U3D 笔记

郑华

2018 年 7 月 18 日

第一章　基础

1.1　Scene 场景- 世界变换

1.2　Console 调试信息 - Crtl+Shift+C

1.3　如何将脚本与具体对象绑定

1. 右键asset 文件夹，创建 C# 脚本
2. 编写脚本
3. 将asset 中的脚本拖拽到 hiearch 视图中的MainCamera 中
4. 如果脚本是作用于场景中的某个物体，**则将该脚本拖拽到该物体上**

1.4　序列化- [SerializedField]

通常情况下，GameObject 上挂的 MonoBehaviour 脚本中的**私有变量**不会显示在 *Inspector* 面板上，即不会被序列化。

但如果指定了 SerializedFiled 特性，就可以被序列化了。

```
public class Test : MonoBehaviour
{
    public string Name;
    [SerializeField]
    private int Hp;
}
```

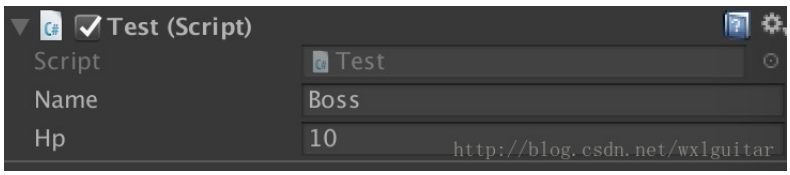

图 1.1: 序列化操作 -在 Inspector 上显示

1.5 常用技巧

- ctrl + d 复制
- shift + 鼠标 等比例缩放
- shift + alt + 鼠标 原地等比例缩放
- 在Unity 编辑器中输入汉字 需要借助其他文本拷贝粘贴
- q、w、e、r、t 在操作 UI 时尽量使用 T，以避免 z 轴发生的变化

第二章　事件

2.1　必然事件

继承自MonoBehaviour 类后，自动会**按序**提供以下方法:

- Awake(): 在加载场景时运行，用于在游戏开始前完成变量初始化、以及游戏状态之类的变量。
- Start(): 在第一次启动游戏时执行，用于游戏对象的初始化，在Awake() 函数之后。
- Update(): 是在每一帧运行时必须执行的函数，用于更新场景和状态。
- FixedUpdate(): 与Update() 函数相似，但是在固定的物理时间后间隔调用，用于物理状态的更新。
- LateUpdate(): 是在Update() 函数执行完成后再次被执行的，有点类似收尾的东西。

2.2　碰撞事件

U3D 的碰撞检测。具体分为三个部分进行实现，碰撞发生进入时、碰撞发生时和碰撞结束，理论上不能穿透

- OnCollisionEnter(Collision collision) 当碰撞物体间刚接触时调用此方法
- OnCollisionStay(Collision collision) 当发生碰撞并保持接触时调用此方法
- OnCollisionExit(Collision collision) 当不再有碰撞时，既从有到无时调用此函数

2.3　触发器事件

类似于红外线开关门，有个具体的范围，然后进入该范围时，执行某种动作，离开该范围时执行某种动作。类似于物体于一个透明的物体进行碰撞检测，理论上需要穿透，在 U3D 中通过

勾选 Is Trigger 来确定该物体是可以穿透的。

- OnTriggerEnter() 当其他碰撞体进入触发器时，执行该方法
- OnTriggerStay() 当其他碰撞体停留在该触发器中，执行该方法
- OnTriggerExit() 当碰撞体离开该触发器时，调用该方法

第三章　实体-人物、物体、组件

3.1　实体类

GameObject 类, 游戏基础对象，用于填充世界。

复制　Instantiate(GameObject) 或 Instantiate(GameObject, position, rotation)

- GameObject 指生成克隆的**游戏对象**，也可以是 **Prefab 的预制品**
- position 克隆对象的初始位置，类型为Vector3
- rotation 克隆对象的初始角度，类型为Quaternion

销毁　Destroy(GameObject xx)- 立即销毁 或Destroy(GameObject xx, Time time)- 几秒后销毁

可见否　通过设置该参数调整该实体是否可以在游戏中显示,具体设置方法为gameObject.SetActive(true

3.2　获取实体上的组件

调用方式　GameObject.GetComponent<Type>().xx = xx;

- cube1.GetComponent<RigidBody>().mass = 20; //设置重量
- cube1.GetComponent<BoxCollider>().isTrigger = true; //**开启 Trigger 穿透方式**检测
- cube2.GetComponent<Test>().enable = false; //**禁用 Test 脚本**

3.3 物理作用实体类

`Rigidbody` 类，一种特殊的游戏对象，该类对象可以在物理系统的控制下来运动。

AddForce() 此方法调用时`rigidBody.AddForce(1, 0, 0);`，会施加给刚体一个瞬时力，在力的作用下，会产生一个加速度进行运动。

AddTorque() 给刚体添加一个扭矩。

Sleep() 使得刚体进入休眠状态，且至少休眠一帧。类似于暂停几帧的意思，这几帧不进行更新、理论位置也不进行更新。

WakeUp() 使得刚体从休眠状态唤醒。

第四章　世界变换

4.1　Transform 类

位置　transform.position = new Vector3(1, 0, 0);

角度　transform.eulerAngles = new Vector3(x, y, z);

旋转　transform.Rotate(x, y, z);

缩放　transform.localScale(x, y, z); // 基准为 1、1、1，数为缩放因子。

平移　transform.Translate(x, y, z);

4.2　注意

在变化的过程中需要乘以 Time.deltaTime , 否则会出现大幅不连贯的画面。

第五章　键盘控制

5.1　普通按键 -keyDown(KeyCode xx)

方式一

- 定义按键码：KeyCode keycode;
- 判断键是否被按下：if(Input.GetKeyDown(keycode)){}
- 在Inspirit -> Keycode 指定关联按键

方式二

- 在Update 中更新添加如下代码
- if(Input.GetKeyDown(KeyCode.UpArrow))
- KeyCode.xx 包括了键盘所有的按键，常用的 AWSD 如下
  - if (Input.GetKeyDown(KeyCode.S))
  - if (Input.GetKeyDown(KeyCode.W))

5.2　根据输入设备 -getAxis()

参数分为两类：

一、触屏类

1. Mouse X 鼠标沿屏幕 X 移动时触发 Mouse Y 鼠标沿屏幕 Y 移动时触发 Mouse ScrollWheel 鼠标滚轮滚动是触发

```
float mouseX = Input.GetAxis("Mouse X");
float mouseY = Input.GetAxis("Mouse Y");
```

二、键盘类

1. Vertical 键盘按上或下键时触发
2. Horizontal 键盘按左或右键时触发

返回值是一个数，正负代表方向

第六章　时间

6.1　Time 类

该类是 U3D 在游戏中获取时间信息的接口类。常用变量如下：

表 6.1: 时间变量对照表

变量名	意义
time	游戏从开始到现在的运行时间，单位为秒
`deltaTime`	从上一帧到当前帧消耗的时间
fixedTime	最近 FixedUpdate 的时间，从游戏开始计算
`fixedDeltaTime`	物理引擎和 FixedUpdate 的更新时间间隔
timeSceneLevelLoad	从当前 Scene 开始到目前为止的时间
realTimeSinceStartup	程序已经运行的时间
`frameCount`	已经渲染的帧的总数

第七章　数学

7.1　Random 类

随机数类

7.2　Mathf 类

数学类

第八章　物理

8.1　流程

- RigidBody：创建，以完成受力接收。
- Physical Material：创建，以完成多种力的添加。
- Material：拖入材质球。

第九章　烘培

9.1　简介

只有静态场景才能完成烘培（Bake）操作，其目的是在游戏编译阶段完成光照和阴影计算，然后以贴图的形式保存在资源中，以这种手段避免在游戏运行中计算光照而带来的 CPU 和 GPU 损耗。

- **如果不烘培：**游戏运行时，这些阴影和反光是由 CPU 和 GPU 计算出来的。
- **如果烘焙：**游戏运行时，直接加载在编译阶段完成的光照和阴影贴图，这样就不用再进行计算，节约资源。

9.2　流程

第十章　寻路

10.1　简介

NPC 完成自动寻路的功能。

10.2　流程

- 将静态场景调至 (Navigation Static)
- 烘焙
- 添加 `Navigation Mesh Agent` 寻路组件
- 在脚本中设置组件的目标地址，添加目标

第十一章　UGUI

在脚本中使用时记得加上`using UnityEngine.UI`

11.1　场景跳转- SceneManager

11.2　锚点- 自适应屏幕

- 定义摄像机观察原点
- 其他物体与锚点 (Anchor) 距离保持一致，因此随着屏幕的变化，其他 UI 组件与屏幕锚点的距离始终保持一致
- 每个物体都可以定义自己的锚点

锚点类型

- 位置类型 左上角、中心等
- 拉伸类型 纵向拉伸适配、横向、整体

11.3　按钮

11.3.1　原始 Button

11.3.2　Image 等 -添加 button 组件

- `create -> UI -> Image`
- `Inspirit -> Add Component -> button`

11.3.3 添加事件处理脚本

- 书写脚本并添加到 Button gameObject 上
- 如果是 Button 组件的话直接在 button 组件上添加，如果是 Image 则添加 button 组件后再添加
- 添加脚本对象到`onClick()` 部分：`+ -> gameObject` 拖进来 `-> 选择脚本中的具体函数`

11.4 文本- Text

11.4.1 添加文字阴影 -shadow 组件

```
addComponent -> shadow
```

11.4.2 添加文子边框 -outline 组件

```
addComponent -> outline
```

11.5 图片- ImageView

11.6 选中标记- Toggle

Toggle 基本

Toggle Group

选项栏设定 将 panel 拖入 toggle 中的`value changed` 部分

预设 确定默认打开哪个 panel，然后将其`IsOn` 勾选，其余取消勾选

11.7 滚动区域、滚动条

11.8 其他工具条

11.9 布局- Layout

- 具体页面下创建空物体 `GameObject`
- 其次在`GameObject` 下添加组件 `-> grid layout group`
- 最后在这个`GameObject` 下创建出各种 Image 组件，然后这些组件将会以`grid layout` 的布局进行自动调整

11.9.1 grid layout group

- 调整`cell size` 进行调整子物件的大小
- `cell size` 的改变只影响子组件的第一层，既最下面一层

11.9.2 horizontal layout group

11.9.3 vertical layout group

第十二章　着色器渲染

第十三章　跨平台发布 apk

13.1　流程

- 安装 JavaSDK、Android Studio 并在 SDK manager 里添加对应的 API 包
- 在 unity 中的edit 选项下的preferences, 并选中External Tools 选项,配置JDK 和Android SDK 安装位置。
- 在 unity 中的File -> Build Settings 中，添加需要添加的场景，并选择对应的平台（Android，IOS）等
- 在 unity 中的Build Settings 中的Player Settings 设置以下几个重要内容。

    1. Company Name
    2. Product Name
    3. Default Icon :192×192
    4. Default Orientation
    5. Other Settings -> Identification : 修改为com.netease(Or Other).TestName(Or Other)

13.2　Apk 安装常见错误

http://mumu.163.com/2017/03/30/25905_680657.html