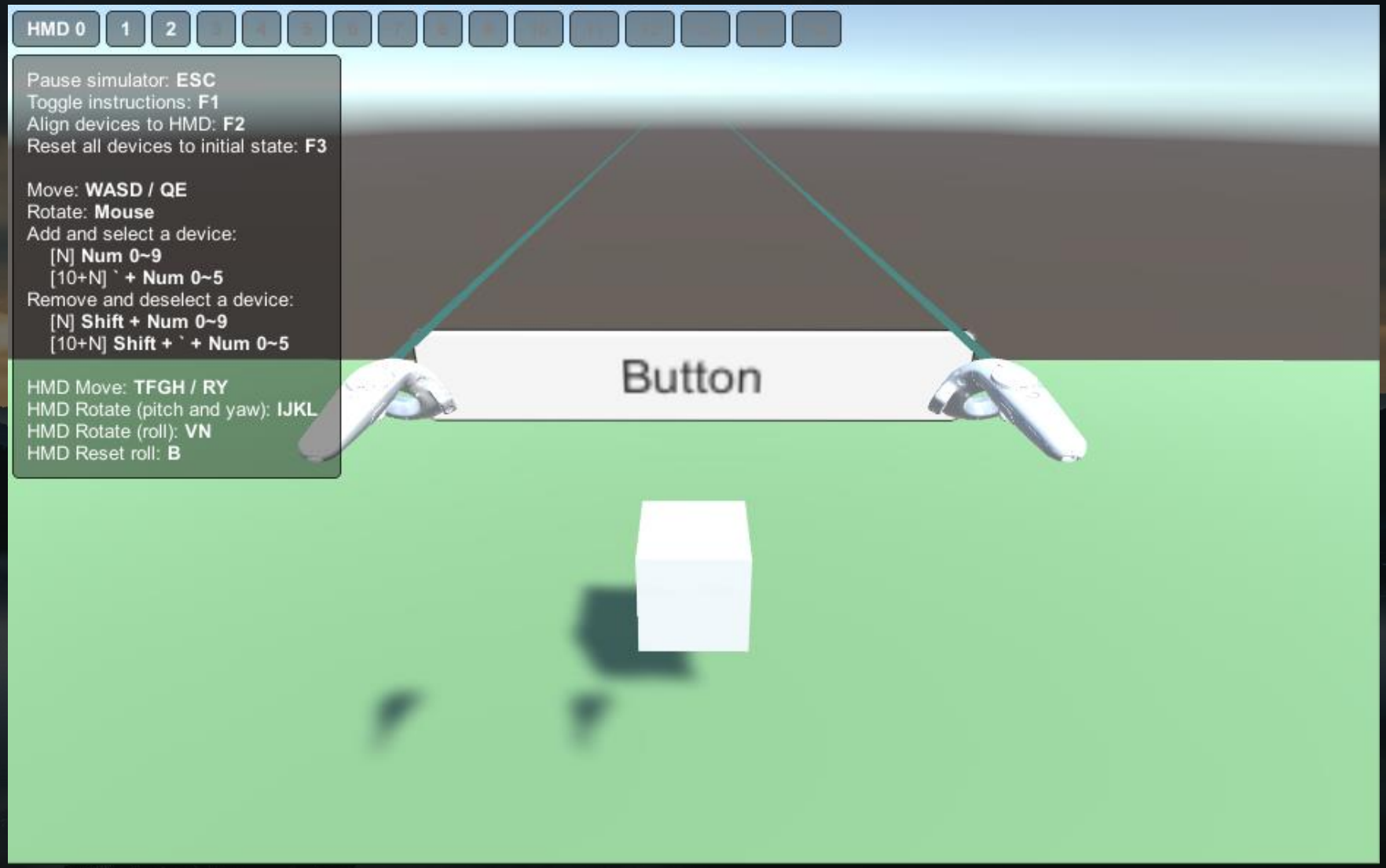
使用VIU模拟器

在Unity Editor中运行/测试

切换到Windows平台

在Unity Editor运行 

使用VIU模拟器


HMD 0
1
2
Pause simulator: ESC
Toggle instructions: F1
Align devices to HMD: F2
Reset all devices to initial state: F3
Move: WASD / QE
Rotate: Mouse
Add and select a device:
[N] Num 0~9
[10+N] ` + Num 0~5
Remove and deselect a device:
[N] Shift + Num 0~9
[10+N] Shift + ` + Num 0~5
HMD Move: TFGH / RY
HMD Rotate (pitch and yaw): IJKL
HMD Rotate (roll): VN
HMD Reset roll: B
Button

VIU示例场景

HTC.UnityPlugin/ViveInputUtility/Examples目录

1.UGUI：手柄与UI交互
4.Teleport：传送
5.ColliderEvent：3D物体交互

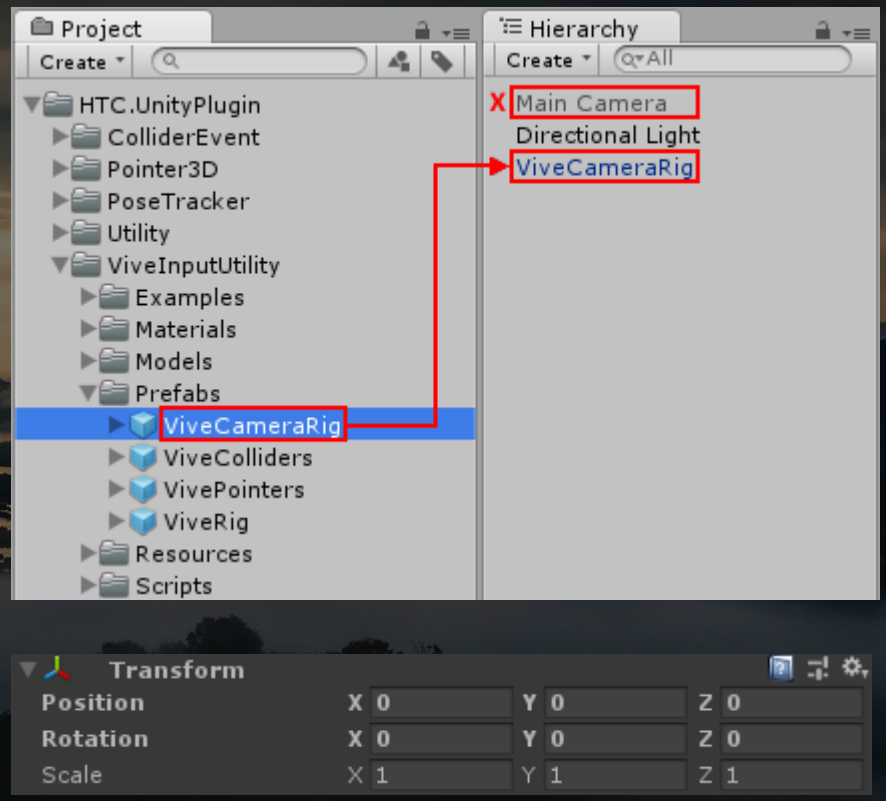
2.2DDragDrop：2D拖拽
3.3DDragDrop：3D拖拽

在场景中使用VIU

新建场景

用ViveCameraRig替换Main Camera

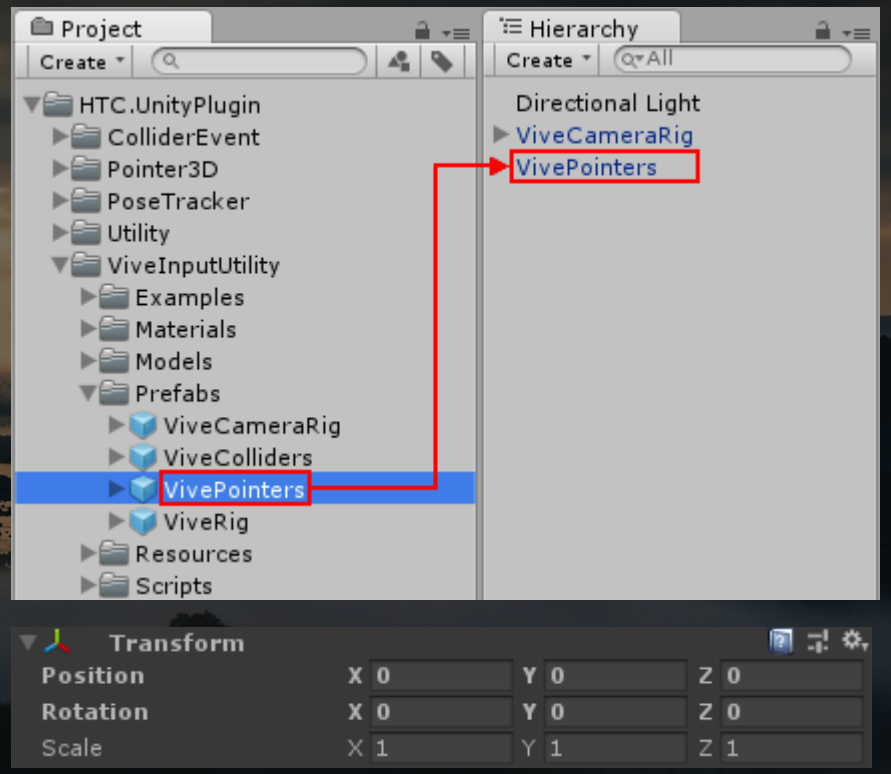
设置ViveCameraRig的位置



手柄与UI交互

在场景中添加VivePointers用于UI交互

设置VivePointers的位置

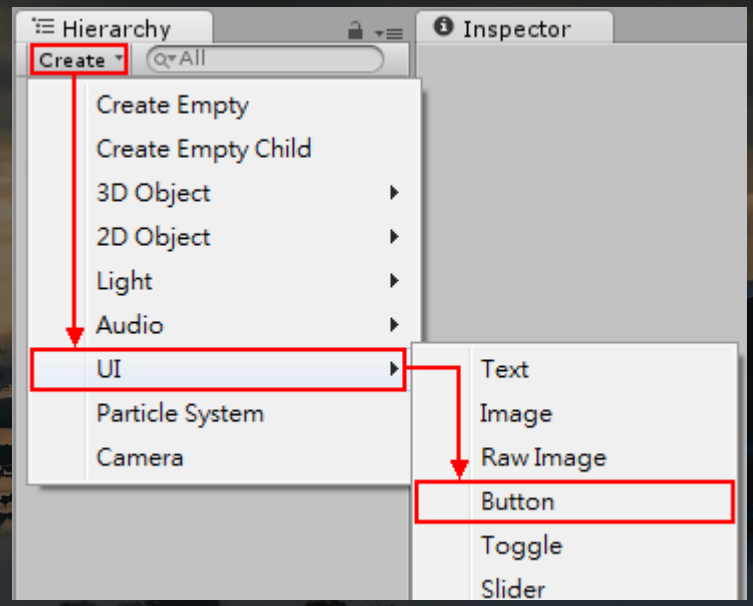


手柄与UI交互

在场景中添加VivePointers用于UI交互

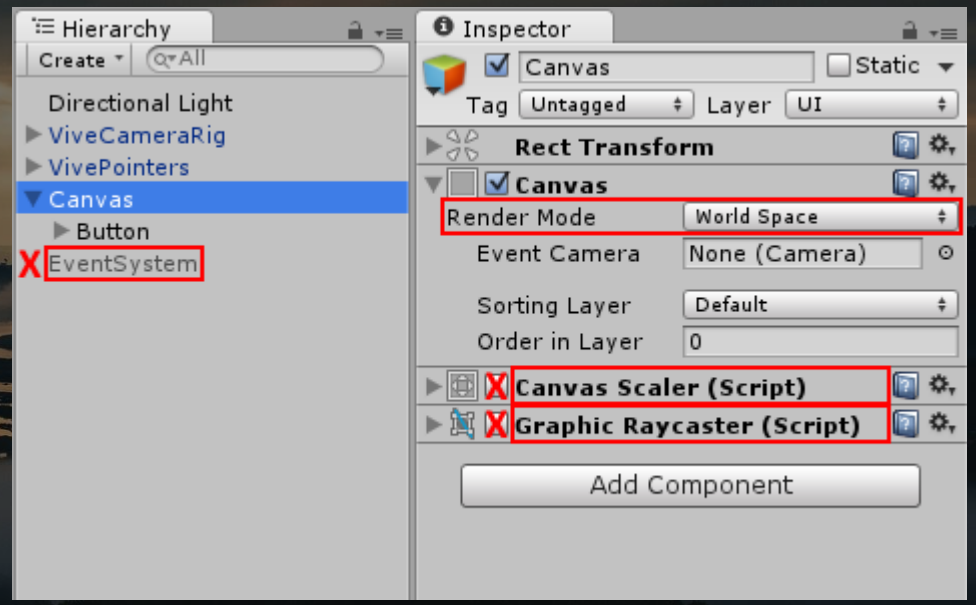
设置VivePointers的位置

添加一个UI按钮



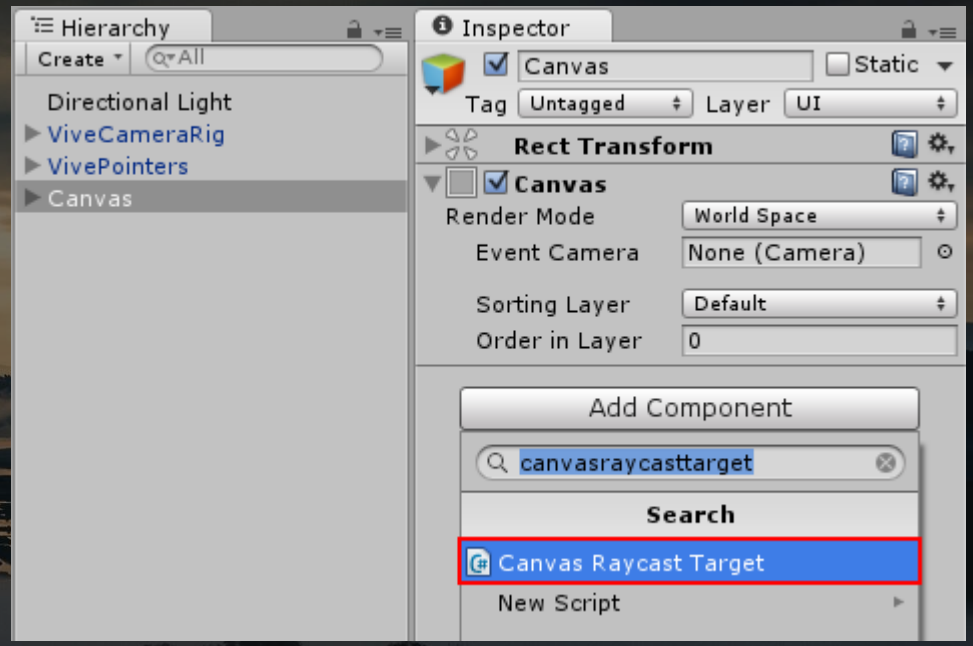
手柄与UI交互

在场景中添加VivePointers用于UI交互
　　设置VivePointers的位置
添加一个UI按钮
修改Canvas设置



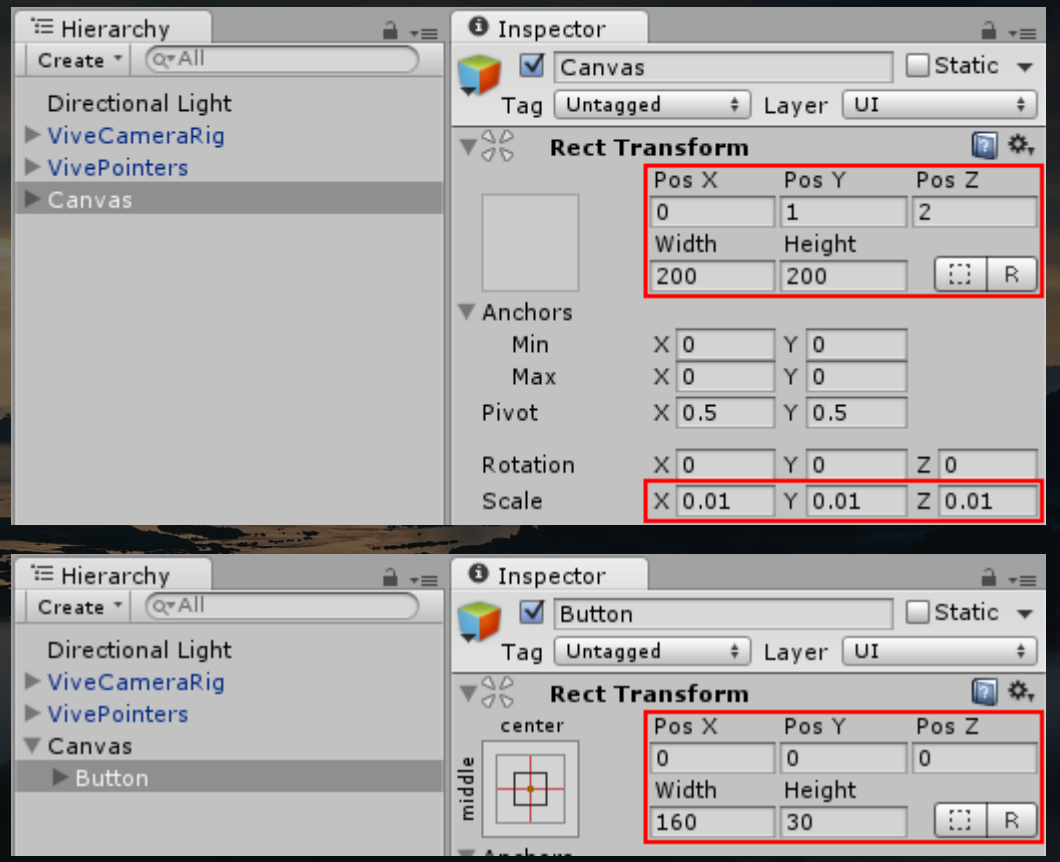
手柄与UI交互

在场景中添加VivePointers用于UI交互
　　设置VivePointers的位置
添加一个UI按钮
修改Canvas设置
添加CanvasRaycastTarget脚本



手柄与UI交互

在场景中添加VivePointers用于UI交互
　　设置VivePointers的位置
添加一个UI按钮
修改Canvas设置
添加CanvasRaycastTarget脚本
把Canvas和button移到可见范围



手柄与UI交互

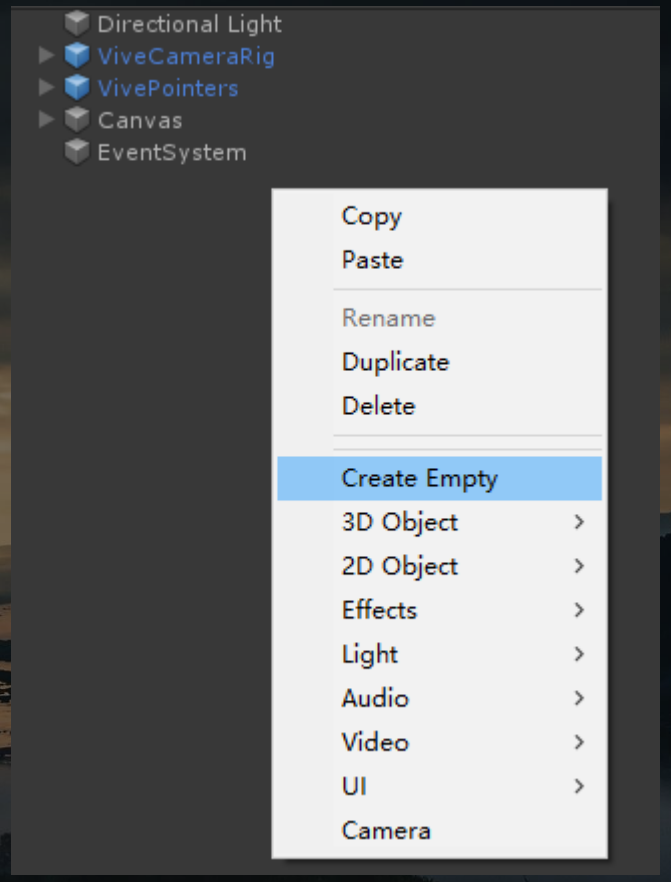
在场景中添加VivePointers用于UI交互
    设置VivePointers的位置
添加一个UI按钮
修改Canvas设置
添加CanvasRaycastTarget脚本
把Canvas和button移到可见范围

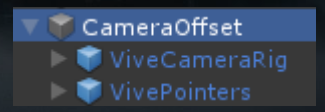
打包运行，用手柄指向按钮，点击trigger键

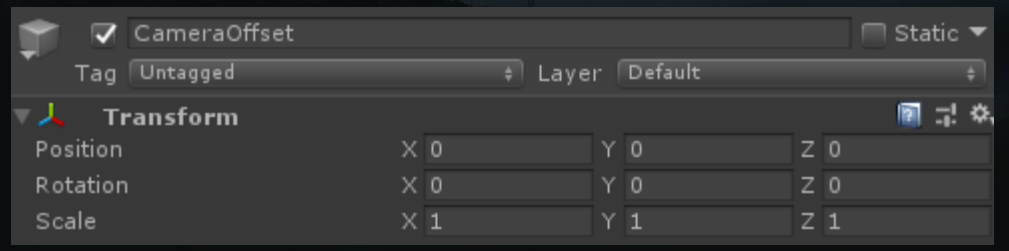
传送

添加位移物件

移动ViveCameraRig和VivePointers物件





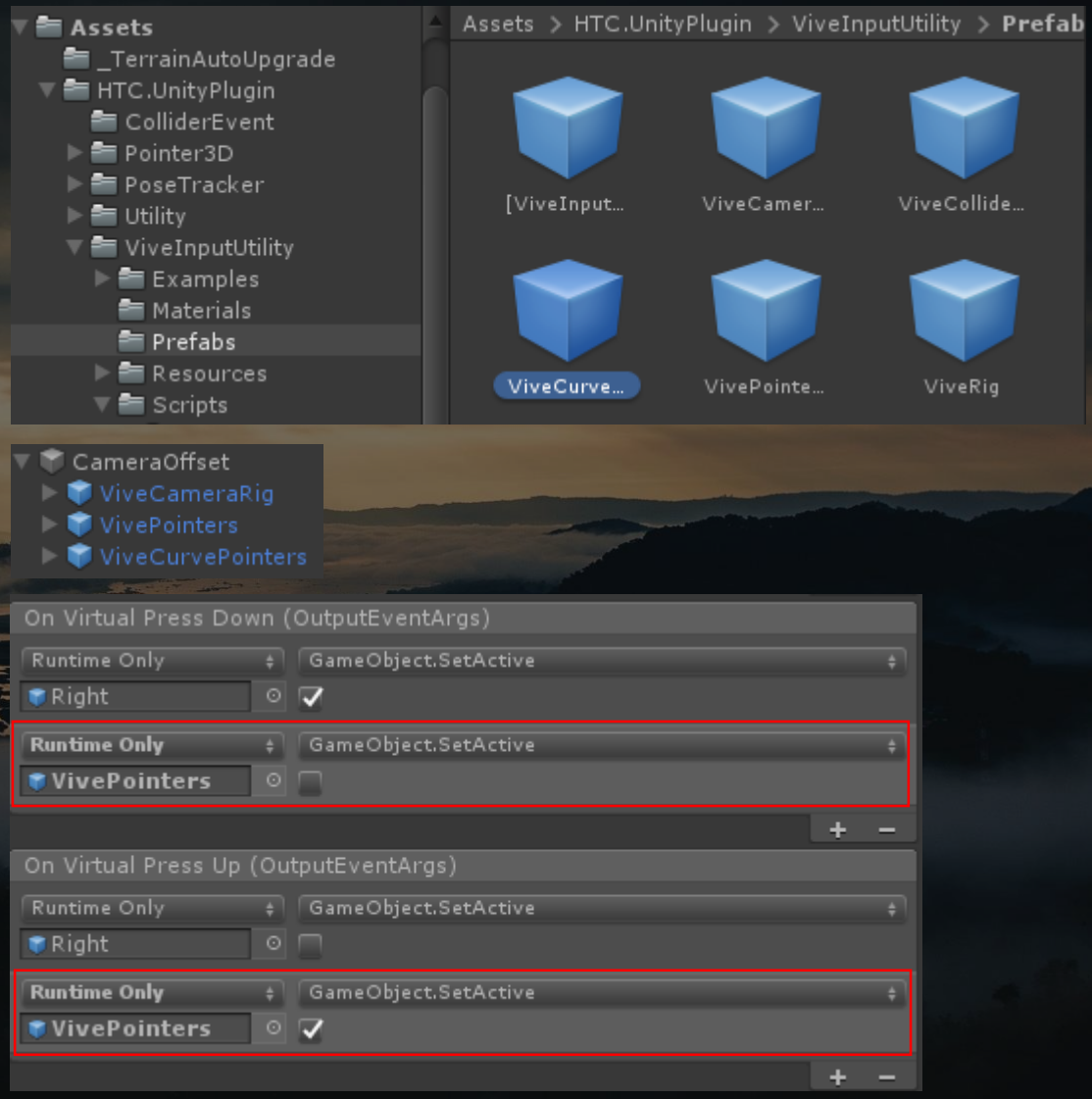


传送

添加位移物件

移动ViveCameraRig和VivePointers物件

添加ViveCurvePointers

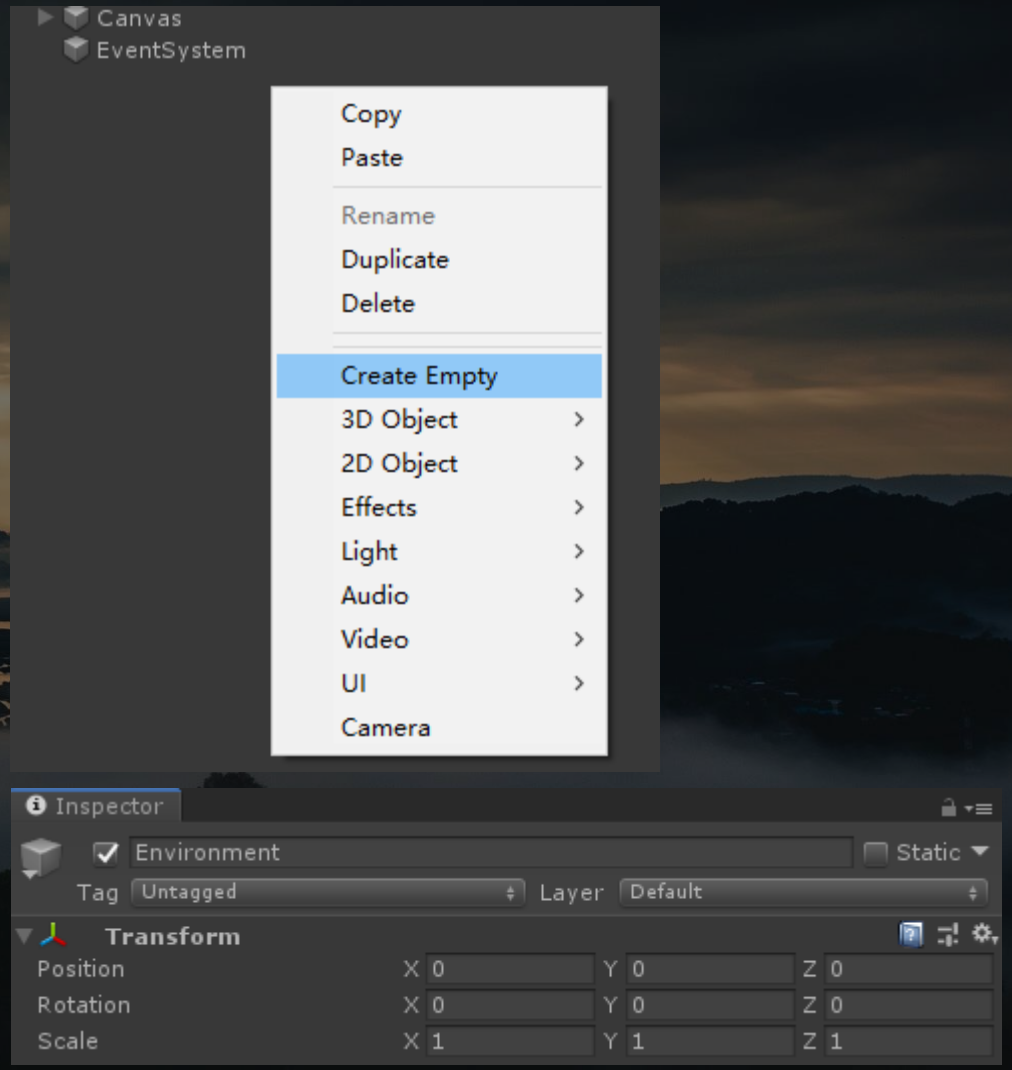


传送

添加位移物件

- 移动ViveCameraRig和VivePointers物件
- 添加ViveCurvePointers

添加环境



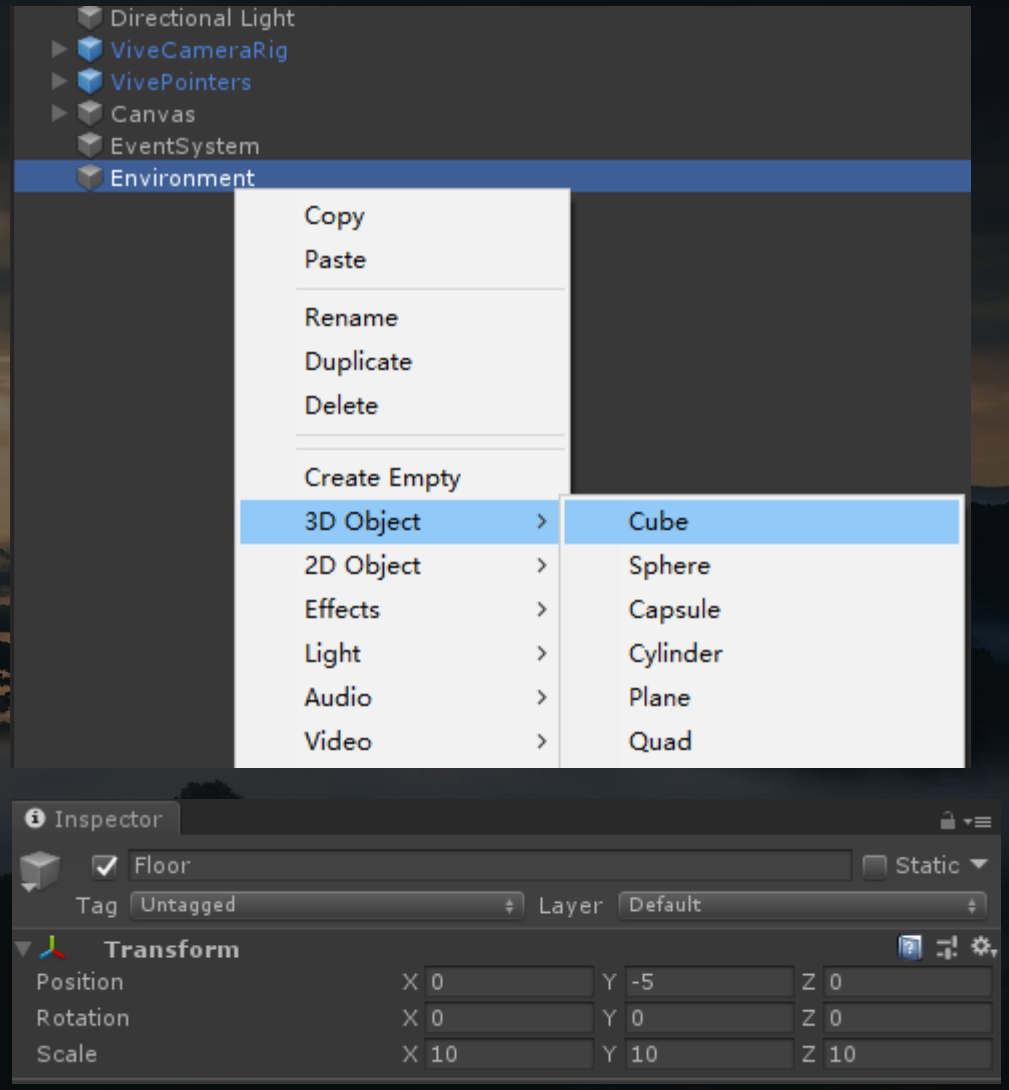
传送

添加位移物件

- 移动ViveCameraRig和VivePointers物件
- 添加ViveCurvePointers

添加环境

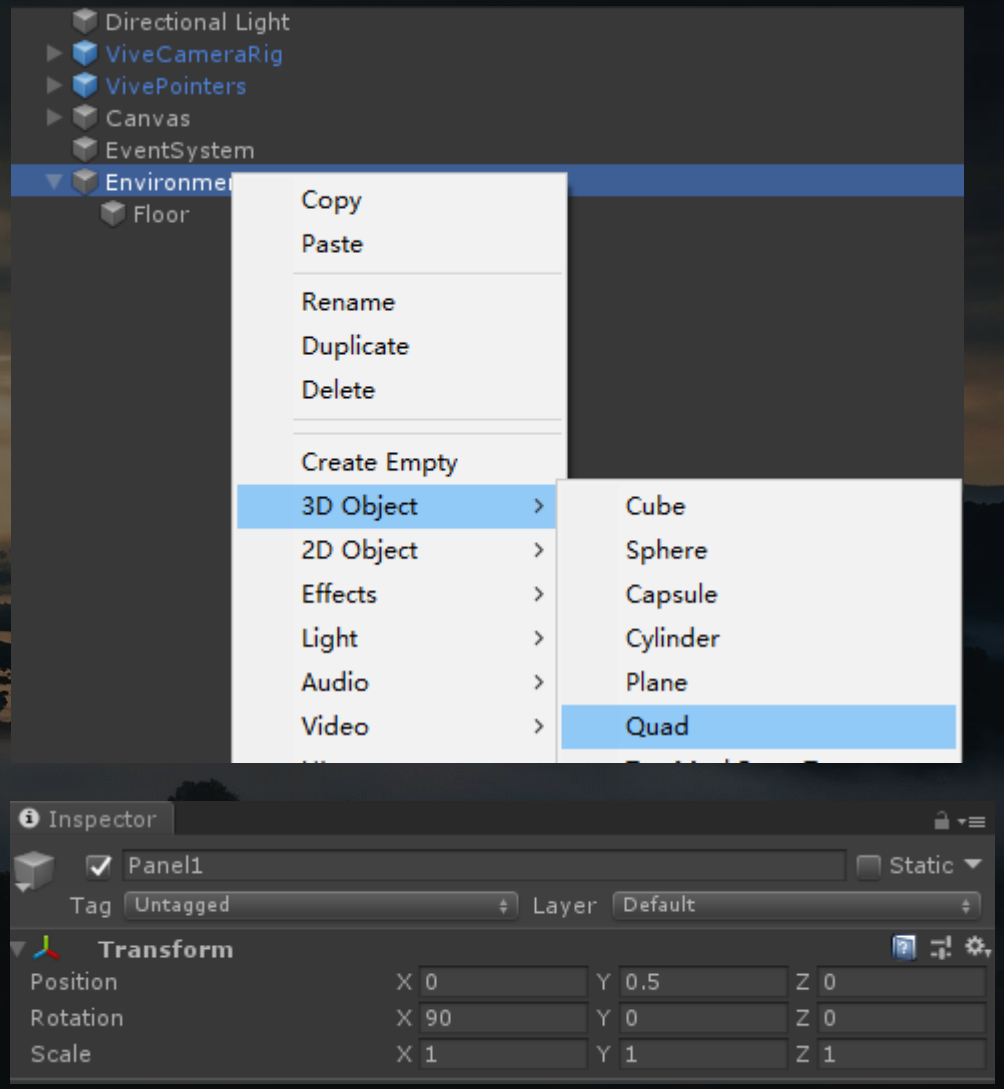
- 添加地板



传送

添加位移物件
- 移动ViveCameraRig和VivePointers物件
- 添加ViveCurvePointers

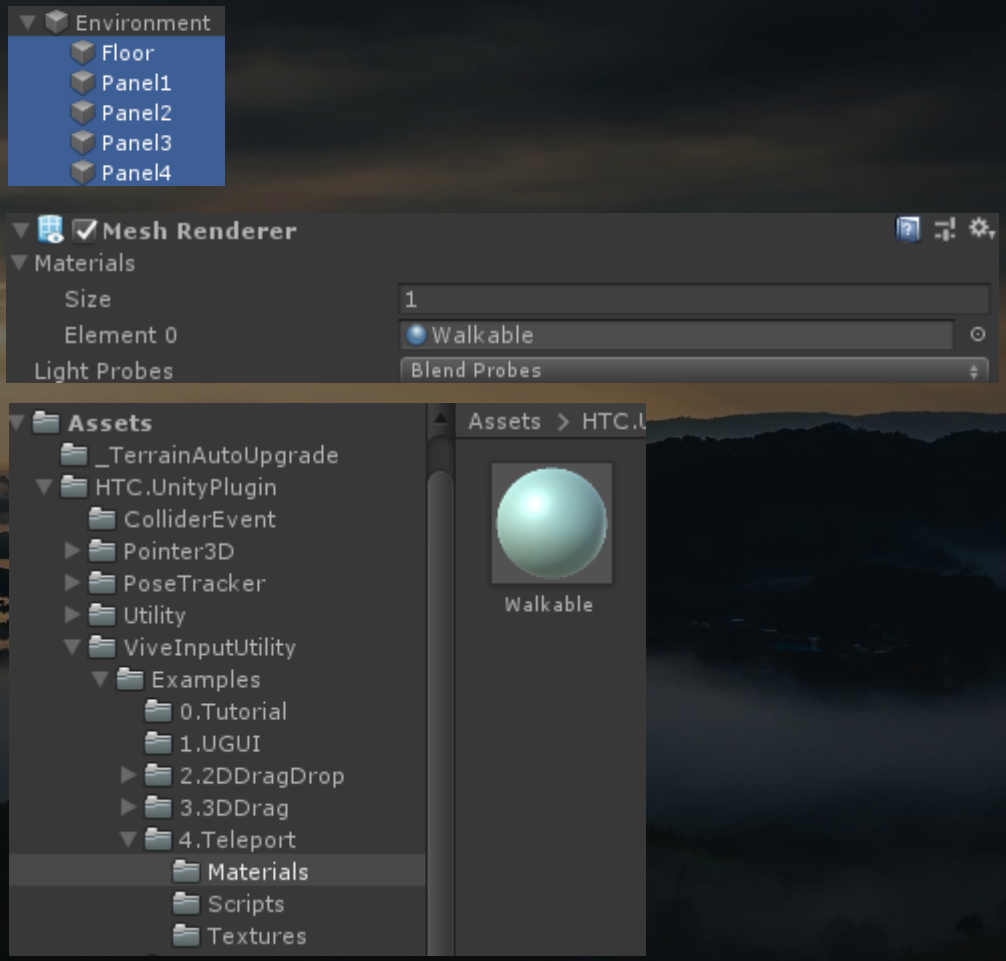
添加环境
- 添加地板
- 添加若干空中的踏板



传送

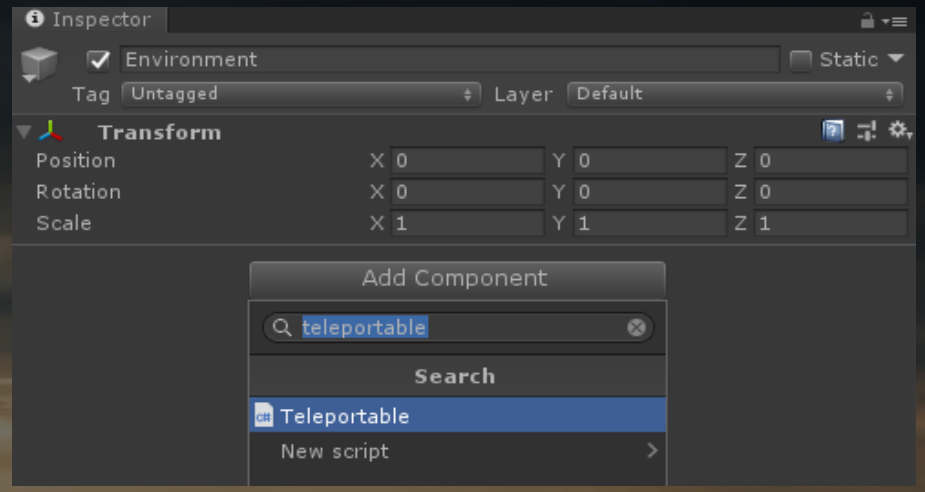
添加位移物件
- 移动ViveCameraRig和VivePointers物件
- 添加ViveCurvePointers

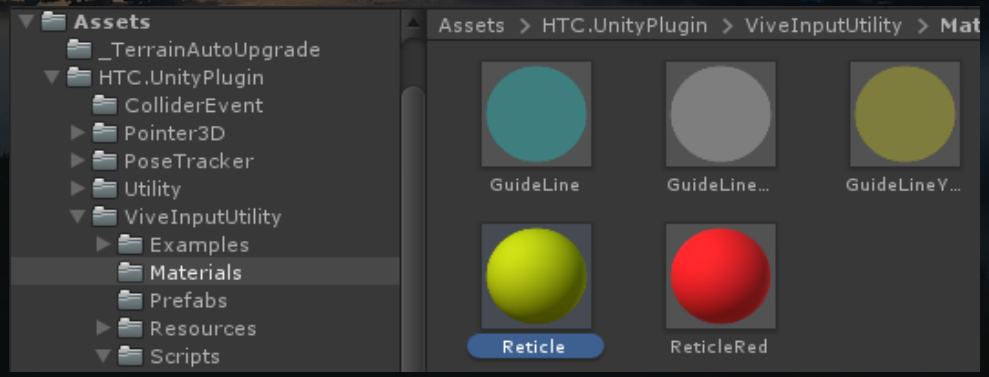
添加环境
- 添加地板
- 添加若干空中的踏板
- 设置地板和踏板颜色



传送

- 添加位移物件
  - 移动ViveCameraRig和VivePointers物件
  - 添加ViveCurvePointers
- 添加环境
  - 添加地板
  - 添加若干空中的踏板
  - 设置地板和踏板颜色
- 添加传送脚本(Teleportable)





传送

添加位移物件
- 移动ViveCameraRig和VivePointers物件
- 添加ViveCurvePointers

添加环境
- 添加地板
- 添加若干空中的踏板
- 设置地板和踏板颜色

添加传送脚本(Teleportable)

添加不可传送的区域
- 在Environment外添加两个Cube

传送

添加位移物件
- 移动ViveCameraRig和VivePointers物件
- 添加ViveCurvePointers

添加环境
- 添加地板
- 添加若干空中的踏板
- 设置地板和踏板颜色

添加传送脚本(Teleportable)

添加不可传送的区域
- 在Environment外添加两个Cube

打包运行，按住触摸板瞄准，松开进行传送

3D物体交互

抓取物体

按住抓取，松开释放

抓住物体

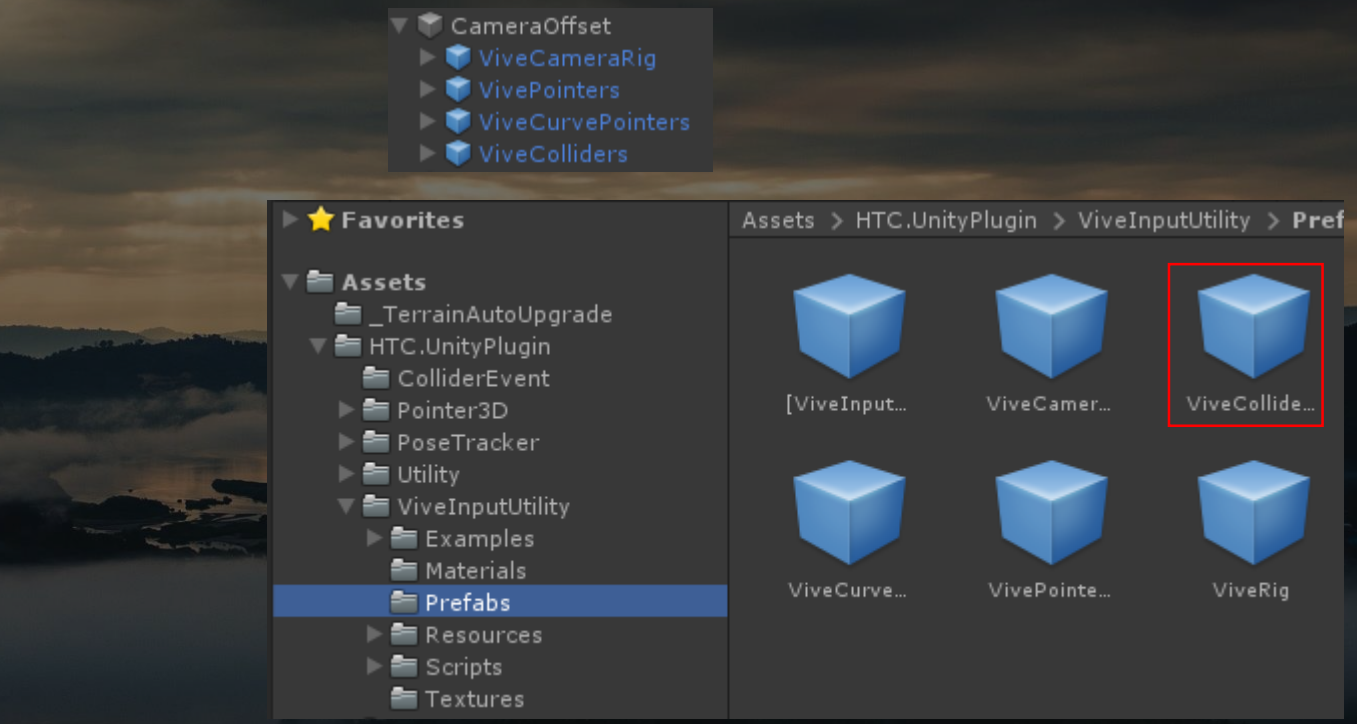
按一次抓取，再按一次释放

开关

手柄触碰并点击按钮触发

物体交互

添加ViveColliders



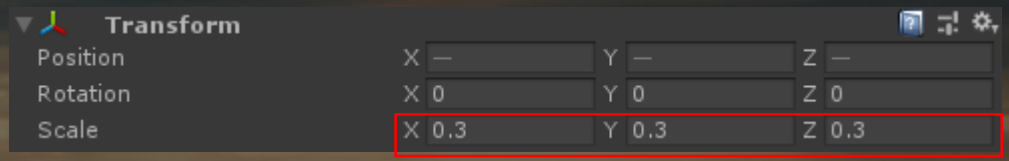
物体交互

添加ViveColliders

使用传送时最后添加的两个cube

可以改名为Grab1和Grab2

把scale调小到0.3

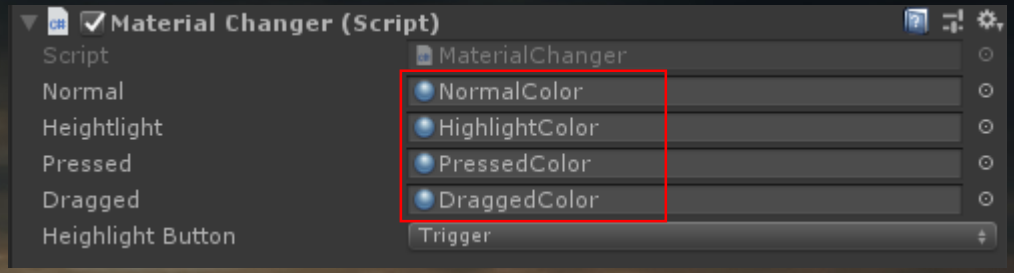


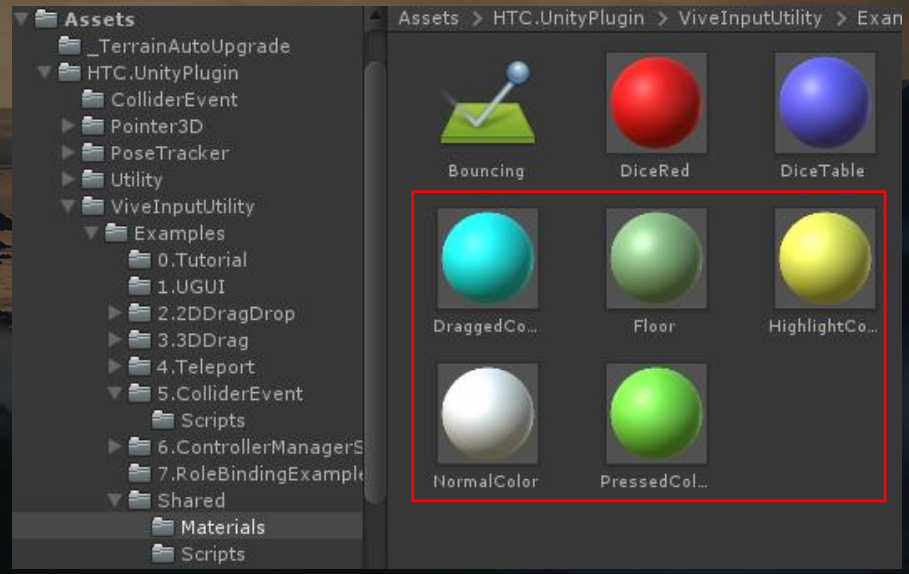
物体交互

- 添加ViveColliders
- 使用传送时最后添加的两个cube
  - 可以改名为Grab1和Grab2
  - 把scale调小到0.3
- 为Grab1添加Basic Grabbable脚本
- 为Grab2添加Sticky Grabbable脚本

物体交互

- 添加ViveColliders
- 使用传送时最后添加的两个cube
  - 可以改名为Grab1和Grab2
  - 把scale调小到0.3
- 为Grab1添加Basic Grabbable脚本
- 为Grab2添加Sticky Grabbable脚本
- 为Grab1和Grab2添加Material Changer脚本





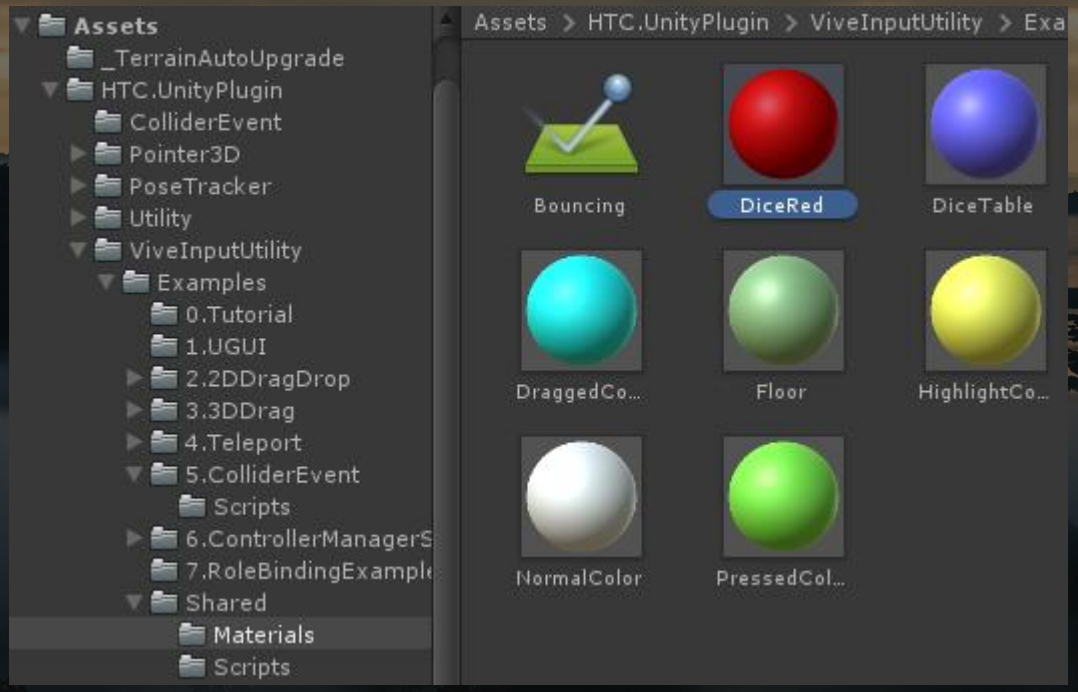
物体交互

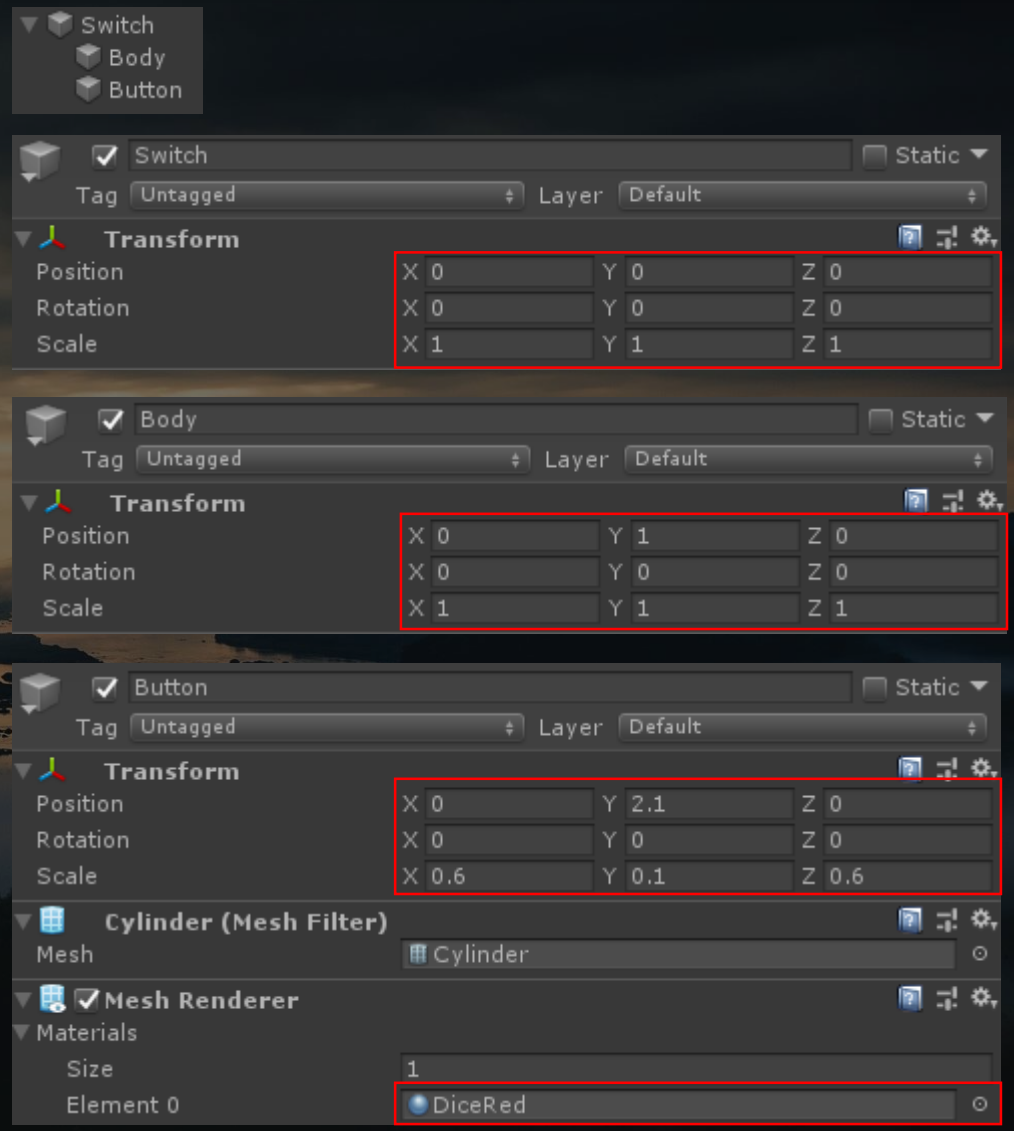
- 添加ViveColliders
- 使用传送时最后添加的两个cube
  - 可以改名为Grab1和Grab2
  - 把scale调小到0.3
- 为Grab1添加Basic Grabbable脚本
- 为Grab2添加Sticky Grabbable脚本
- 为Grab1和Grab2添加Material Changer脚本

- 打包运行，传送到Grab1和Grab2附近，用扳机键抓取

开关交互

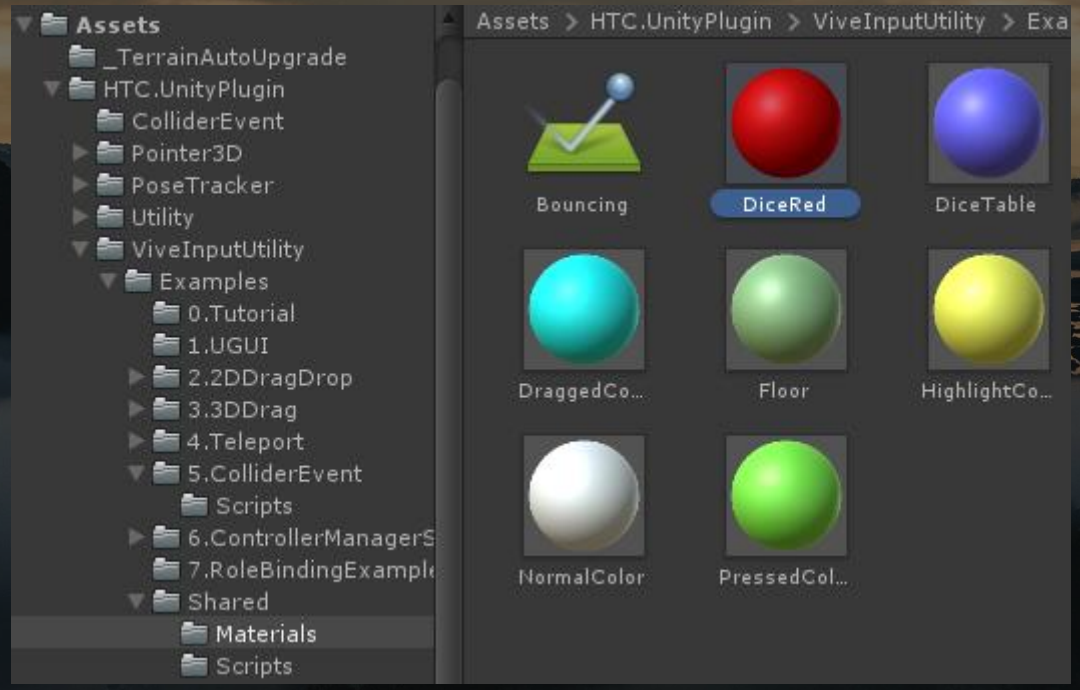
- 添加一个空GameObject，命名为Switch
  - Switch里添加两个Cylinder：Body和Button

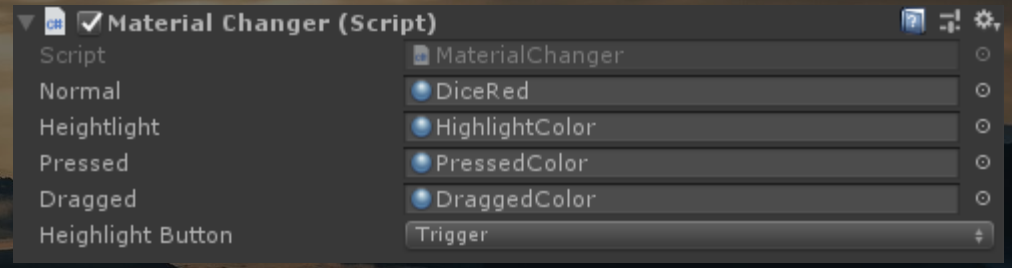




开关交互

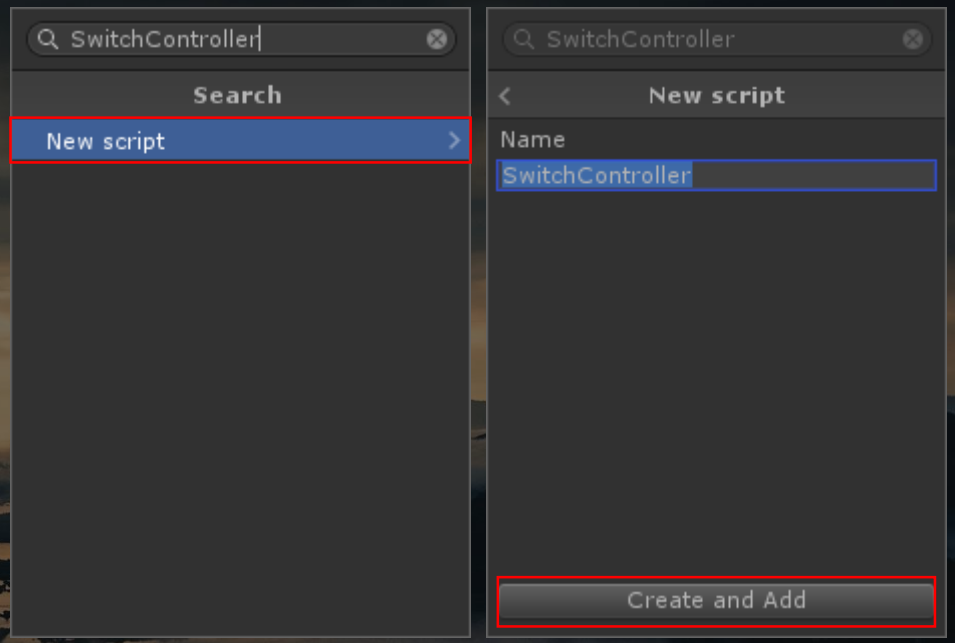
- 添加一个空GameObject，命名为Switch
  - Switch里添加两个Cylinder：Body和Button
  - 在Button上添加一个Material Changer





开关交互

- 添加一个空GameObject，命名为Switch
  - Switch里添加两个Cylinder：Body和Button
  - 在Button上添加一个Material Changer
- 在Switch上新建一个脚本SwitchController
  - 脚本生成在Assets目录下
  - 实现按下按钮时开灯/关灯效果



```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
  IColliderEventPressUpHandler,
  IColliderEventPressEnterHandler,
  IColliderEventPressExitHandler {
}
```

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
}
```

Unity 脚本基类

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
}
```

按钮点击事件接口

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
}
```

按钮按下事件接口

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
  IColliderEventPressUpHandler,
  IColliderEventPressEnterHandler,
  IColliderEventPressExitHandler {
}
```

按钮释放事件接口

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton =
  ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
}
```

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton =
  ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
}
```

类的公开属性会在Inspector中显示

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton =
  ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
}
```

选择光源，用于修改亮度

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton =
  ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
}
```

选择按钮位置，用于按钮按下的动画效果

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton =
  ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
}
```

动画效果时按钮位置的额外位移

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton =
  ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
}
```

触发开关的手柄按钮

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour,
 IColliderEventPressUpHandler,
 IColliderEventPressEnterHandler,
 IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton =
  ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
}
```

局部变量，缓存当前触发的事件

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;
public class SwitchController : MonoBehaviour, IColliderEventPressUpHandler, IColliderEventPressEnterHandler, IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton = ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;

 public void OnColliderEventPressEnter(ColliderButtonEventData eventData) {
  if (eventData.button == activeButton && activeEvent == null) {
   activeEvent = eventData;
   button.localPosition += buttonOffset;
  }
 }
 public void OnColliderEventPressExit(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) {
   activeEvent = null;
   button.localPosition -= buttonOffset;
  }
 }
```

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;
public class SwitchController : MonoBehaviour, IColliderEventPressUpHandler, IColliderEventPressEnterHandler, IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton = ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
```

```
public void OnColliderEventPressEnter(ColliderButtonEventData eventData) {
 if (eventData.button == activeButton && activeEvent == null) {
  activeEvent = eventData;
  button.localPosition += buttonOffset;
 }
}
```

IColliderEventPressEnterHandler接口的函数实现

```
public void OnColliderEventPressExit(ColliderButtonEventData eventData) {
 if (activeEvent == eventData) {
  activeEvent = null;
  button.localPosition -= buttonOffset;
 }
}
```

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;
public class SwitchController : MonoBehaviour, IColliderEventPressUpHandler, IColliderEventPressEnterHandler, IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton = ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
```

```
public void OnColliderEventPressEnter(ColliderButtonEventData eventData) {
 if (eventData.button == activeButton && activeEvent == null) {
  activeEvent = eventData;
  button.localPosition += buttonOffset;
 }
}
public void OnColliderEventPressExit(ColliderButtonEventData eventData) {
 if (activeEvent == eventData) {
  activeEvent = null;
  button.localPosition -= buttonOffset;
 }
}
```

IColliderEventPressExitHandler接口的函数实现

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;
public class SwitchController : MonoBehaviour, IColliderEventPressUpHandler, IColliderEventPressEnterHandler, IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton = ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;

 public void OnColliderEventPressEnter(ColliderButtonEventData eventData) {
  if (eventData.button == activeButton && activeEvent == null) {
   activeEvent = eventData;
   button.localPosition += buttonOffset;
  }
 }
 public void OnColliderEventPressExit(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) {
   activeEvent = null;
   button.localPosition -= buttonOffset;
  }
 }
```

如果手柄按钮吻合并且没有触发中的事件

1. 缓存触发中的事件
2. 修改按钮位置(动画效果)

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;
public class SwitchController : MonoBehaviour, IColliderEventPressUpHandler, IColliderEventPressEnterHandler, IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton = ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;

 public void OnColliderEventPressEnter(ColliderButtonEventData eventData) {
  if (eventData.button == activeButton && activeEvent == null) {
   activeEvent = eventData;
   button.localPosition += buttonOffset;
  }
 }
 public void OnColliderEventPressExit(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) {
   activeEvent = null;
   button.localPosition -= buttonOffset;
  }
 }
```

如果是触发中的事件:

1. 清楚触发中的事件
2. 修改按钮位置(动画效果)

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour, IColliderEventPressUpHandler, IColliderEventPressEnterHandler, IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton = ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
 public void OnColliderEventPressEnter(ColliderButtonEventData eventData) {
  if (eventData.button == activeButton && activeEvent == null) {
   activeEvent = eventData;
   button.localPosition += buttonOffset;
  }
 }
 public void OnColliderEventPressExit(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) {
   activeEvent = null;
   button.localPosition -= buttonOffset;
  }
 }
 public void OnColliderEventPressUp(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) light.intensity = 1 - light.intensity;
 }
```

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour, IColliderEventPressUpHandler, IColliderEventPressEnterHandler, IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton = ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
 public void OnColliderEventPressEnter(ColliderButtonEventData eventData) {
  if (eventData.button == activeButton && activeEvent == null) {
   activeEvent = eventData;
   button.localPosition += buttonOffset;
  }
 }
 public void OnColliderEventPressExit(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) {
   activeEvent = null;
   button.localPosition -= buttonOffset;
  }
 }
```

```
public void OnColliderEventPressUp(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) light.intensity = 1 - light.intensity;
}
```

IColliderEventPressUpHandler接口的函数实现

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;

public class SwitchController : MonoBehaviour, IColliderEventPressUpHandler, IColliderEventPressEnterHandler, IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton = ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;
 public void OnColliderEventPressEnter(ColliderButtonEventData eventData) {
  if (eventData.button == activeButton && activeEvent == null) {
   activeEvent = eventData;
   button.localPosition += buttonOffset;
  }
 }
 public void OnColliderEventPressExit(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) {
   activeEvent = null;
   button.localPosition -= buttonOffset;
  }
 }
 public void OnColliderEventPressUp(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) light.intensity = 1 - light.intensity;
 }
}
```

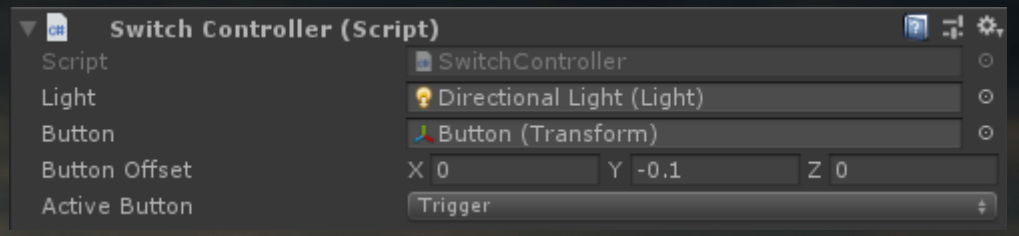
如果按钮事件正确，修改光源的亮度

```
using UnityEngine;
using HTC.UnityPlugin.ColliderEvent;
public class SwitchController : MonoBehaviour, IColliderEventPressUpHandler, IColliderEventPressEnterHandler, IColliderEventPressExitHandler {
 public Light light;
 public Transform button;
 public Vector3 buttonOffset;
 public ColliderButtonEventData.InputButton activeButton = ColliderButtonEventData.InputButton.Trigger;
 private ColliderButtonEventData activeEvent = null;

 public void OnColliderEventPressEnter(ColliderButtonEventData eventData) {
  if (eventData.button == activeButton && activeEvent == null) {
   activeEvent = eventData;
   button.localPosition += buttonOffset;
  }
 }
 public void OnColliderEventPressExit(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) {
   activeEvent = null;
   button.localPosition -= buttonOffset;
  }
 }
 public void OnColliderEventPressUp(ColliderButtonEventData eventData) {
  if (activeEvent == eventData) light.intensity = 1 - light.intensity;
 }
}
```

开关交互

- 设置Switch Controller选项



- 打包运行，传送到开关附近，手柄移动到开关按钮上，按下扳机键触发开关

-


休息一下


美术

03 游戏制作流程概述

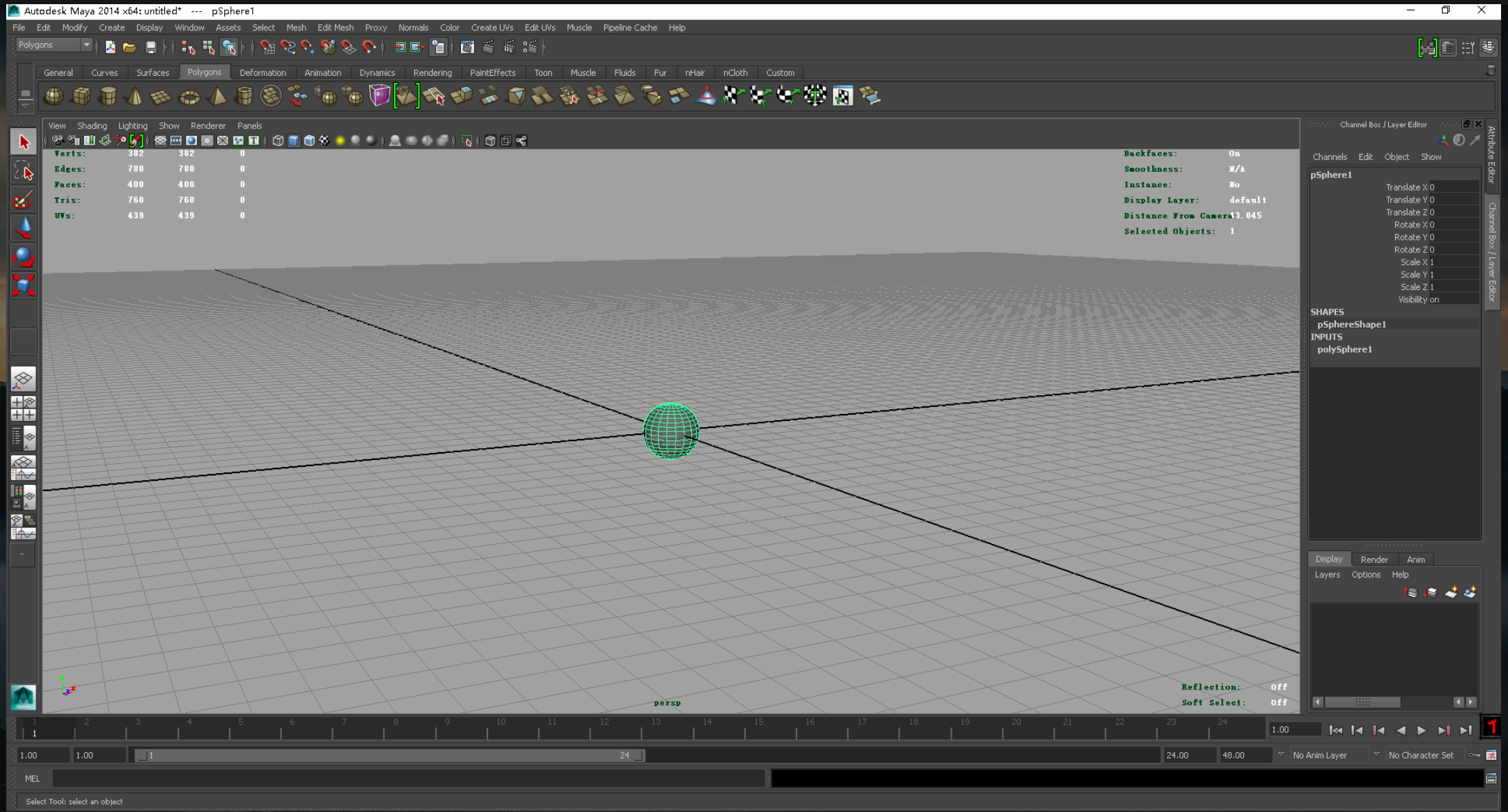
游戏制作流程概述

- 团队配置：策划-美术-程序-音效
- 游戏制作流程：策划-2D-3D-程序-引擎-测试
- 职位：

-策划：主，系统，数值，剧情，关卡，脚本
-2D美术：概念图，角色原画，场景原画，UI
-3D美术：角色，场景，动作
-程序：客户端，服务器
-引擎：地编，TA,特效，音乐，音效
-测试

04 美术资源制作及使用概述

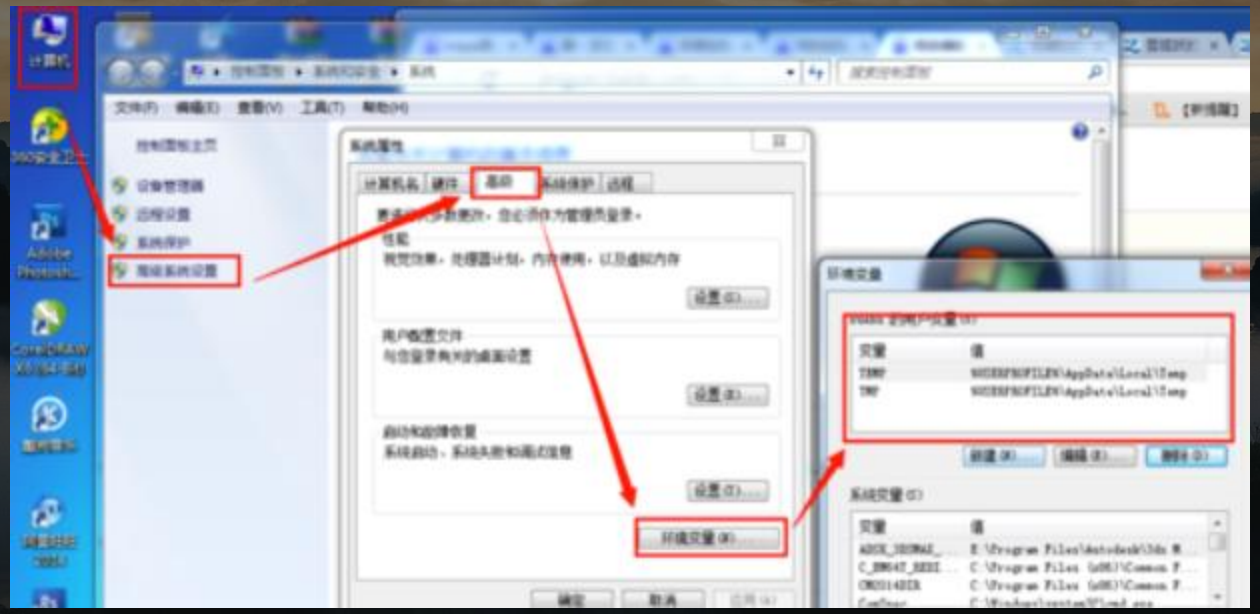
MAYA简介


Autodesk Maya 2014 x64: untitled* --- pSphere1
File Edit Modify Create Display Window Assets Select Mesh Edit Mesh Proxy Normals Color Create UVs Edit UVs Muscle Pipeline Cache Help
Polygons
General Curves Surfaces Polygons Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur nHair nCloth Custom
View Shading Lighting Show Renderer Panels
Verts: 382 382 0
Edges: 780 780 0
Faces: 400 400 0
Tris: 760 760 0
UVs: 439 439 0
Backfaces: On
Smoothness: N/A
Instance: No
Display Layer: default
Distance From Camera 3.045
Selected Objects: 1
persp
Reflection: Off
Soft Select: Off
Channel Box / Layer Editor
Channels Edit Object Show
pSphere1
Translate X 0
Translate Y 0
Translate Z 0
Rotate X 0
Rotate Y 0
Rotate Z 0
Scale X 1
Scale Y 1
Scale Z 1
Visibility on
SHAPES
pSphereShape1
INPUTS
polySphere1
Display Render Anim
Layers Options Help
Attribute Editor
Channel Box / Layer Editor
No Anim Layer
No Character Set
MEL
Select Tool: select an object

MAYA中英文切换

安装后默认英文环境

将maya改成中文版需要右键计算机，属性，高级系统设置，环境变量。

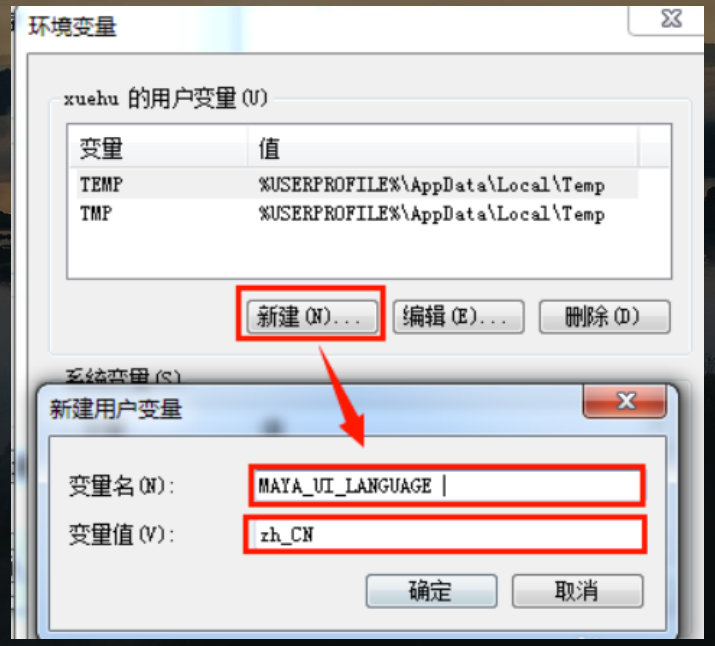


MAYA中英文切换

安装后默认英文环境

将maya改成中文版需要右键计算机，属性，高级系统设置，环境变量

新建一个环境变量，变量名输入MAYA_UI_LANGUAGE，变量值输入zh_CN就是启用中文界面，变量值输入en_US就是启用英文界面



MAYA中英文切换

安装后默认英文环境

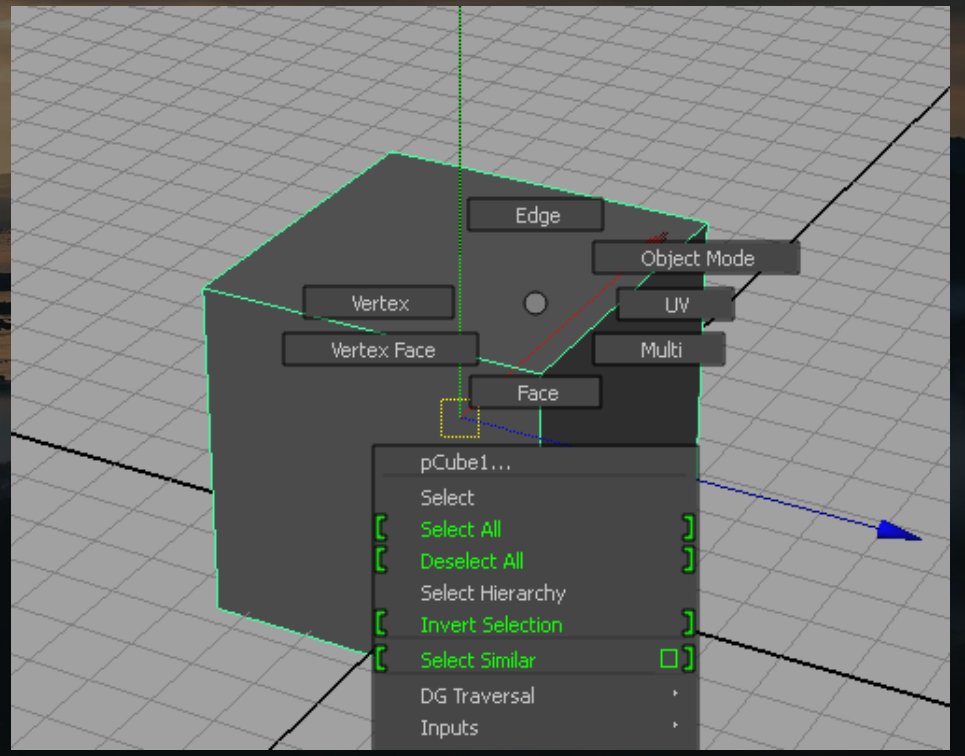
将maya改成中文版需要右键计算机，属性，高级系统设置，环境变量

新建一个环境变量，变量名输入MAYA_UI_LANGUAGE，变量值输入zh_CN就是启用中文界面，变量值输入en_US就是启用英文界面

新建好maya环境变量完成后，然后我们重新启动maya

MAYA操作方式

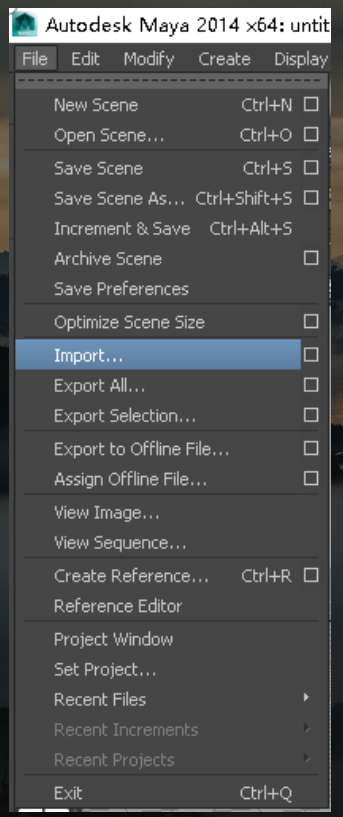
Alt+鼠标左键按住旋转=旋转镜头
鼠标滚轮=Alt+鼠标右键按住平移=推拉摄像机镜头
选中物体F键=聚焦选中物体当前位置
鼠标移至物体按住右键-拖拽-松开鼠标选择编辑模式
选中物体，W=位移，E=旋转，R=缩放

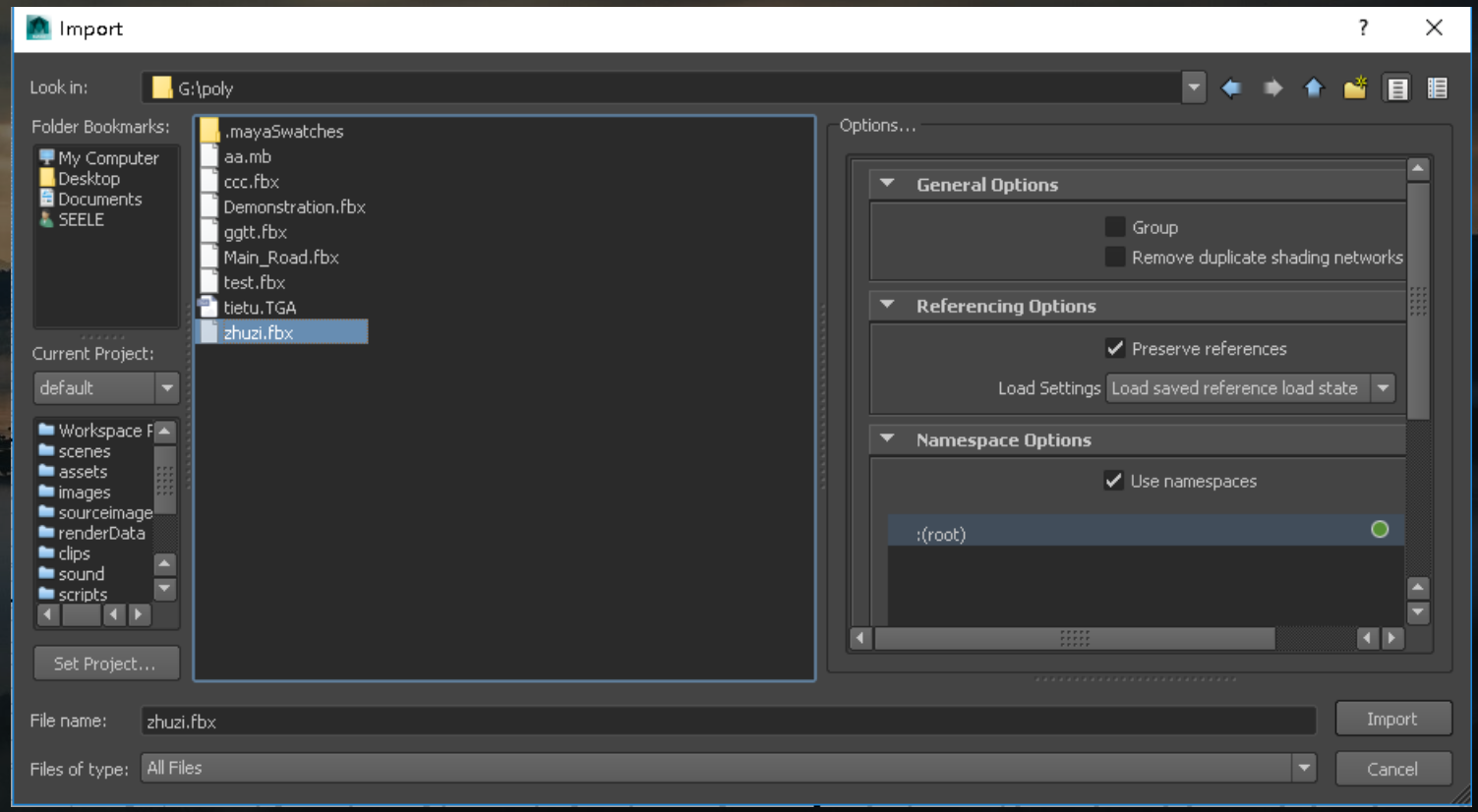


注意中英输入法

MAYA导入导出

导入：文件-导入-选择要导入资源-点击导入

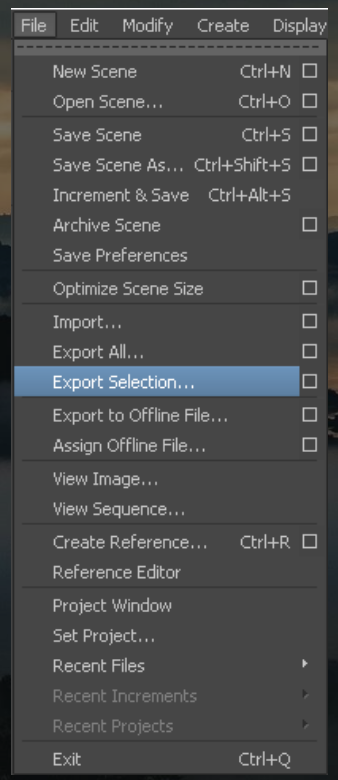


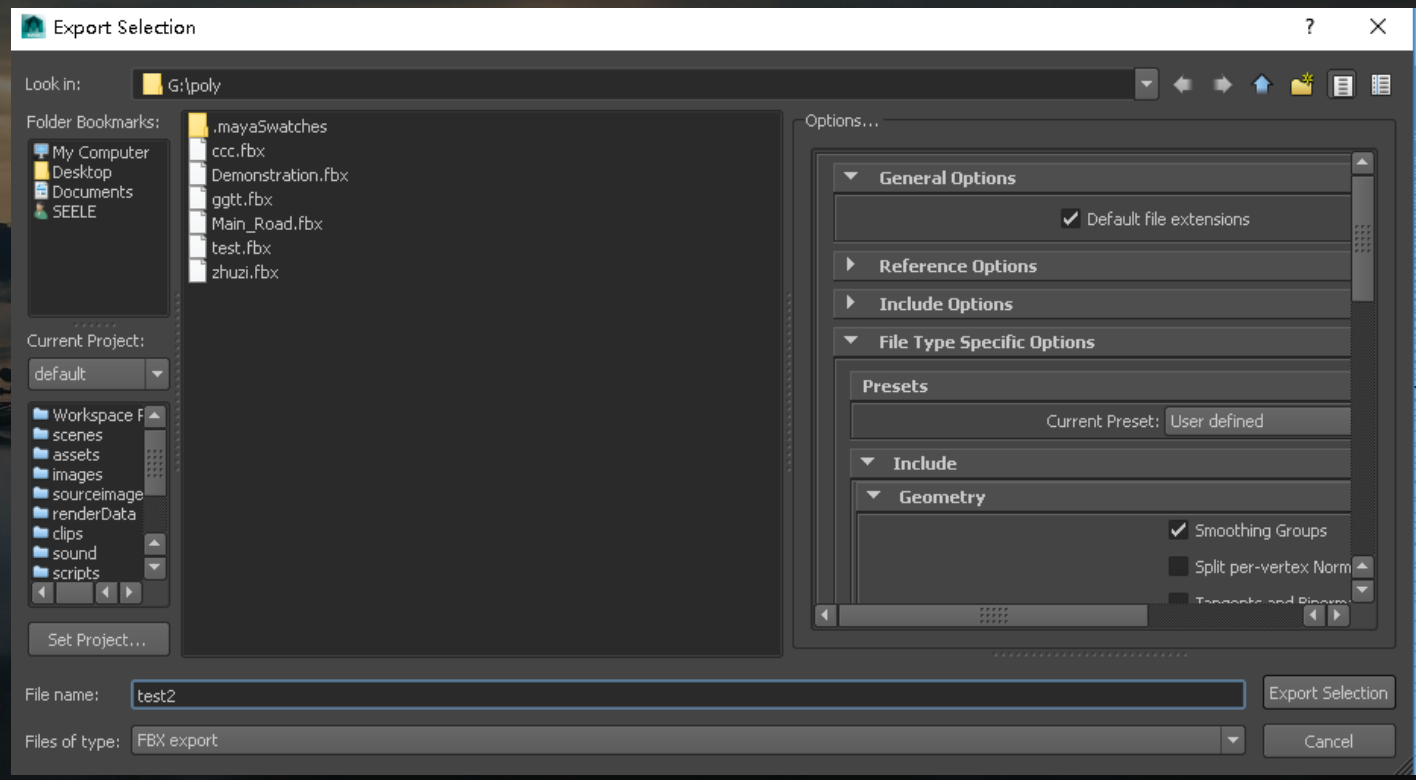


MAYA导入导出

导入：文件-导入-选择要导入资源-点击导入

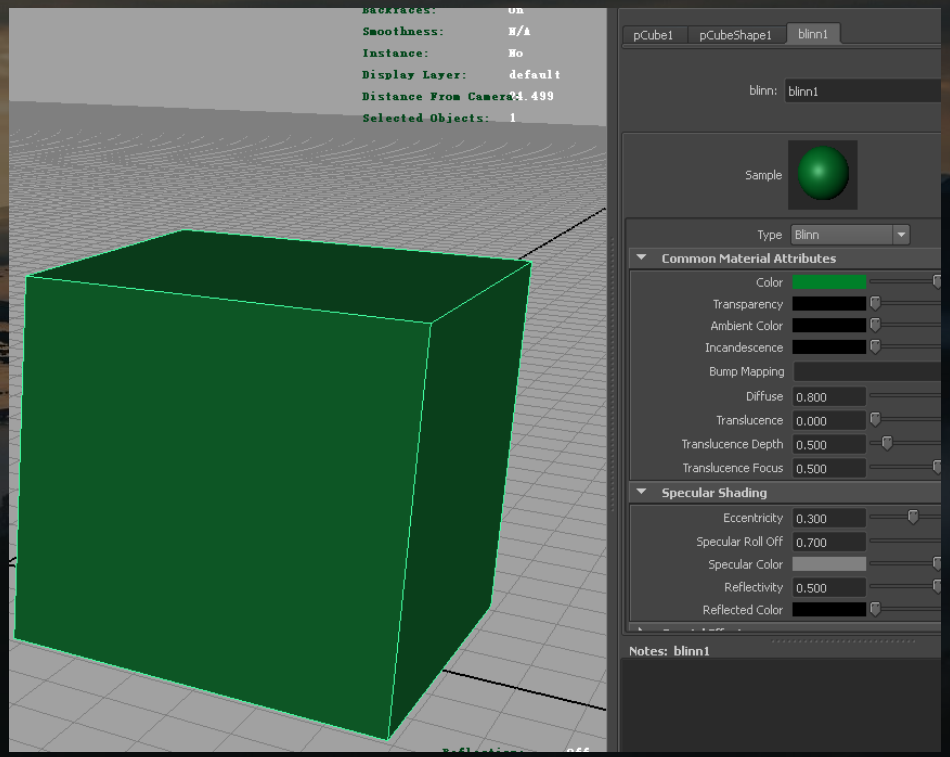
导出：选择要导出物体-文件-导出选择物体-输入名字-点击导出选择





OBJ与FBX文件区别

- OBJ与FBX文件区别
- 建立一个Cube，赋予Blinn材质，颜色调成绿色

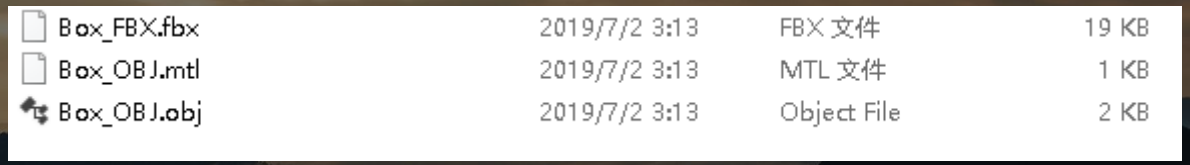


OBJ与FBX文件区别

- OBJ与FBX文件区别
- 建立一个Cube，赋予Blinn材质，颜色调成绿色
- 分别导出为:

-OBJ格式文件Box_OBJ

-FBX格式文件Box_FBX



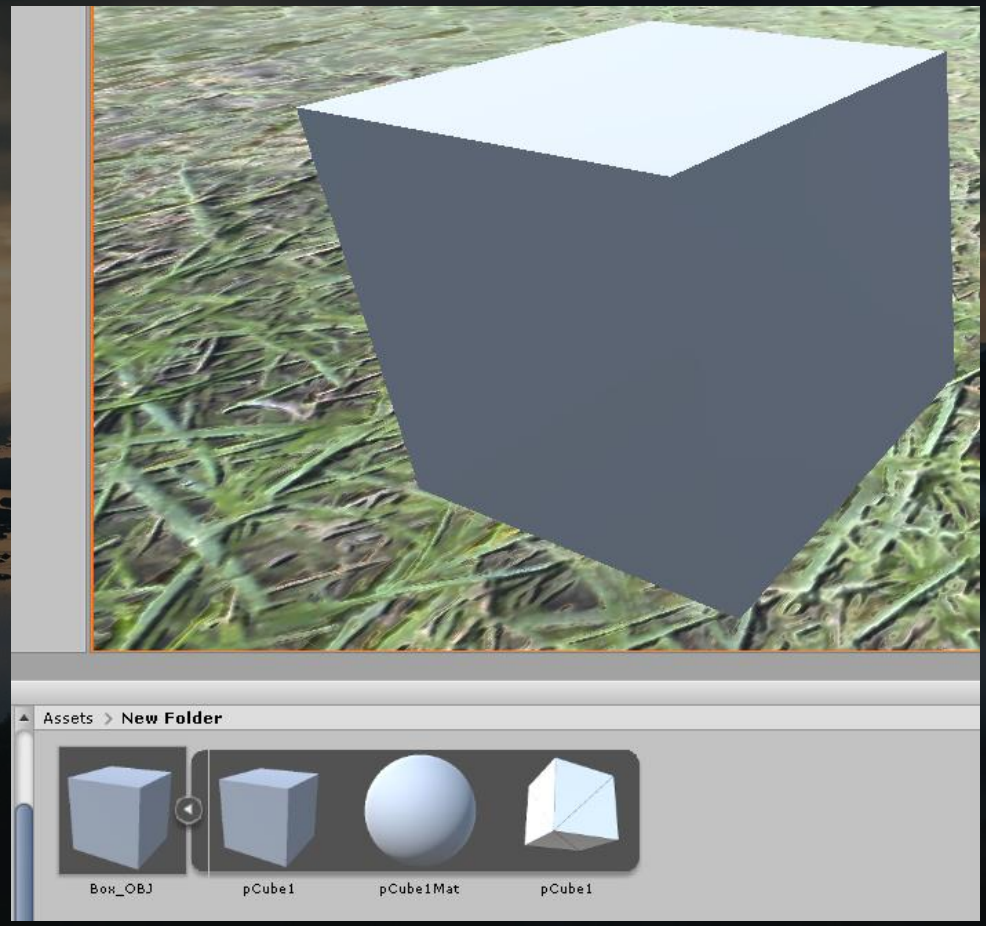
OBJ与FBX文件区别

- OBJ与FBX文件区别
- 建立一个Cube，赋予Blinn材质，颜色调成绿色
- 分别导出为:

-OBJ格式文件Box_OBJ

-FBX格式文件Box_FBX

首先将OBJ文件导入Unity



OBJ与FBX文件区别

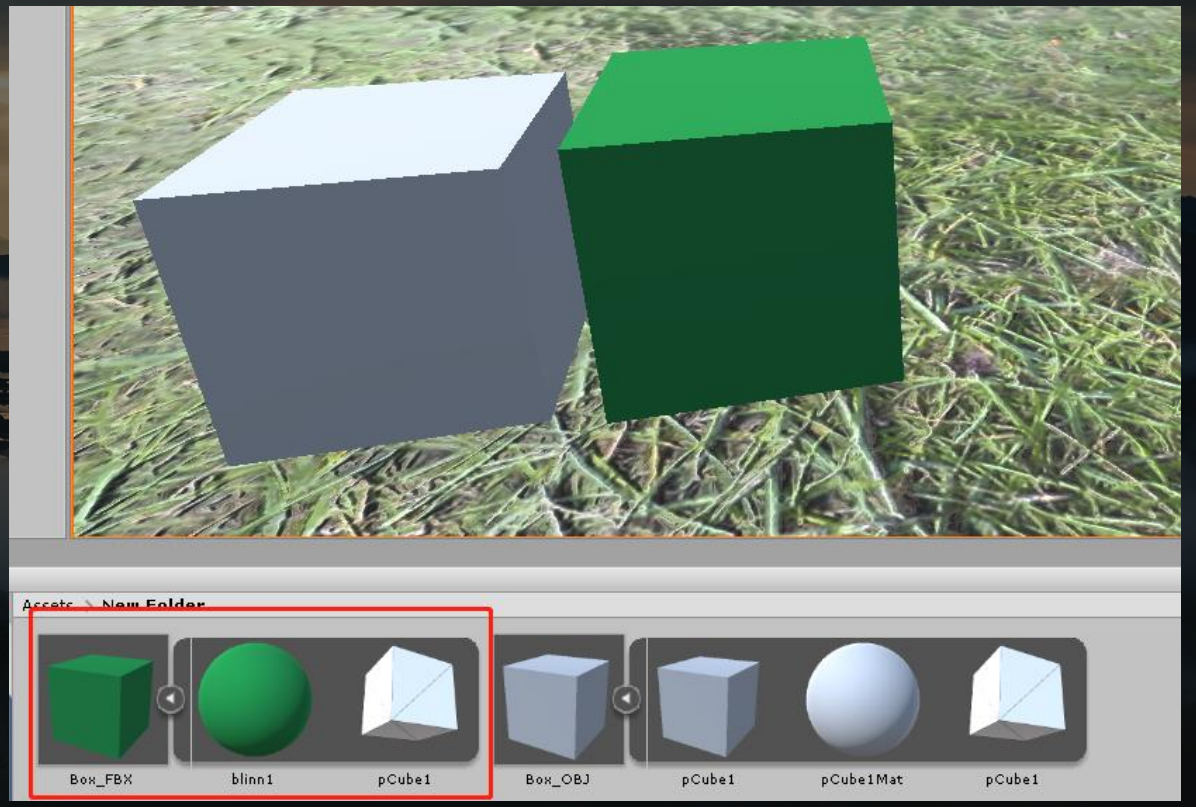
- OBJ与FBX文件区别
- 建立一个Cube，赋予Blinn材质，颜色调成绿色
- 分别导出为:

-OBJ格式文件Box_OBJ

-FBX格式文件Box_FBX

首先将OBJ文件导入Unity

再将FBX文件导入Unity



OBJ与FBX文件区别

- OBJ与FBX文件区别
- 建立一个Cube，赋予Blinn材质，颜色调成绿色
- 分别导出为:

-OBJ格式文件Box_OBJ

-FBX格式文件Box_FBX

首先将OBJ文件导入Unity

再将FBX文件导入Unity

可以发现，FBX继承了MAYA中的被赋予材质球及相关属性，OBJ却没有继承

OBJ与FBX文件区别

- OBJ与FBX文件区别
- 建立一个Cube，赋予Blinn材质，颜色调成绿色
- 分别导出为:

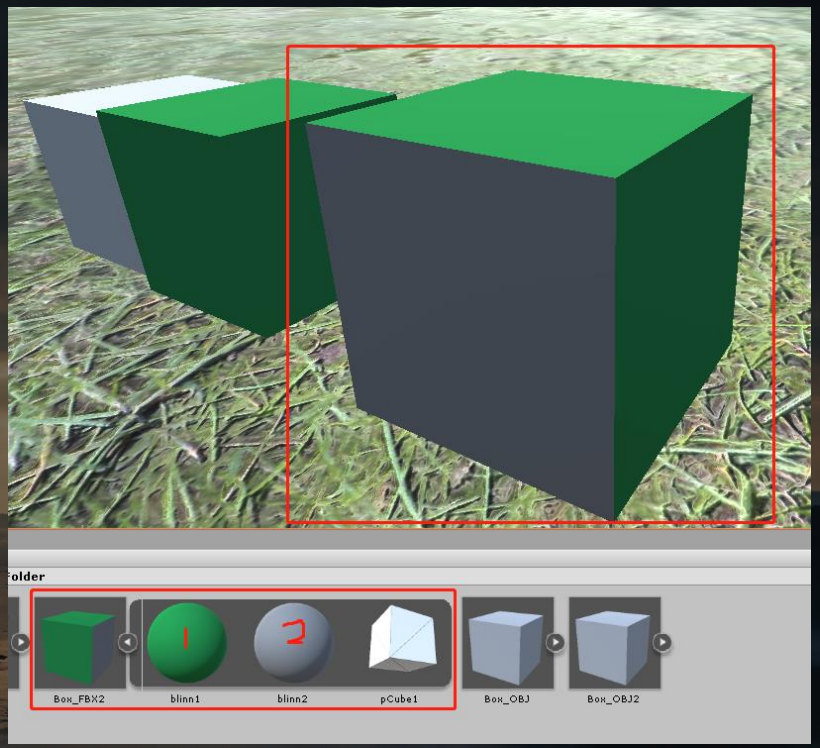
-OBJ格式文件Box_OBJ

-FBX格式文件Box_FBX

首先将OBJ文件导入Unity

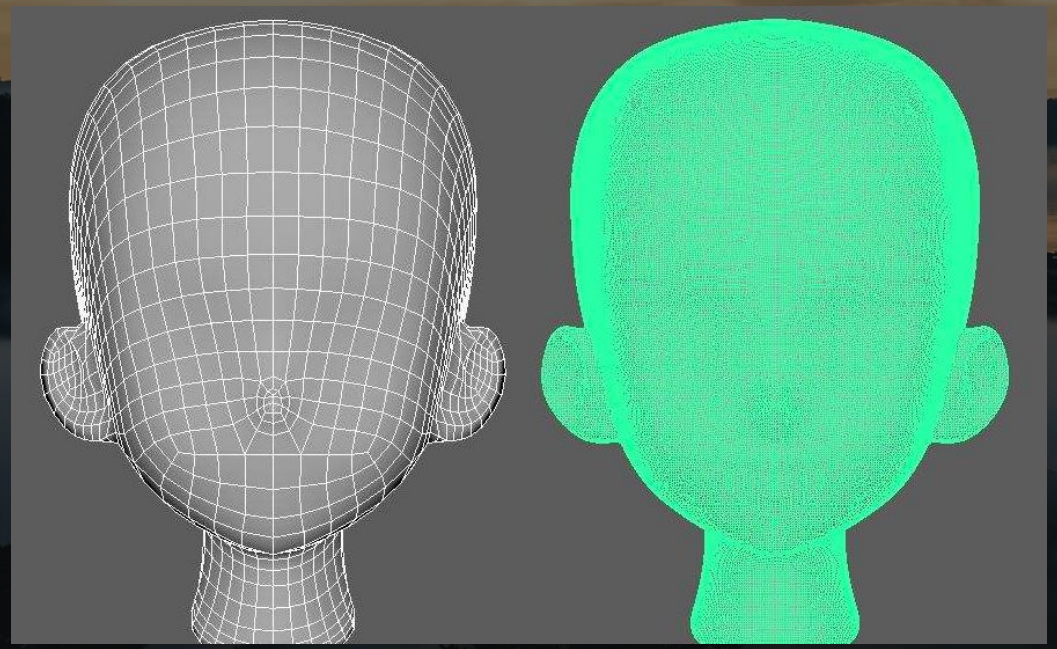
再将FBX文件导入Unity

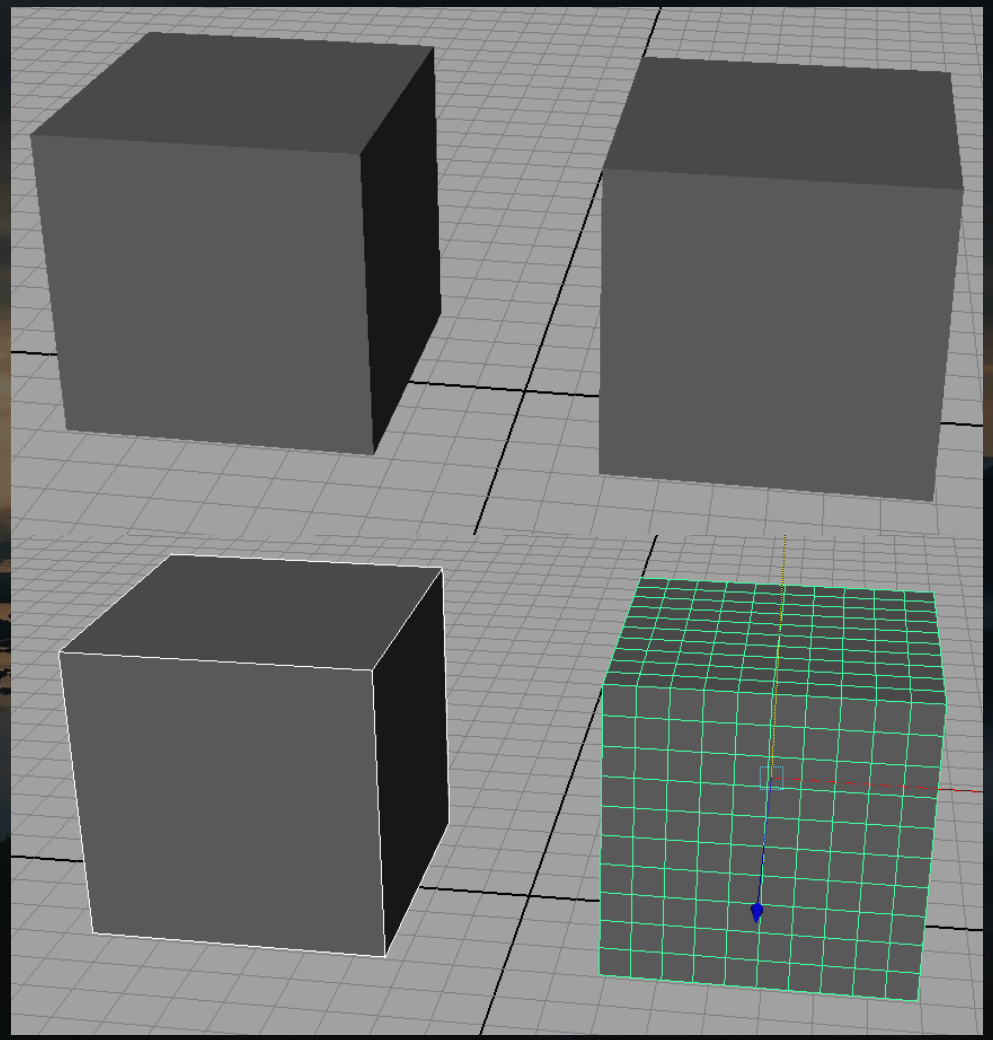
可以发现，FBX继承了MAYA中的被赋予材质球及相关属性，OBJ却没有继承多维子材质



资源优化

为何要进行资源的优化?
什么样的资源需要优化?
如何优化?

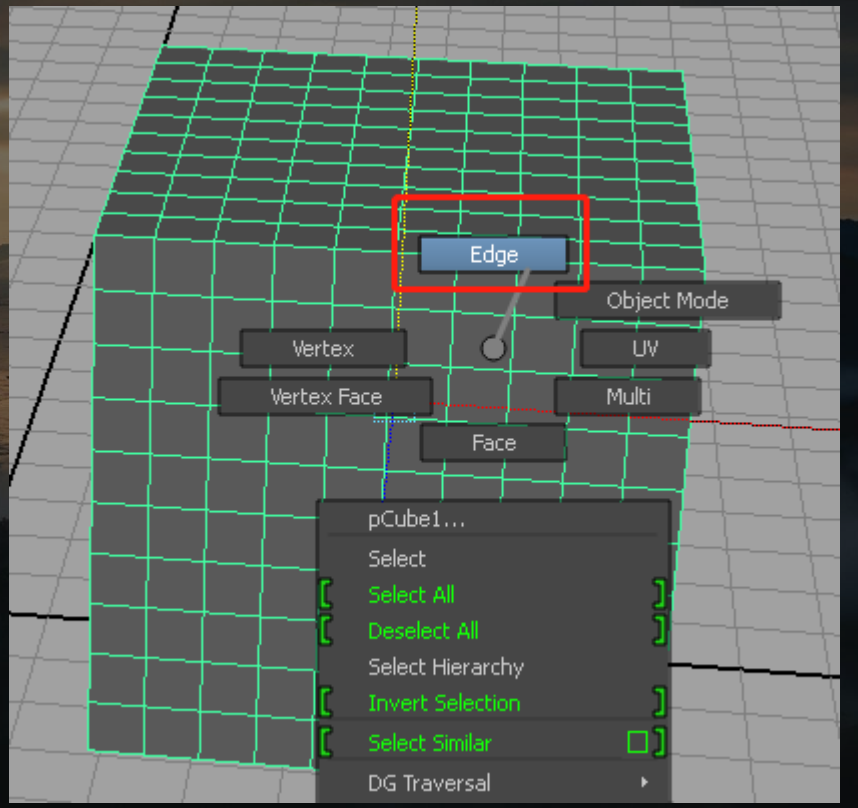




资源优化

面数优化-MAYA举例

1.鼠标右键在模型上按住-出现菜单-上滑至Edge-松鼠标

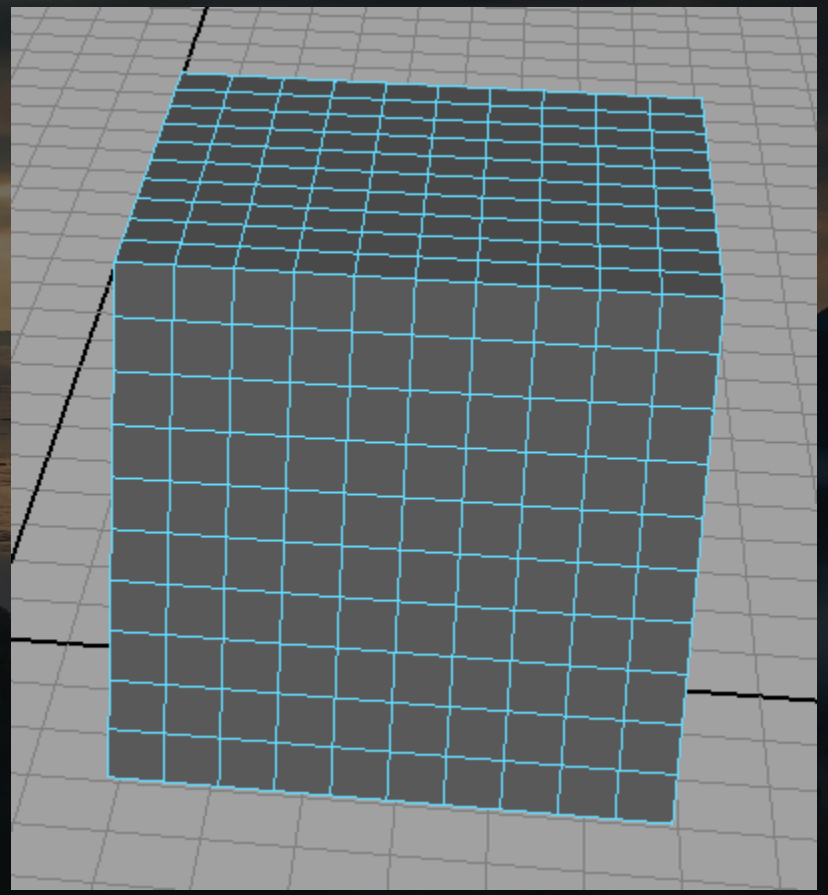


资源优化

面数优化-MAYA举例

1.鼠标右键在模型上按住-出现菜单-上滑至Edge-松鼠标

2.变为蓝色线编辑状态



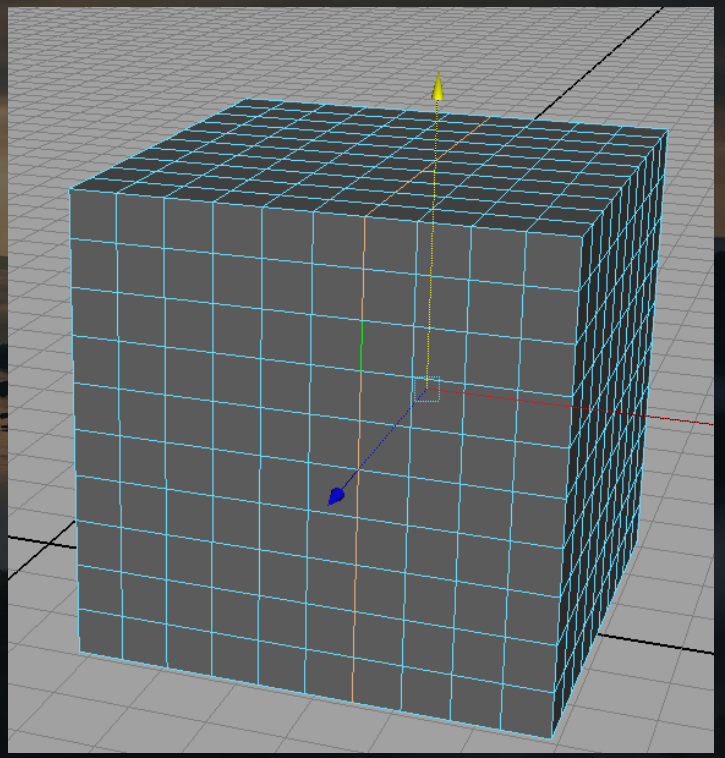
资源优化

面数优化-MAYA举例

1.鼠标右键在模型上按住-出现菜单-上滑至Edge-松鼠标

2.变为蓝色线编辑状态

3.鼠标左键选择不需要的线段



资源优化

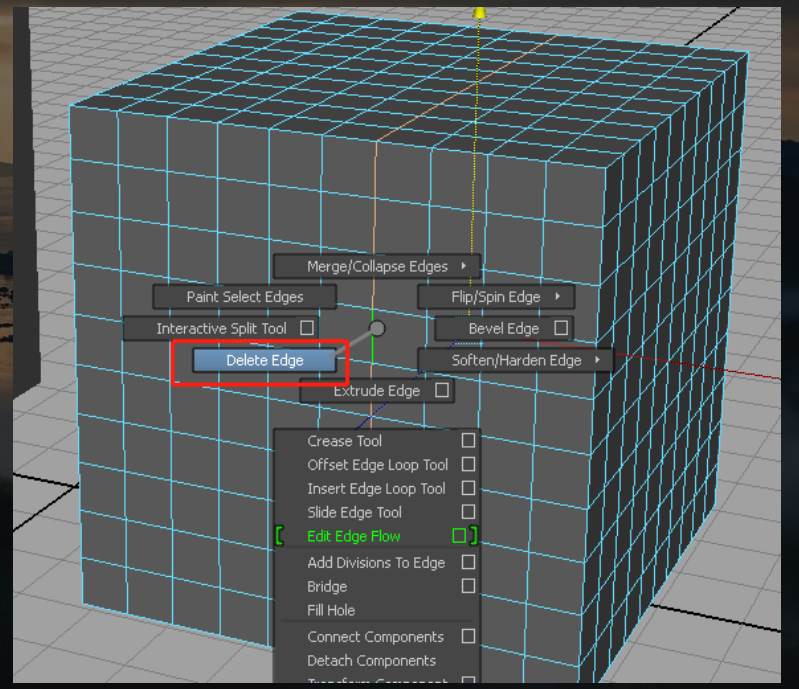
面数优化-MAYA举例

1.鼠标右键在模型上按住-出现菜单-上滑至Edge-松鼠标

2.变为蓝色线编辑状态

3.鼠标左键选择不需要的线段

4.按住Shift+鼠标右键(按住)-出现菜单-滑至删除-松开鼠标



资源优化

面数优化-MAYA举例

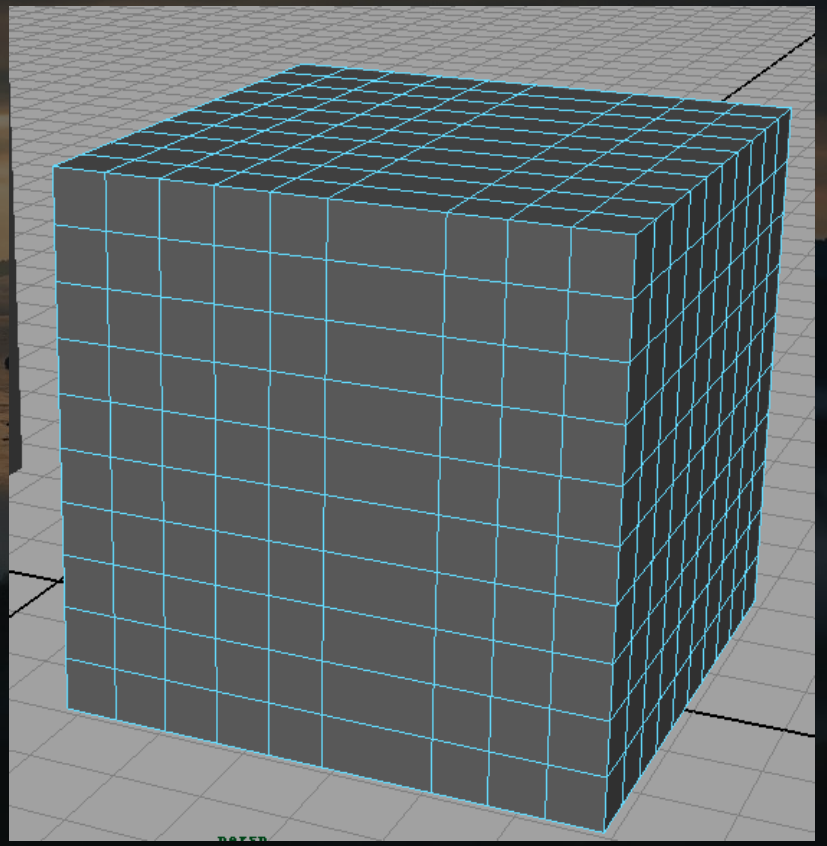
1.鼠标右键在模型上按住-出现菜单-上滑至Edge-松鼠标

2.变为蓝色线编辑状态

3.鼠标左键选择不需要的线段

4.按住Shift+鼠标右键(按住)-出现菜单-滑至删除-松开鼠标

5.线段删除完毕



资源优化

面数优化-MAYA举例

1.鼠标右键在模型上按住-出现菜单-上滑至Edge-松鼠标

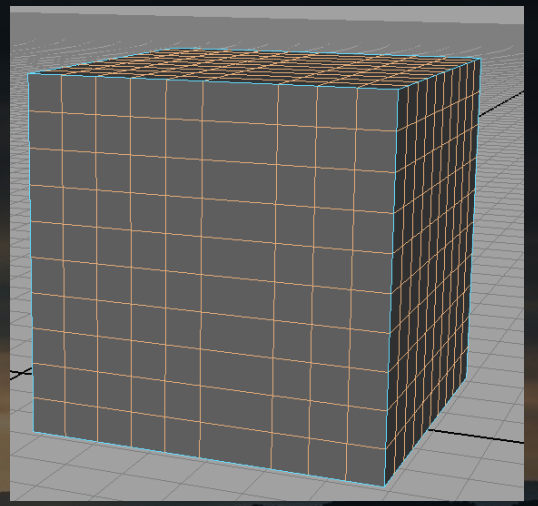
2.变为蓝色线编辑状态

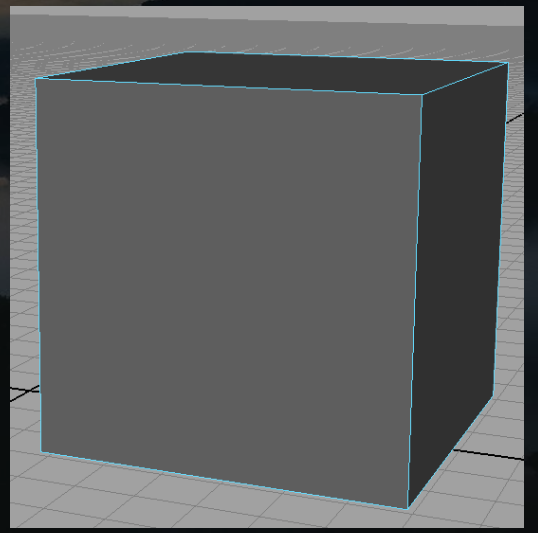
3.鼠标左键选择不需要的线段

4.按住Shift+鼠标右键(按住)-出现菜单-滑至删除-松开鼠标

5.线段删除完毕

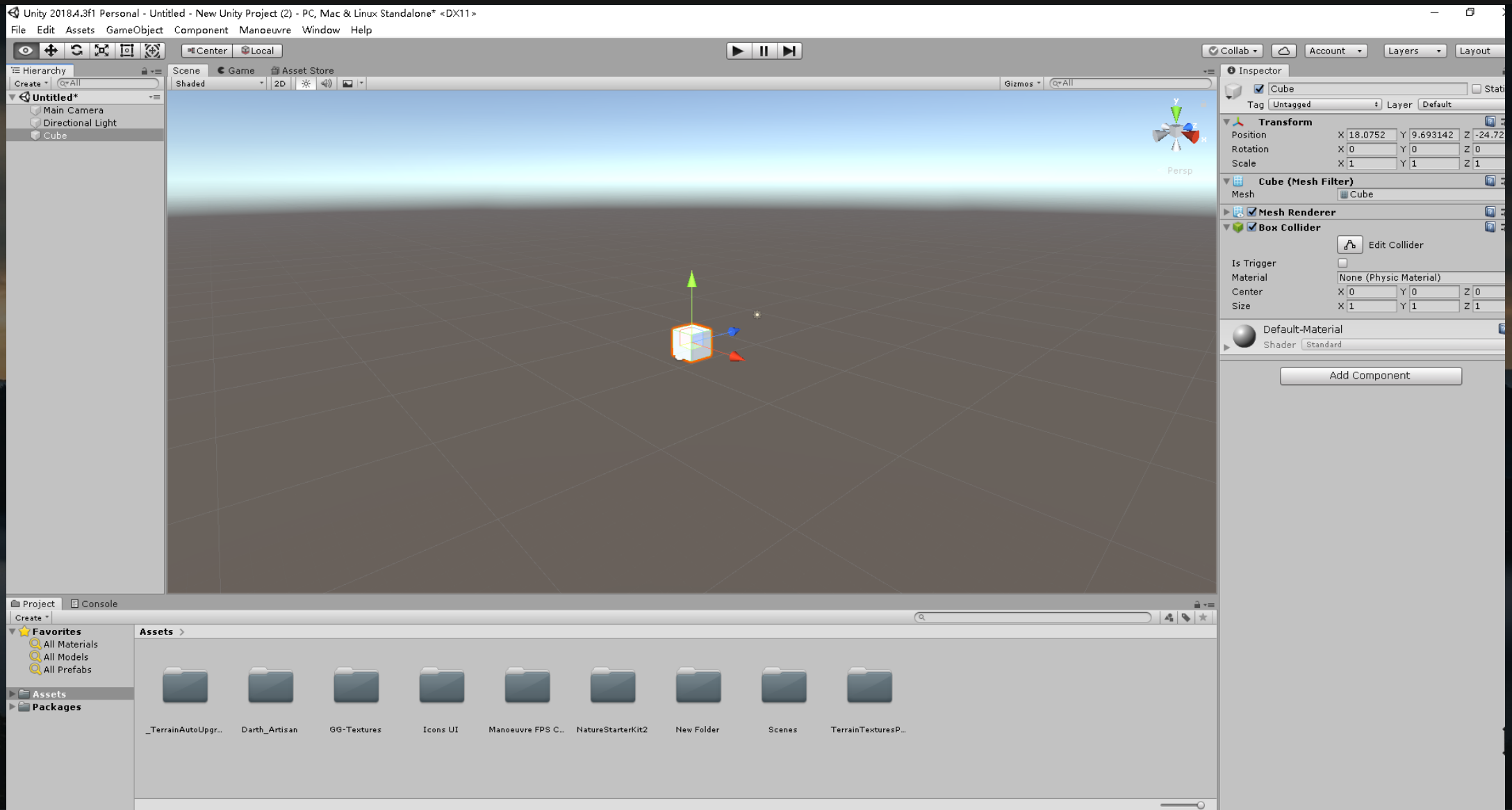
6.重复以上操作，将无用的线段删除，减低面数





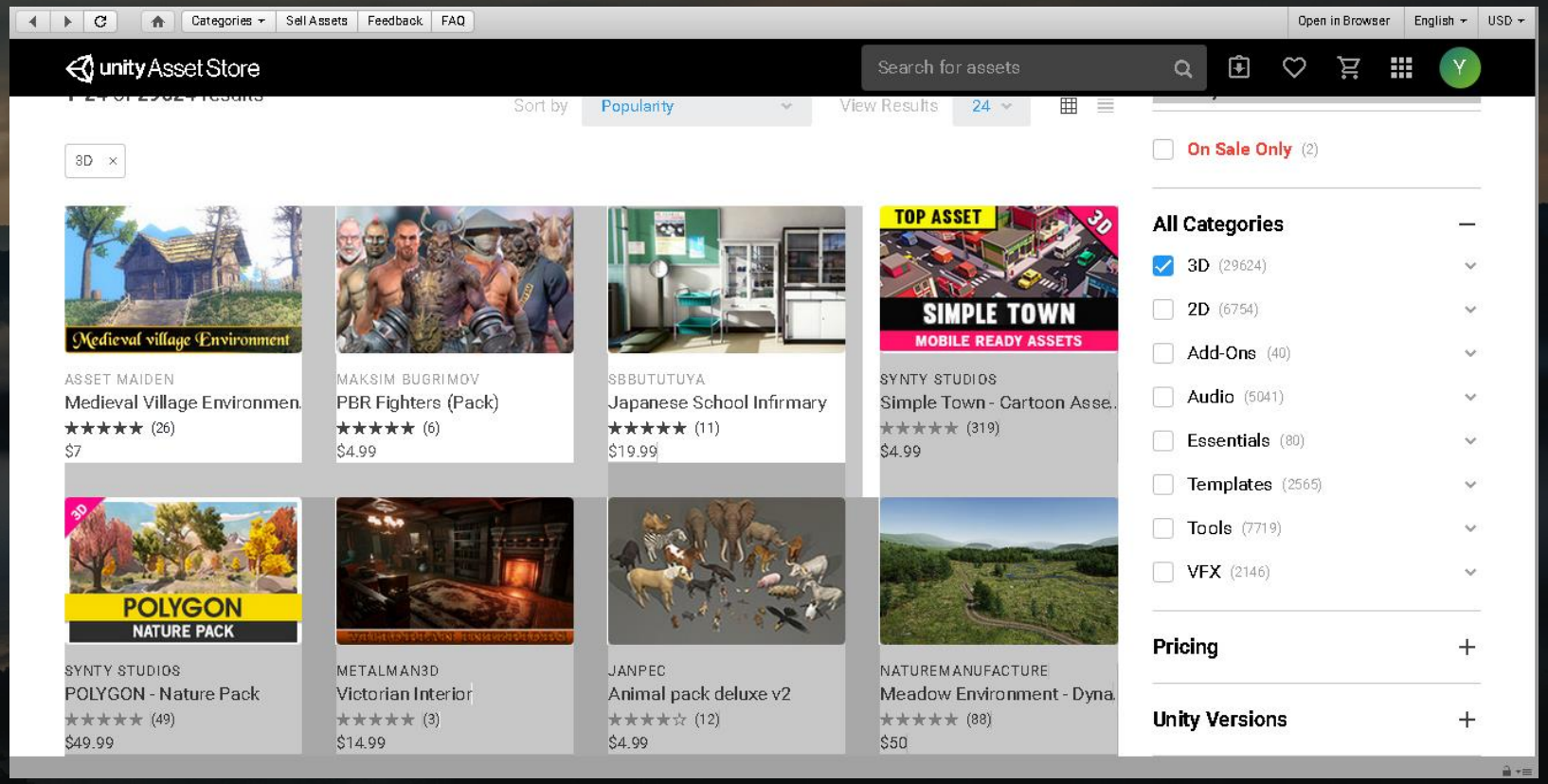
05 Unity使用概述

Unity界面简介


Unity 2018.4.3f1 Personal - Untitled - New Unity Project (2) - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Manoeuvre Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Create
Untitled*
Main Camera
Directional Light
Cube
Scene Game Asset Store
Shaded 2D Gizmos
Persp
Inspector
Cube
Tag Untagged Layer Default
Transform
Position X 18.0752 Y 9.693142 Z -24.72
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Cube (Mesh Filter)
Mesh Cube
Mesh Renderer
Box Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1
Default-Material
Shader Standard
Add Component
Project Console
Create
Favorites
All Materials
All Models
All Prefabs
Assets
Packages
Assets
_TerrainAutoUpgr...
Darth_Artisan
GG-Textures
Icons UI
Manoeuvre FPS C...
NatureStarterKit2
New Folder
Scenes
TerrainTexturesP...

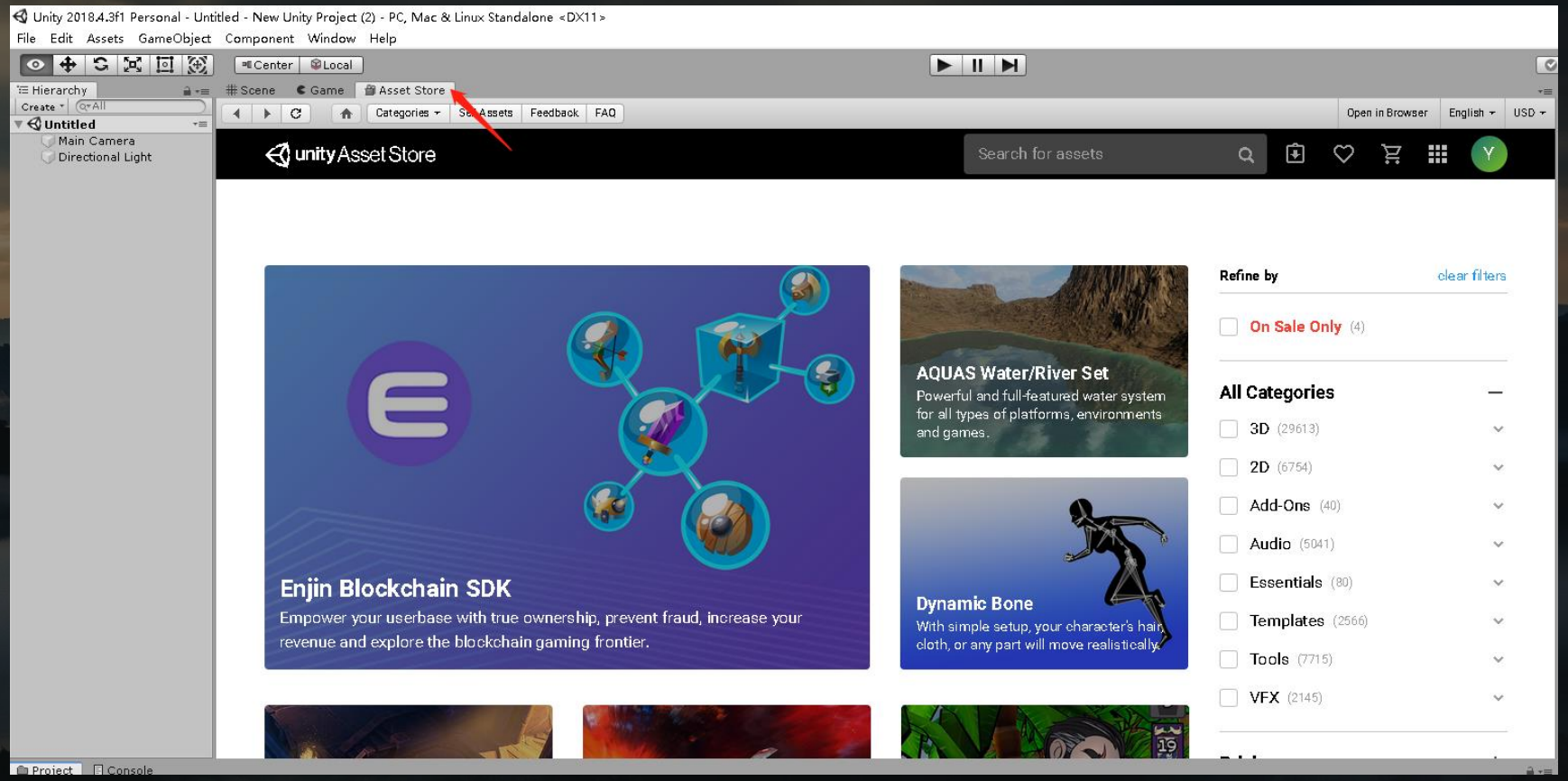
Unity Store

- 什么是Unity Store?
- 我们能用Unity Store做什么?



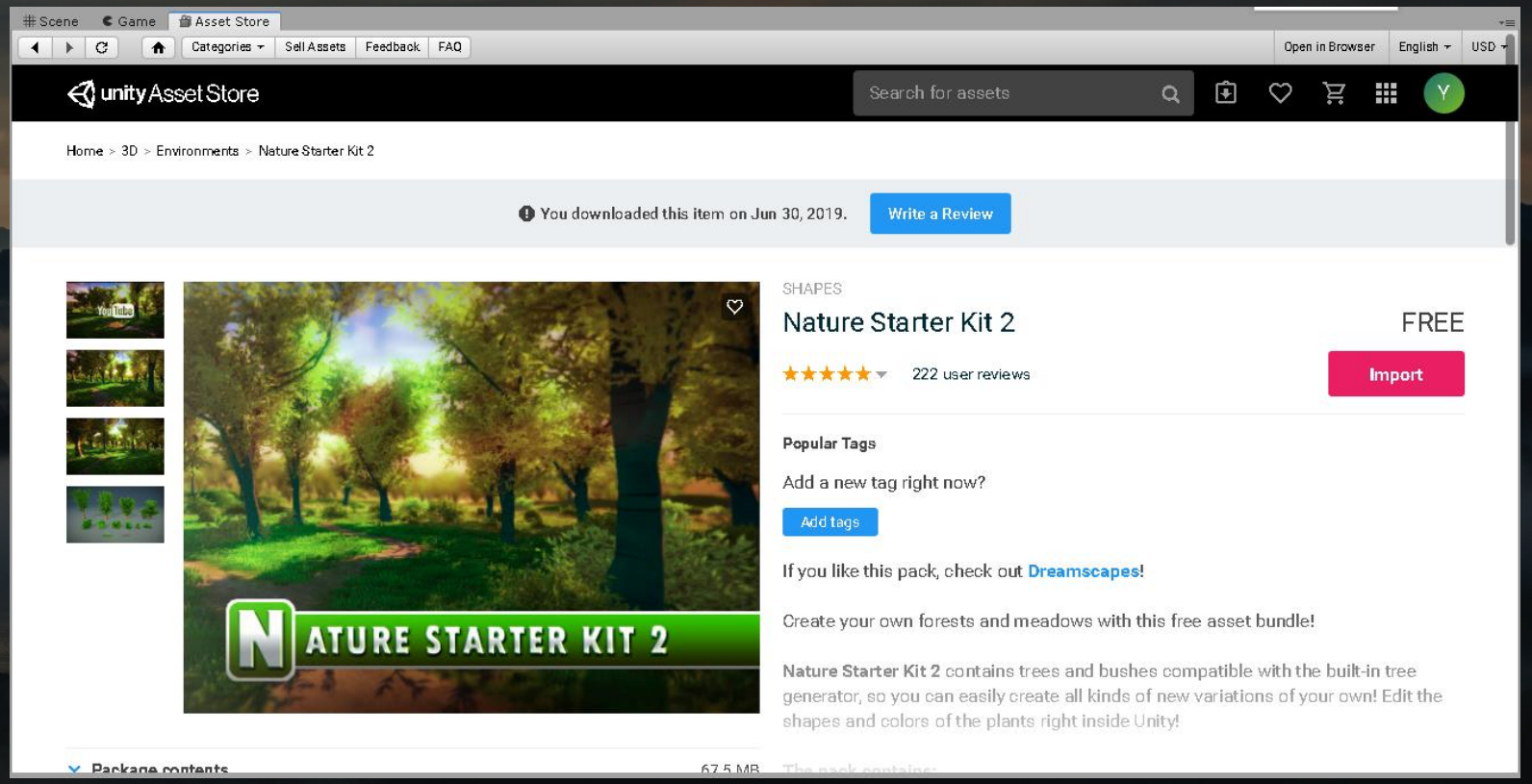
Unity Store

- 打开Asset Store



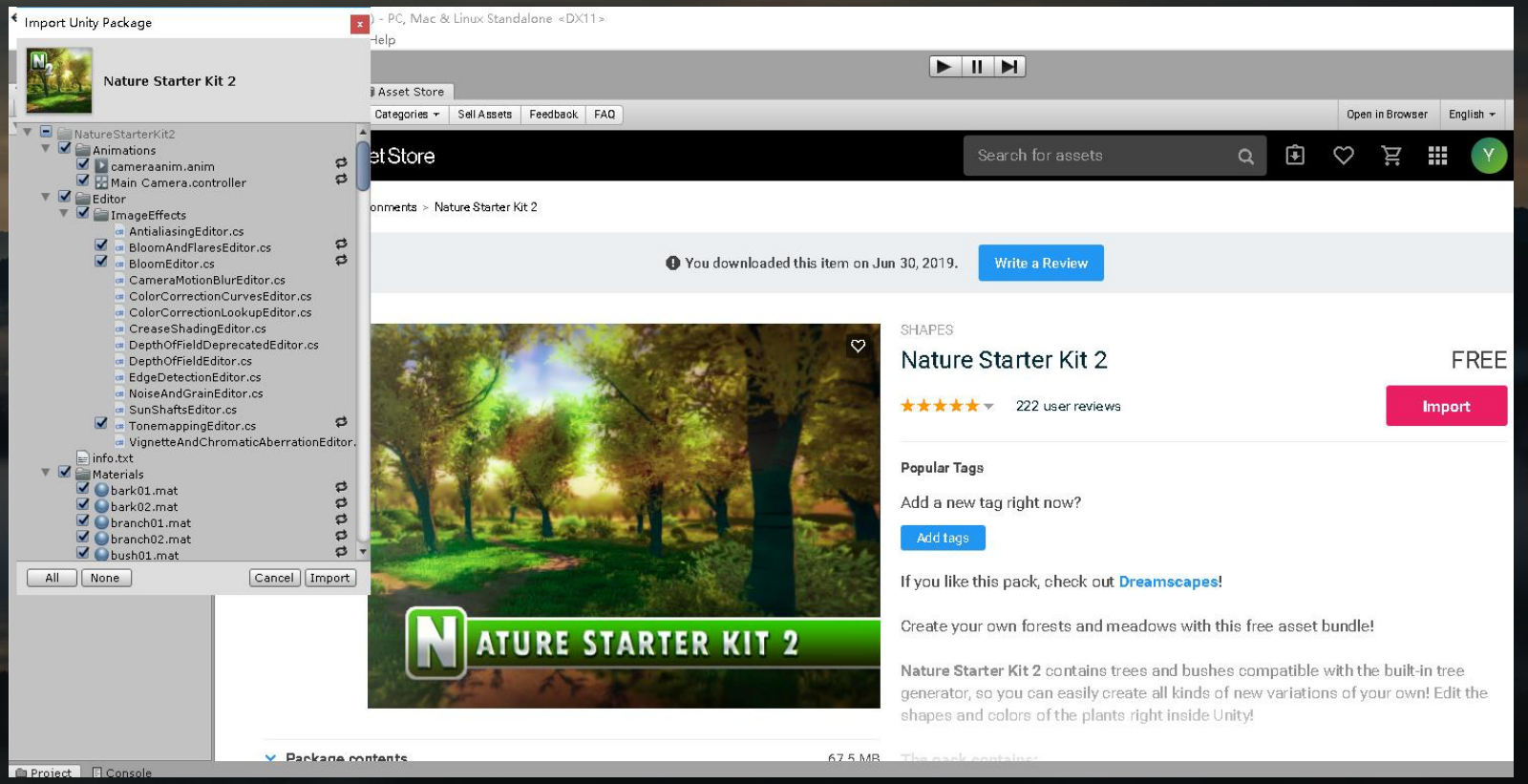
Unity Store

- 打开Asset Store
- 选择想要下载的资源-点击下载-下载完毕后点击导入



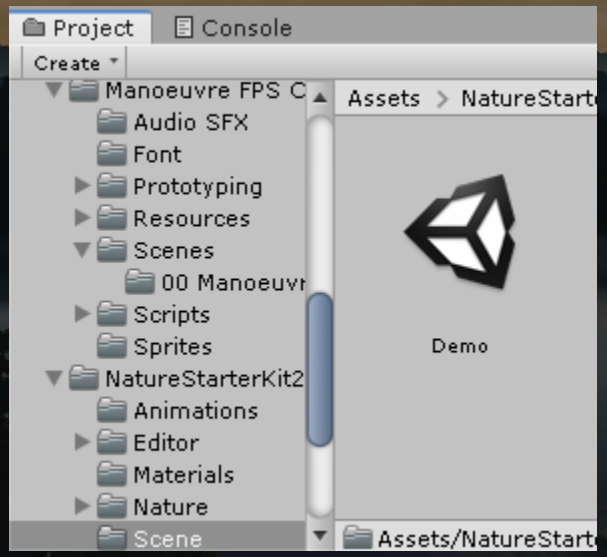
Unity Store

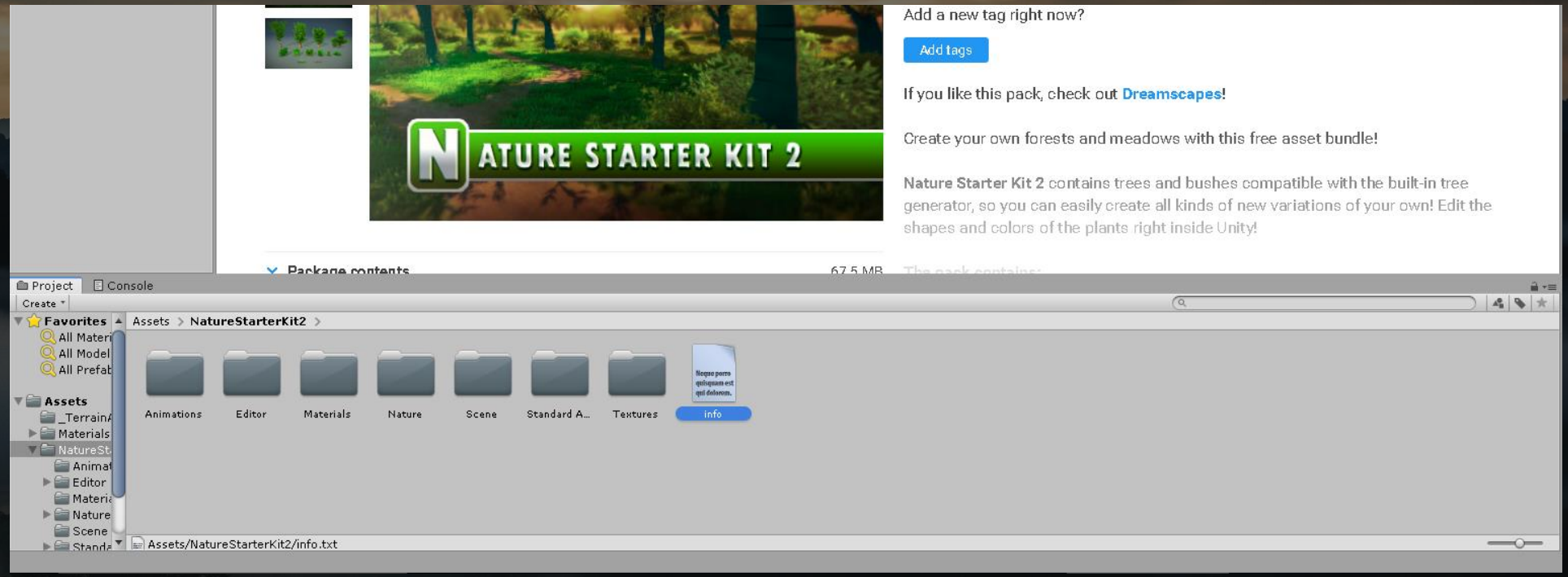
- 打开Asset Store
- 选择想要下载的资源-点击下载-下载完毕后点击导入
- 导入



Unity Store

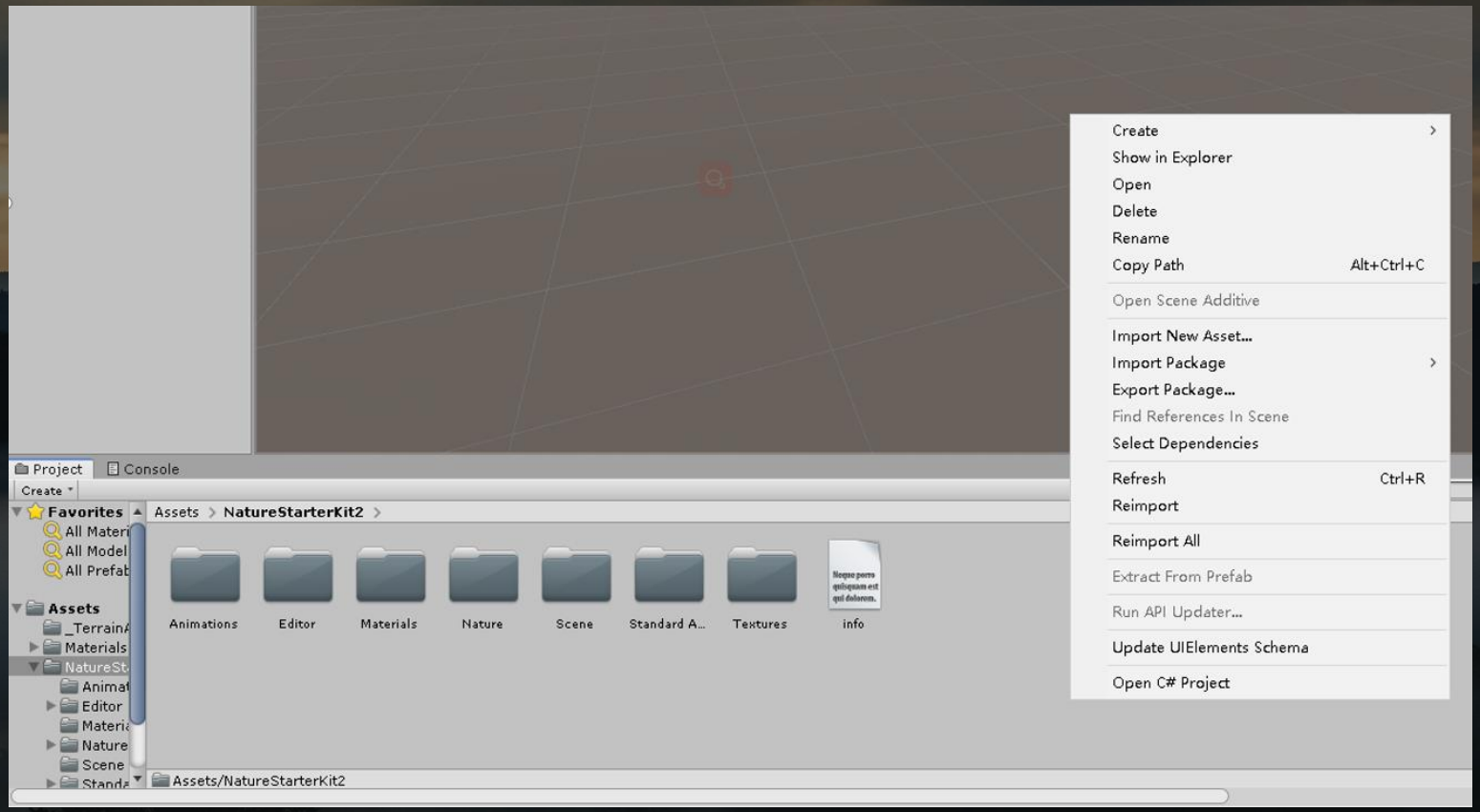
- 打开Asset Store
- 选择想要下载的资源-点击下载-下载完毕后点击导入
- 导入
- 查看文件并使用





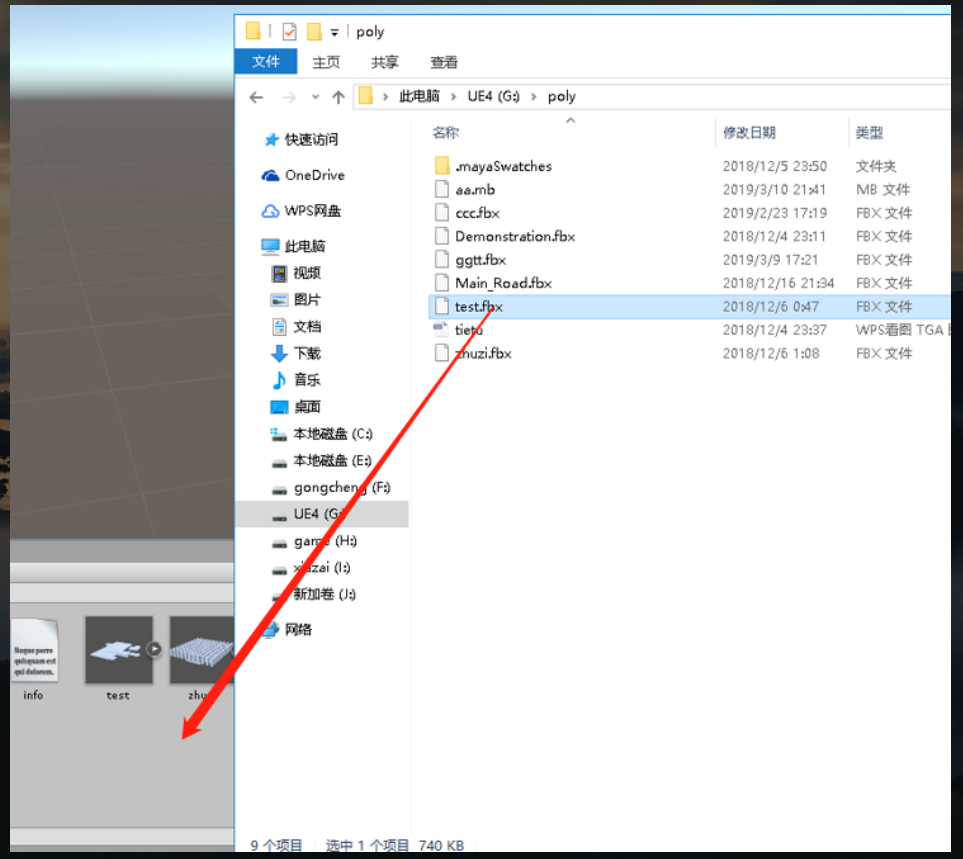
美术资源导入Unity引擎及应用

- 面板处单击鼠标右键-选择导入新的资产-窗口中选择要导入的资源-导入



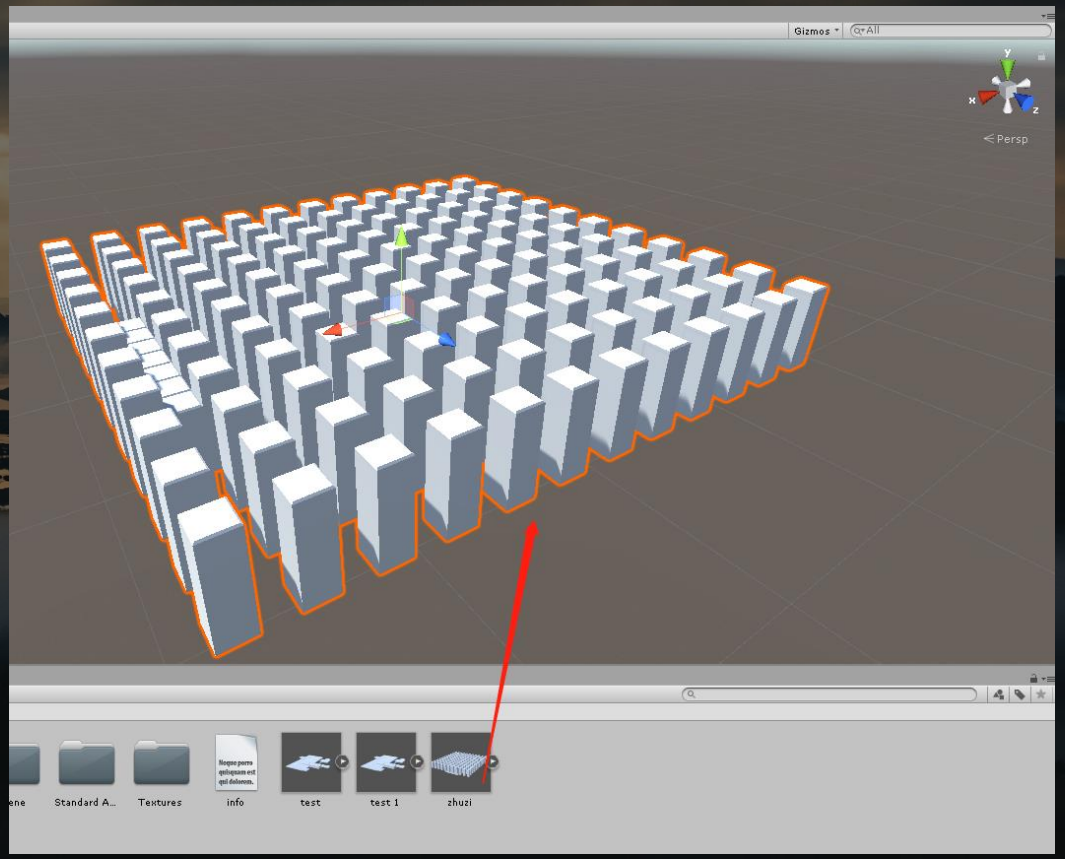
美术资源导入Unity引擎及应用

- 面板处单击鼠标右键-选择导入新的资产-窗口中选择要导入的资源-导入
- 文件夹中将需要导入的资源拖拽至面板处



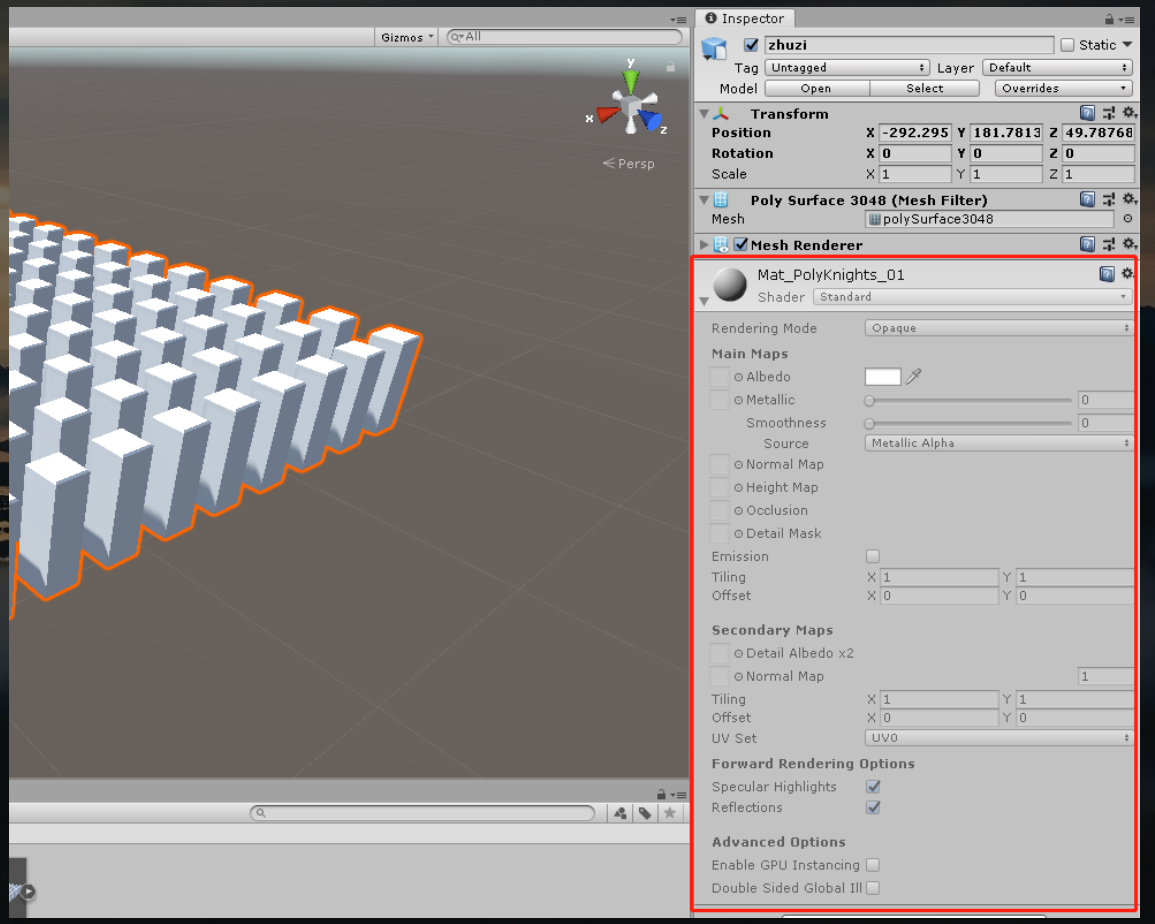
美术资源导入Unity引擎及应用

- 面板处单击鼠标右键-选择导入新的资产-窗口中选择要导入的资源-导入
- 文件夹中将需要导入的资源拖拽至面板处
- 面板中左键拖拽资源至场景中



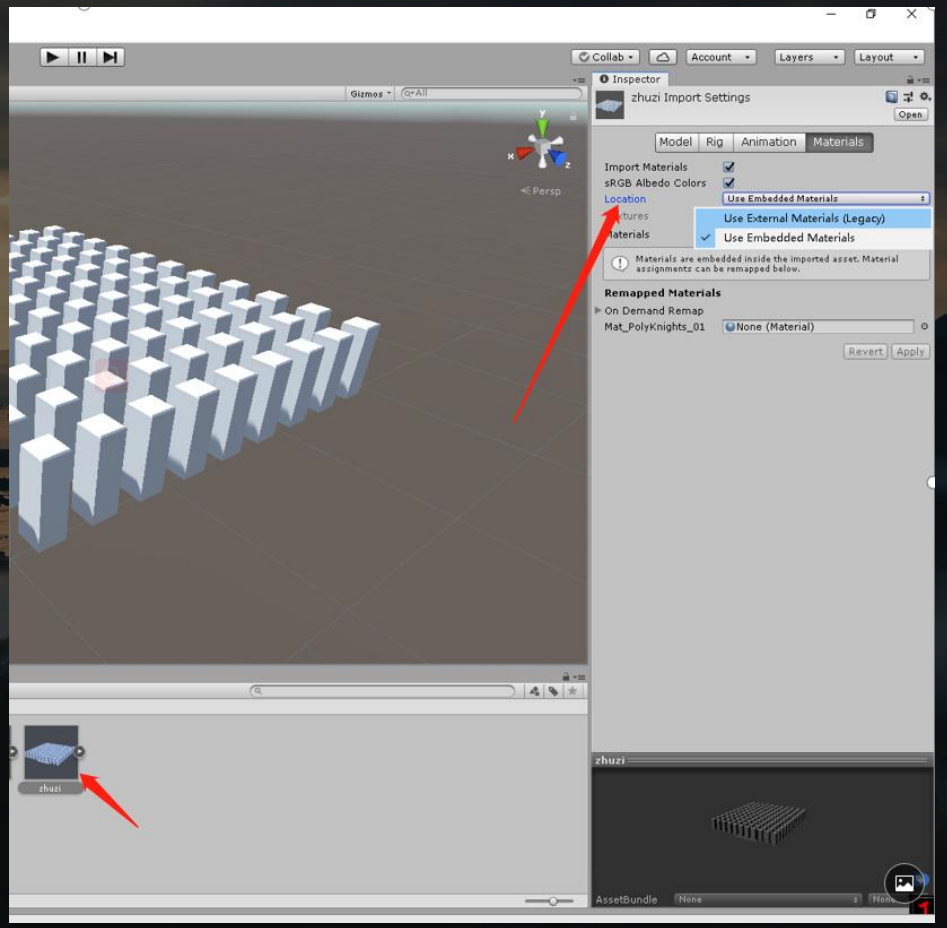
美术资源导入Unity引擎及应用

- 面板处单击鼠标右键-选择导入新的资产-窗口中选择要导入的资源-导入
- 文件夹中将需要导入的资源拖拽至面板处
- 面板中左键拖拽资源至场景中
- 右边为属性栏，红框中为材质属性



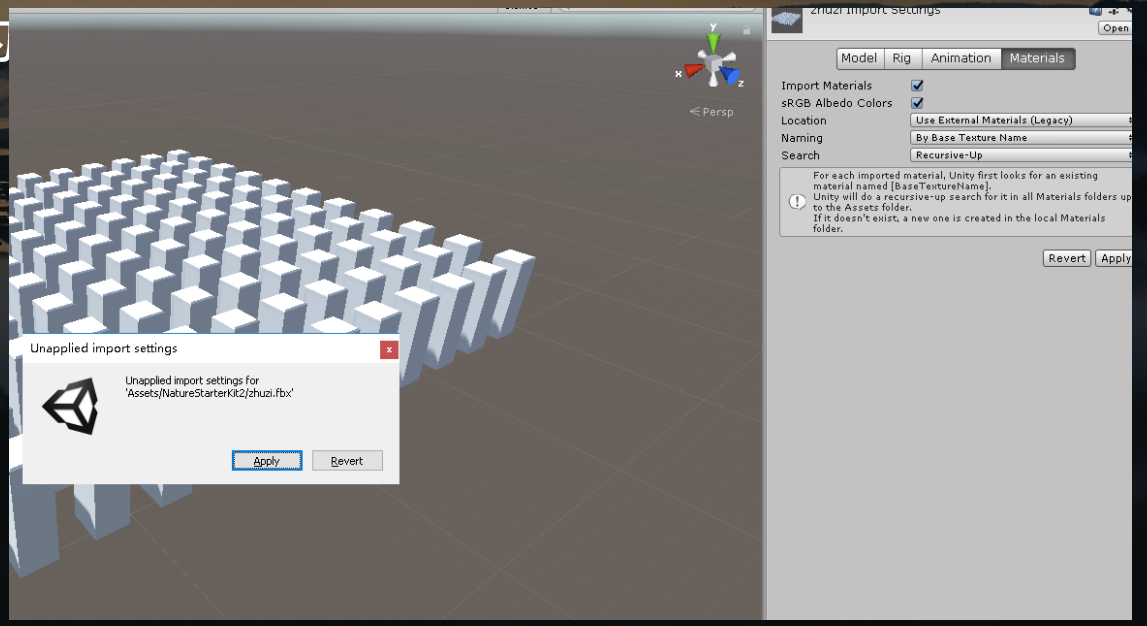
美术资源导入Unity引擎及应用

- 面板处单击鼠标右键-选择导入新的资产-窗口中选择要导入的资源-导入
- 文件夹中将需要导入的资源拖拽至面板处
- 面板中左键拖拽资源至场景中
- 右边为属性栏，红框中为材质属性
- 如材质球为无法编辑状态，参考图中选项改为1



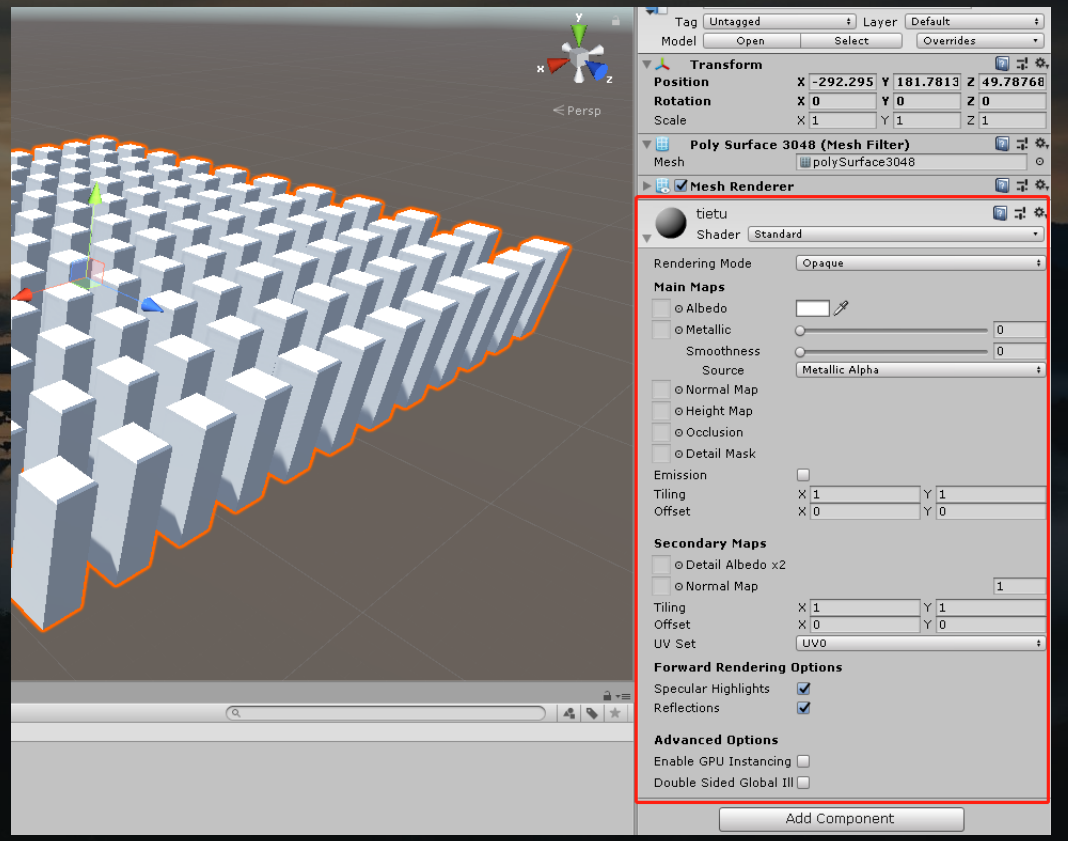
美术资源导入Unity引擎及应用

- 面板处单击鼠标右键-选择导入新的资产-窗口中选择要导入的资源-导入
- 文件夹中将需要导入的资源拖拽至面板处
- 面板中左键拖拽资源至场景中
- 右边为属性栏，红框中为材质属性
- 如材质球为无法编辑状态，参考图中选项改为
- Apply



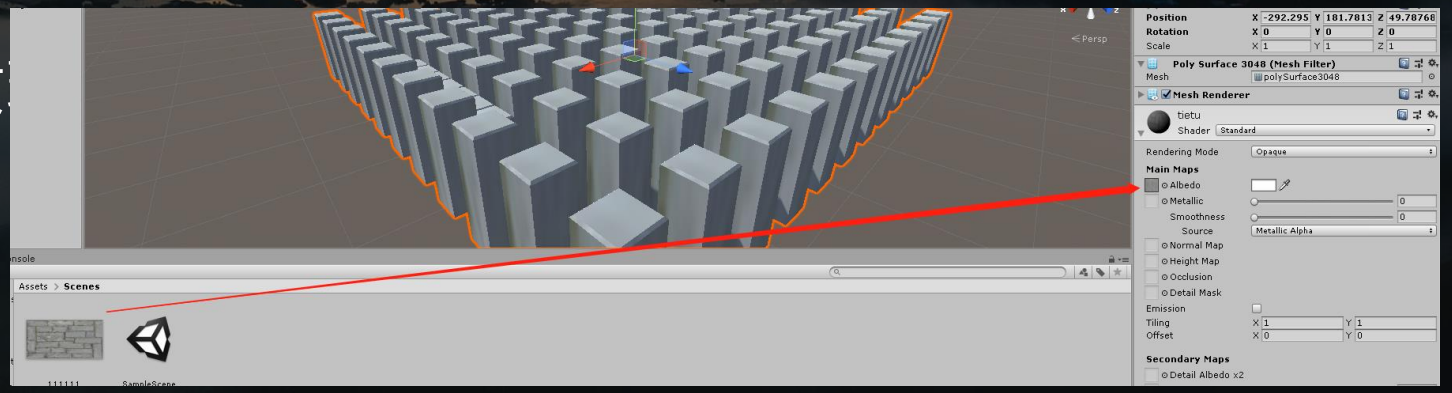
美术资源导入Unity引擎及应用

- 面板处单击鼠标右键-选择导入新的资产-窗口中选择要导入的资源-导入
- 文件夹中将需要导入的资源拖拽至面板处
- 面板中左键拖拽资源至场景中
- 右边为属性栏，红框中为材质属性
- 如材质球为无法编辑状态，参考图中选项改为1
- Apply
- 已变为可编辑状态材质球



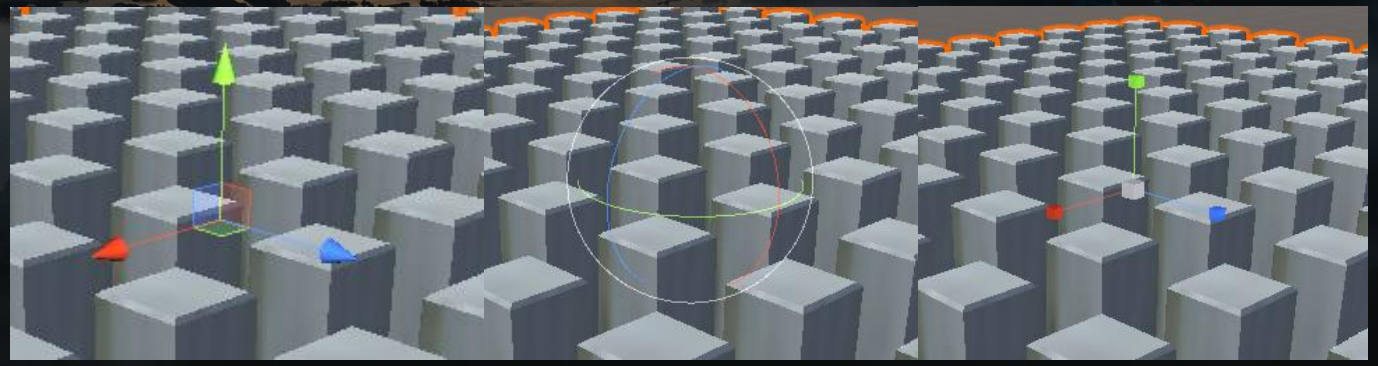
美术资源导入Unity引擎及应用

- 面板处单击鼠标右键-选择导入新的资产-窗口中选择要导入的资源-导入
- 文件夹中将需要导入的资源拖拽至面板处
- 面板中左键拖拽资源至场景中
- 右边为属性栏，红框中为材质属性
- 如材质球为无法编辑状态，参考图中选项改为1
- Apply
- 已变为可编辑状态材质球
- 拖拽贴图资源至颜色层，赋予材质



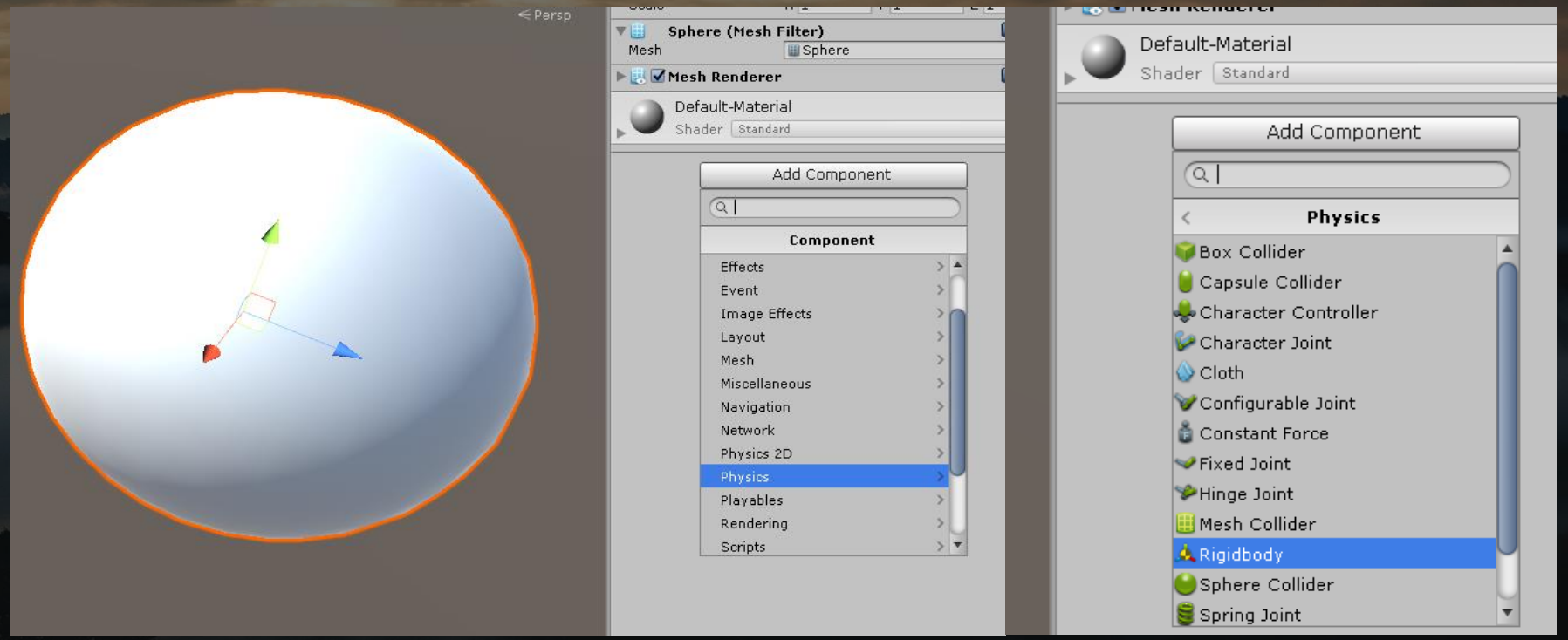
美术资源导入Unity引擎及应用

- 面板处单击鼠标右键-选择导入新的资产-窗口中选择要导入的资源-导入
- 文件夹中将需要导入的资源拖拽至面板处
- 面板中左键拖拽资源至场景中
- 右边为属性栏，红框中为材质属性
- 如材质球为无法编辑状态，参考图中选项改为1
- Apply
- 已变为可编辑状态材质球
- 拖拽贴图资源至颜色层，赋予材质球贴图，其他贴图同理
- W移动 E旋转 R缩放
- 鼠标右键按住+WASD点击=前后左右
- 鼠标右键按住移动=视角旋转
- 鼠标滚轮=视角前推/后拉



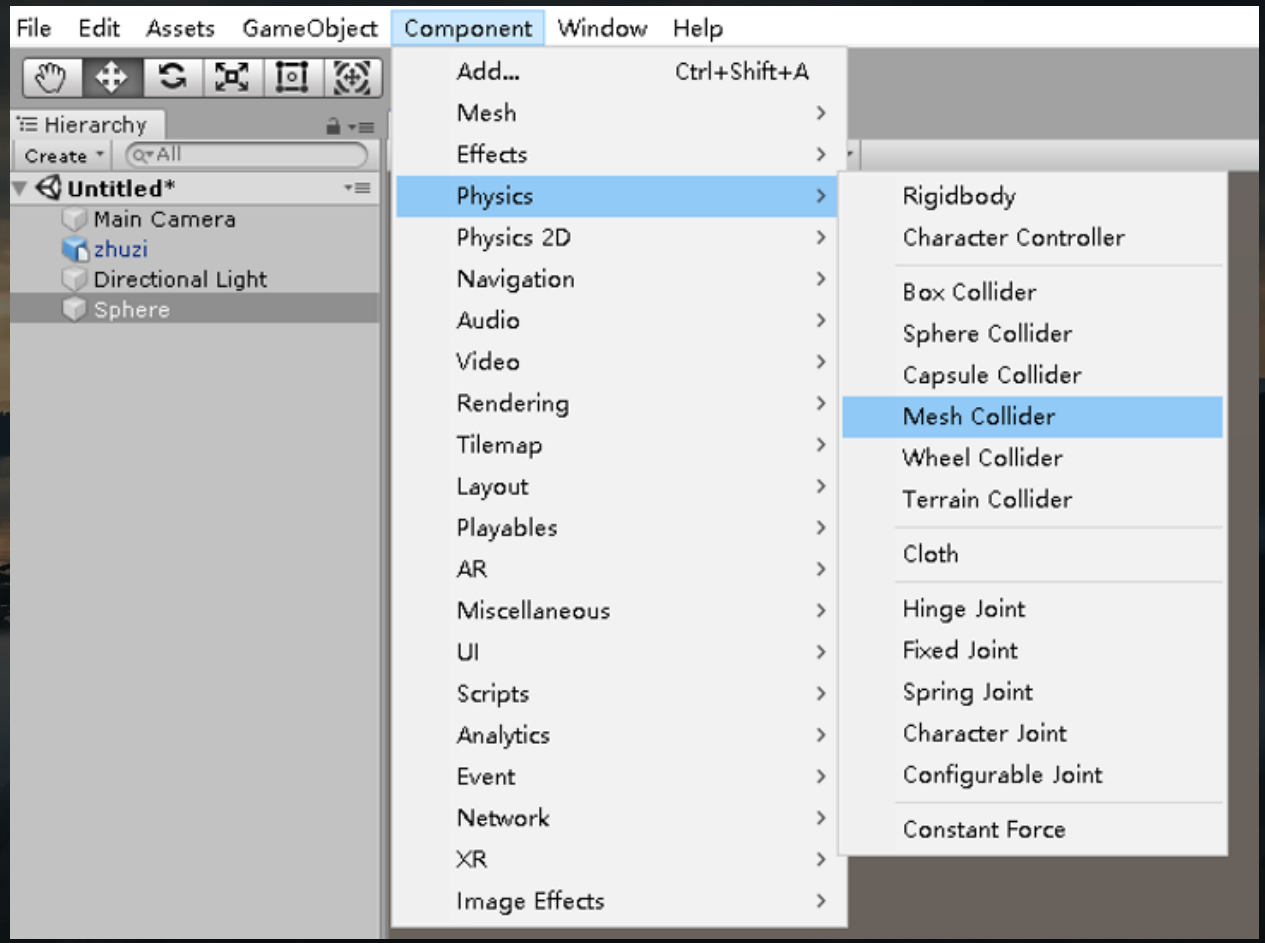
碰撞

- 什么是碰撞？
- Unity如何添加碰撞？
- 添加组件-物理-刚体组件Rigidbody



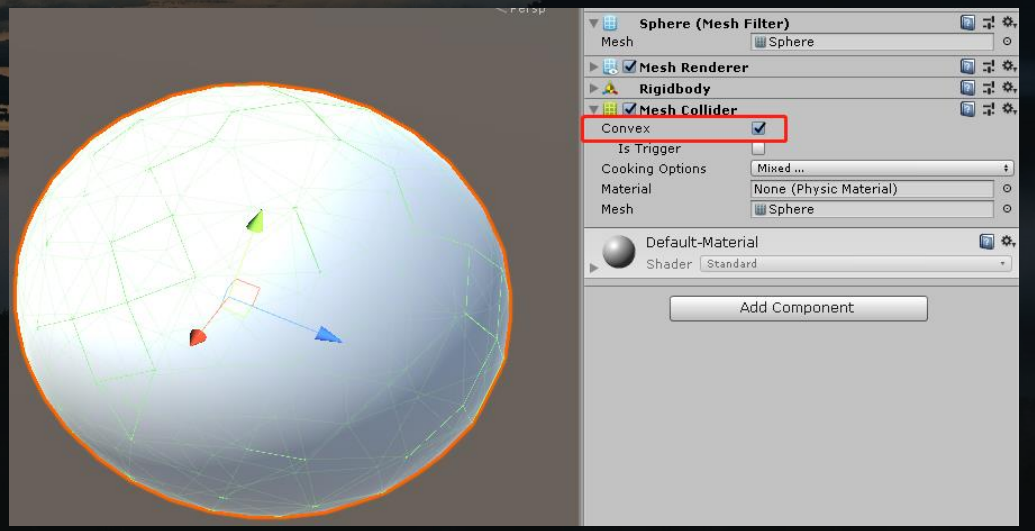
碰撞

- 什么是碰撞？
- Unity如何添加碰撞？
- 添加组件-物理-刚体组件Rigidbody
- 添加一个Mesh Collider



碰撞

- 什么是碰撞？
- Unity如何添加碰撞？
- 添加组件-物理-刚体组件Rigidbody
- 添加一个Mesh Collider
- 勾选Convex，会默认根据自己的模型生成一个碰撞范围



碰撞

- 什么是碰撞？
- Unity如何添加碰撞？
- 添加组件-物理-刚体组件Rigidbody
- 添加一个Mesh Collider
- 勾选Convex，会默认根据自己的模型生成一个碰撞范围
- Unity中参与碰撞的物体分2大块：1.发起碰撞的物体2.接收碰撞的物体

碰撞

- 什么是碰撞？
- Unity如何添加碰撞？
- 添加组件-物理-刚体组件Rigidbody
- 添加一个Mesh Collider
- 勾选Convex，会默认根据自己的模型生成一个碰撞范围
- Unity中参与碰撞的物体分2大块：1.发起碰撞的物体2.接收碰撞的物体

1. 发起碰撞物体有：Rigodbody , CharacterController
2. 接收碰撞物体由：所有的Collider

碰撞

- 什么是碰撞？
- Unity如何添加碰撞？
- 添加组件-物理-刚体组件Rigidbody
- 添加一个Mesh Collider
- 勾选Convex，会默认根据自己的模型生成一个碰撞范围
- Unity中参与碰撞的物体分2大块：1.发起碰撞的物体2.接收碰撞的物体

1. 发起碰撞物体有：Rigidbody , CharacterController
2. 接收碰撞物体由：所有的Collider

工作的原理为：发生碰撞的物体中必须要有“发起碰撞”的物体。否则，碰撞不响应

碰撞

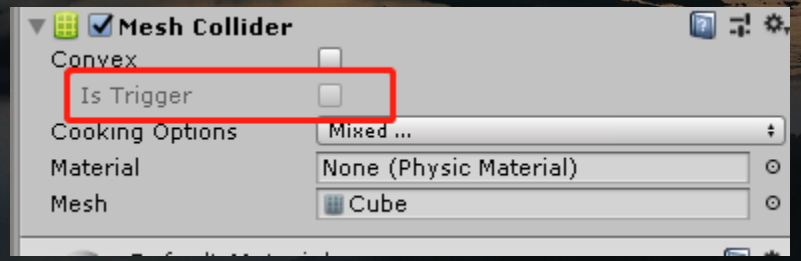
- Is Trigger： boolean型参数

当发生碰撞反应的时候，会先检查此属性

当激活此选项时，会调用碰撞双方的脚本 OnTrigger，反之，脚本方面没有任何反应

当激活此选项时，不会发生后续物理的反应。反之，发生后续的物理反应

总结：Is Trigger 好比是一个物理功能的开关，是要“物理功能”还是要“OnTrigger脚本”



碰撞

•碰撞练习：新建一个场景，添加一个Plane作为地面，调整合适摄像机角度观察

1.拖拽一个导入的模型资源至场景中，作为测试A模型

2.Ctrl+C,Ctrl+V复制A模型，移至旁边作为B模型，为B模型添加Rigidbody组件

3.复制B模型移至旁边作为C模型，为C模型添加Mesh Collider并且勾选Convex

4.单击Play按钮，观察A,B,C三个模型碰撞地面的不同效果

UI

UGUI:

•画布Canvas

-Canvas就类似我们的作画的画布，而在Canvas上的控件，则类似画布上的图画，画布是画的载体，同时我们也可以Canvas也是控件的载体

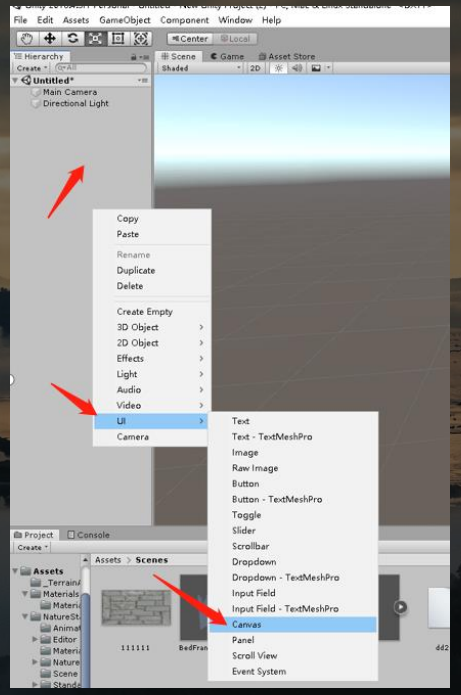
-所有的UI都应该放在Canvas里面(子层)。Canvas是一个带有Canvas组件的Game Object。

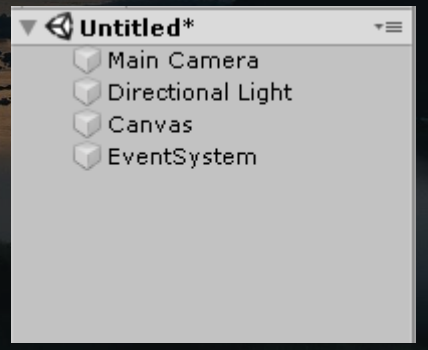
UI

•添加画布:

层级面板右键-UI-Canvas

工具栏GameObject-UI-Canvas





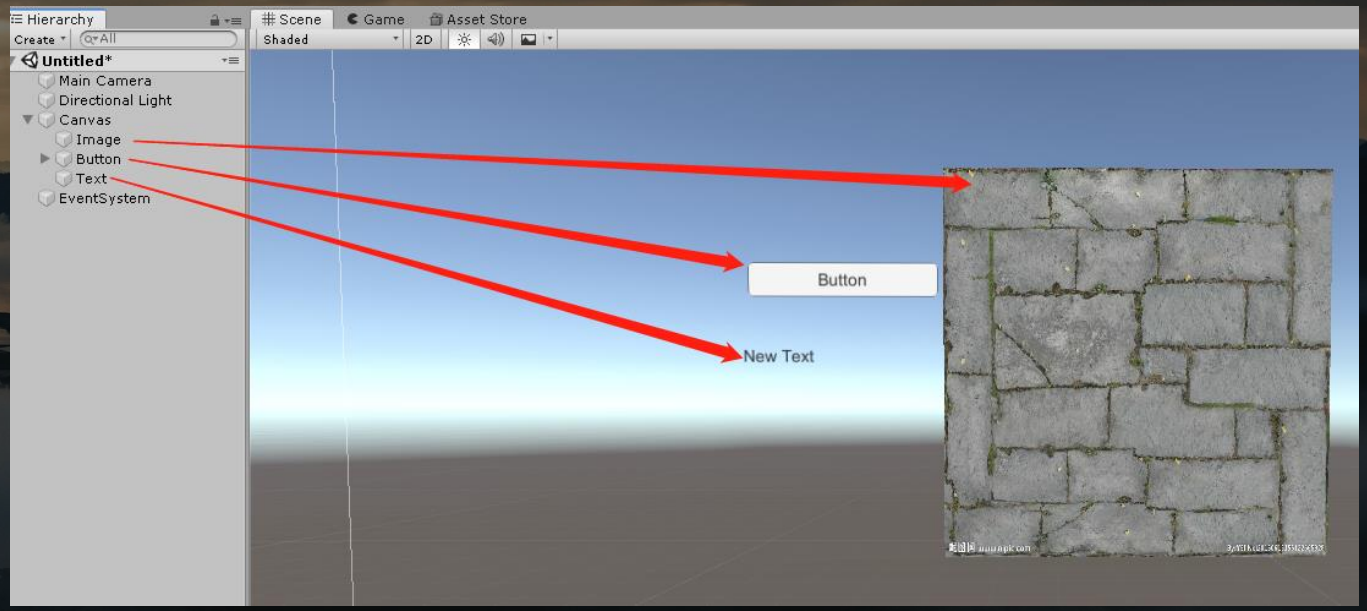
UI

•添加画布:

层级面板右键-UI-Canvas

工具栏GameObject-UI-Canvas

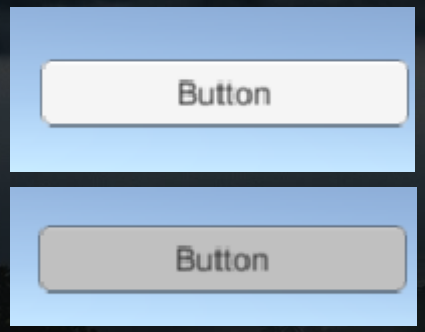
添加相应UI控件，图片/按钮/文本

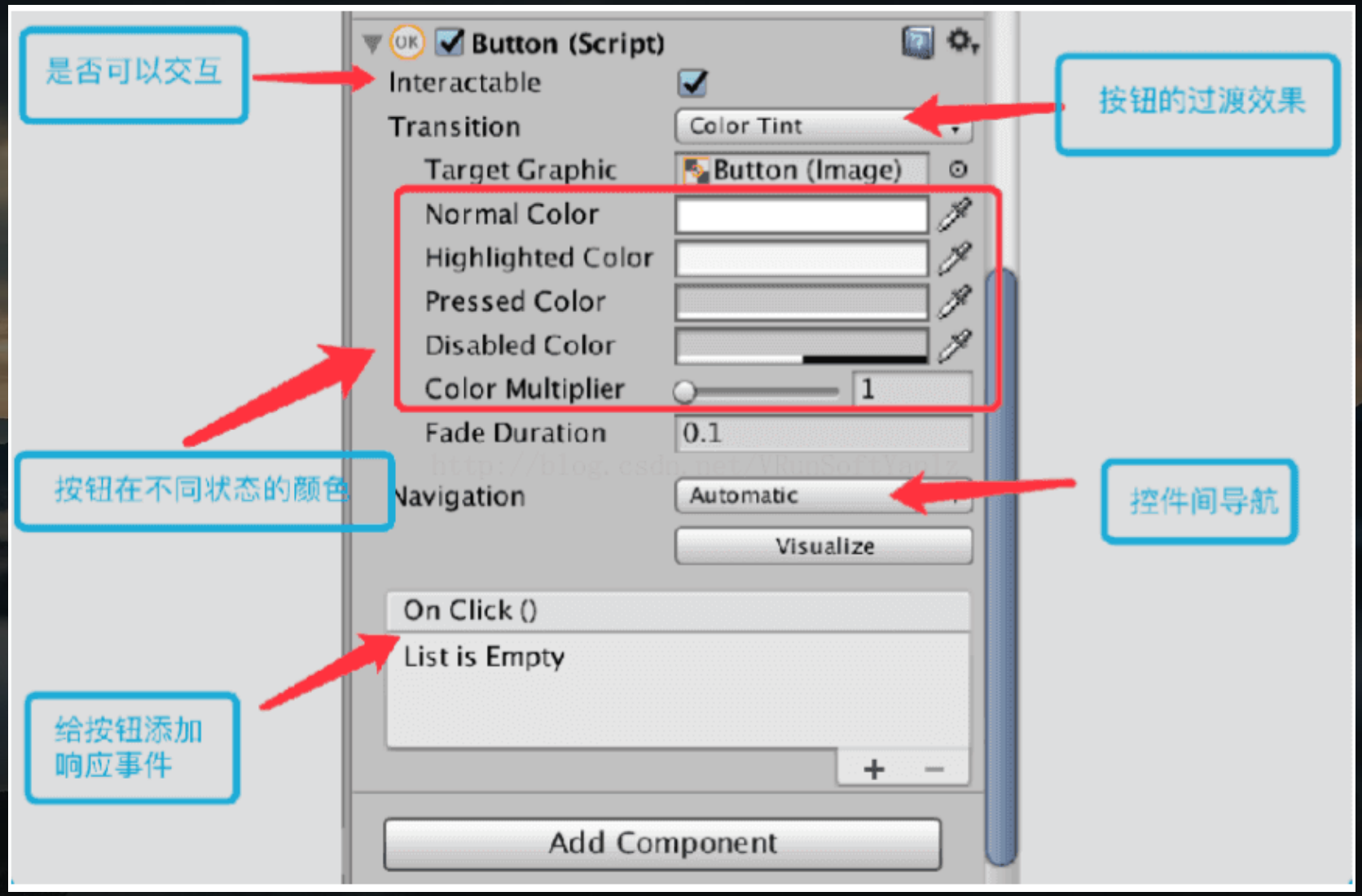


UI

UGUI:

•以Button举例

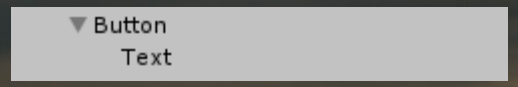




UI

UGUI:

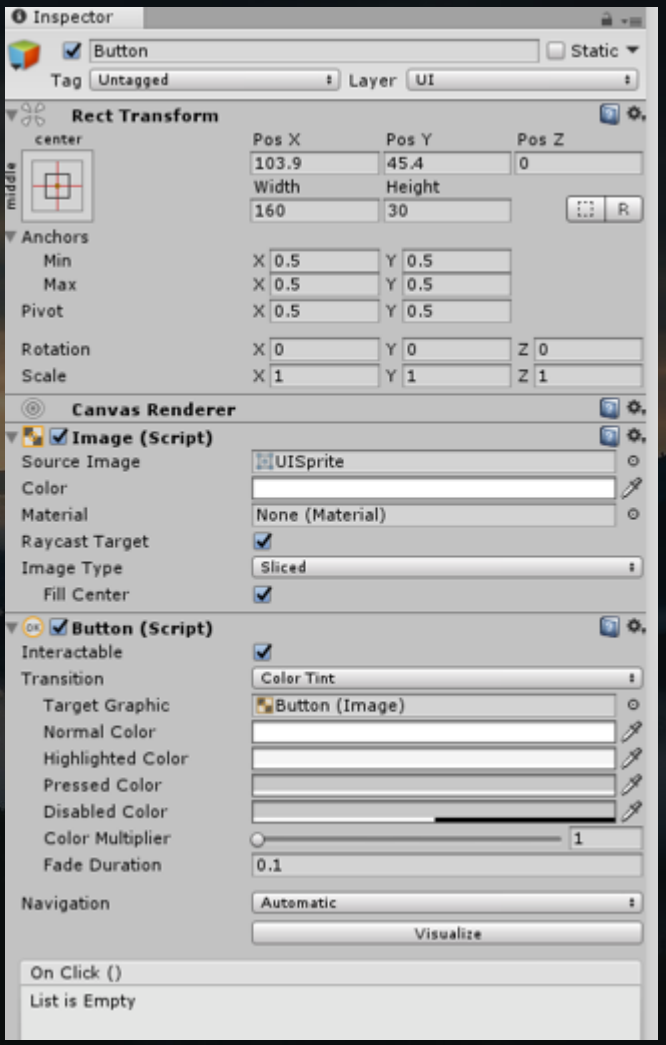
- 以Button举例
- Button组成



UI

UGUI:

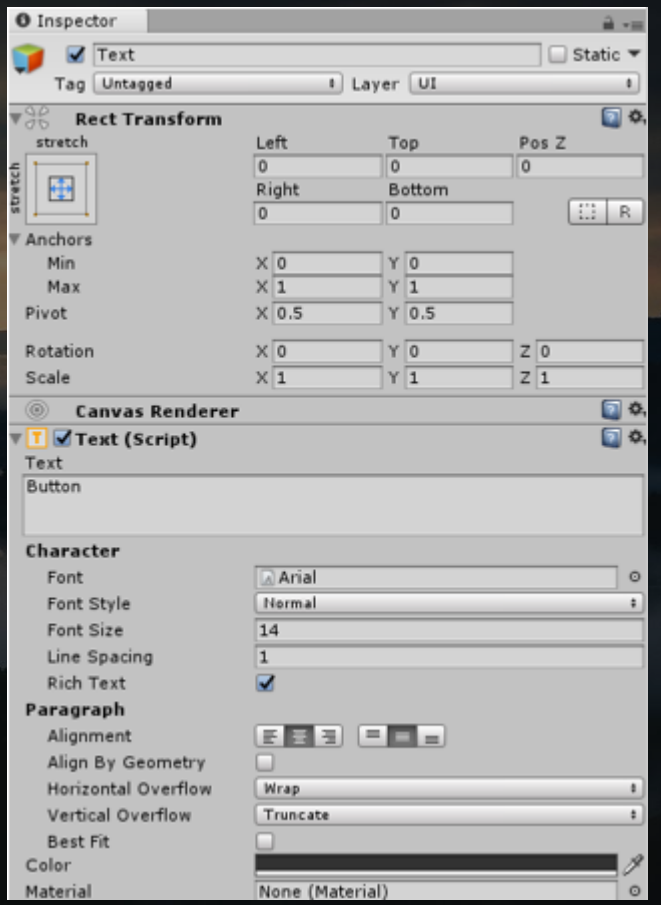
- 以Button举例
- Button组成
- Button控件： Rect Transform， Canvas Renderer， Image和Button



UI

UGUI:

- 以Button举例
- Button组成
- Button控件： Rect Transform， Canvas Renderer， Image和Button
- Text控件： Rect Transform， Canvas Renderer， Text



UI

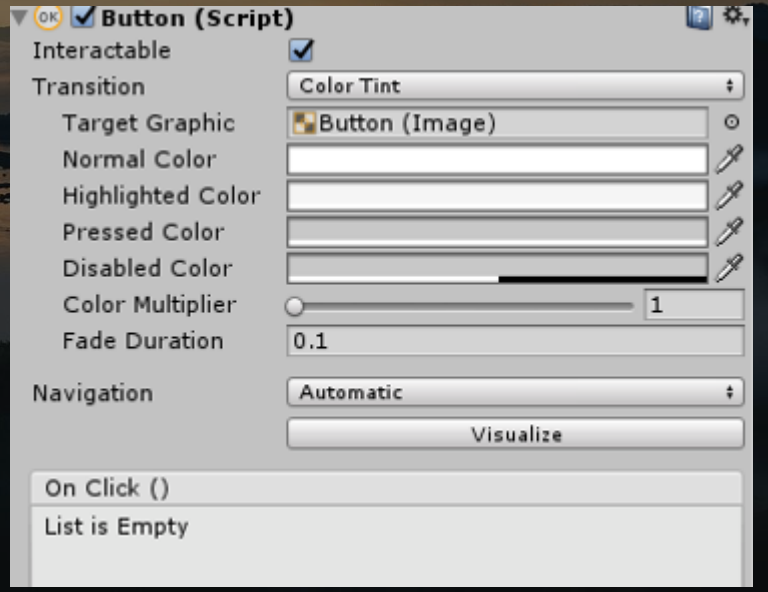
UGUI:

- 以Button举例
- Button组成
- Button控件： Rect Transform， Canvas Renderer， Image和Button
- Text控件： Rect Transform， Canvas Renderer， Text
- Button组件：

Interactable： 是否交互

Transition： 过渡方式

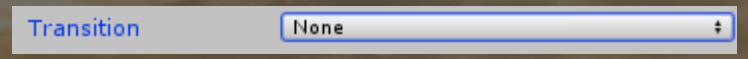
Navigation： 导航设置



UI

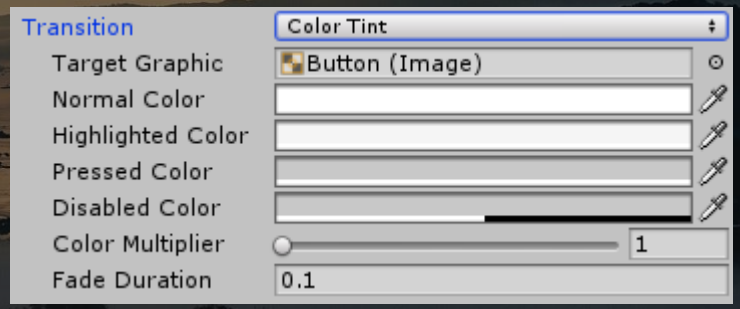
UGUI：

- 以Button举例
- Transition
- None：无过渡



UI

UGUI:
•以Button举例
•Transition
•None：无过渡
•Color Tint：颜色过渡
-Target Graphic：作用目标
-Normal Color：默认颜色
-Highlighted Color：高亮颜色，选中或鼠标进入
-Pressed Color：按下颜色
-Disabled Color：禁用颜色
-Color Multiplier：颜色切换系数，系数越大变化越快
-Fade Duration：淡出持续时间，颜色过渡时间



UI

UGUI:

•以Button举例

•Transition

•Sprite Swap：图片切换

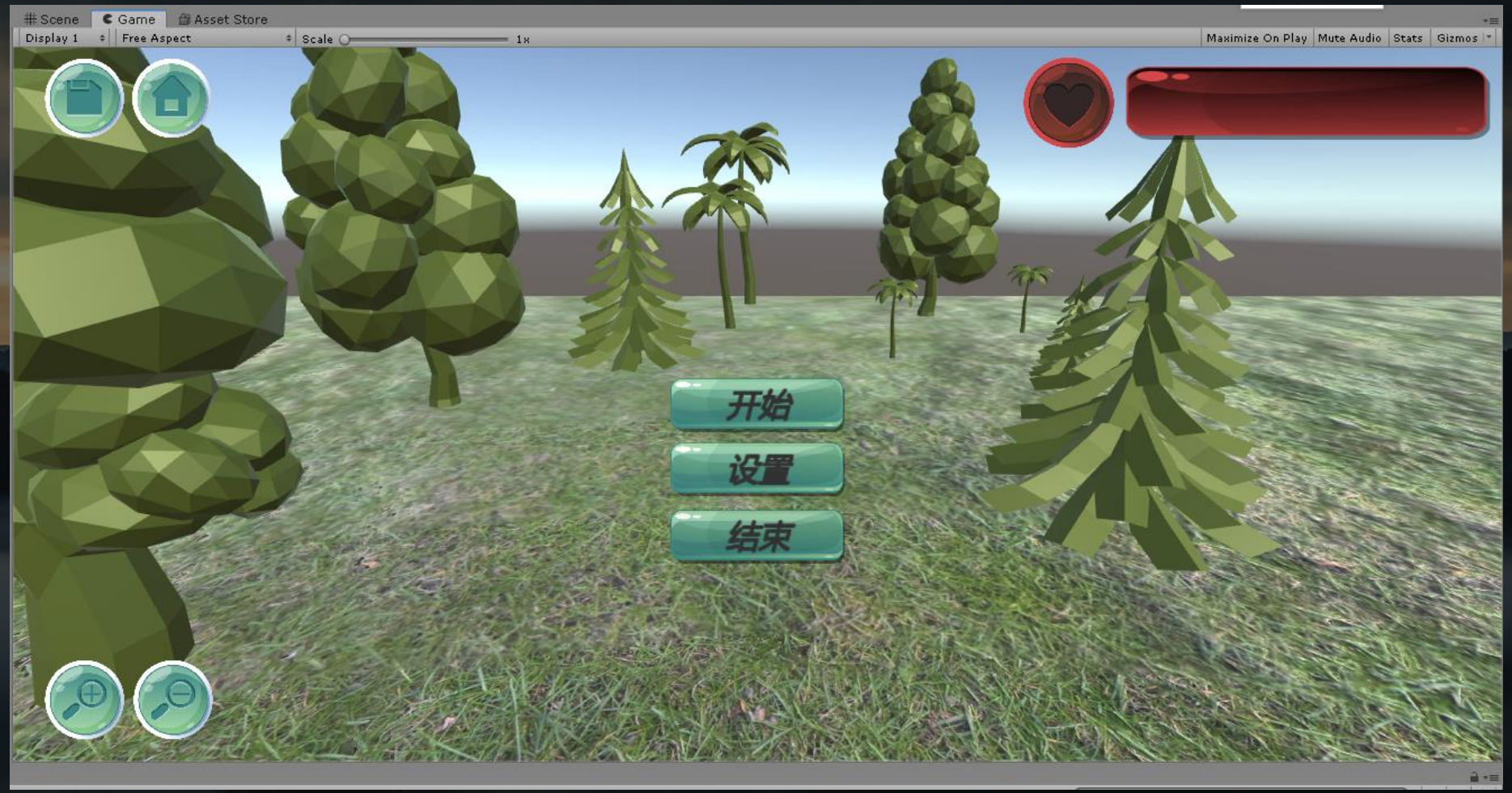
-Target Graphic：作用目标

-Highlighted Sprite：高亮图片

-Pressed Sprite：按下图片

-Disabled Sprite：禁用图片

综合练习


Scene
Game
Asset Store
Display 1
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos
开始
设置
结束

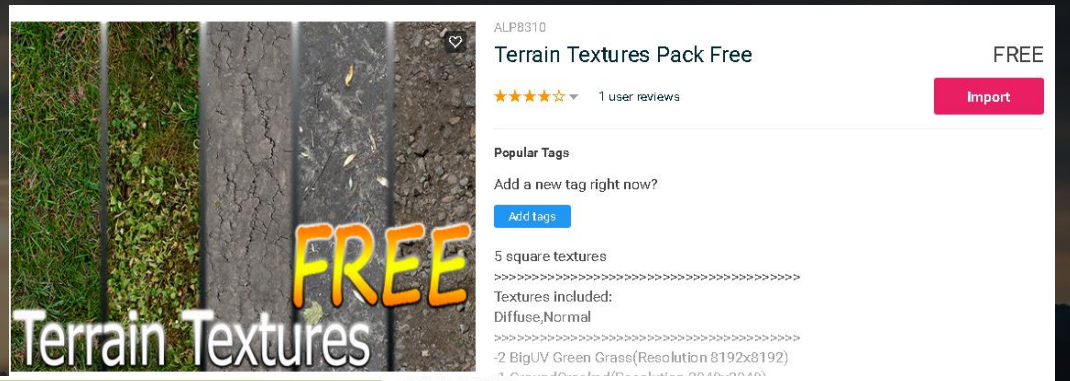
综合练习

Unity Store中下载以下三种免费资源库并导入工程

-Terrain Textures Pack Free

-Free Trees

-Icons UI



综合练习

Unity Store中下载以下三种免费资源库并导入工程
-Terrain Textures Pack Free
-Free Trees
-Icons UI
知识点:
Unity Store的使用
模型的创建及移动/旋转/缩放的应用
为模型赋予材质球，赋予贴图并且灵活运用材质球相关属性
摄像机视角调整，熟悉摄像机的移动及旋转
UGUI的创建
Icon位置的调整及对应组件属性的应用


END