U3D笔记

郑华

2018年7月16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

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(tru
可以显示，false 则隐藏

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　时间

5.1　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 5.1: 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

第六章　数学

6.1　Random 类

随机数类

6.2　Mathf 类

数学类

第七章　物理

7.1　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。

第八章　烘培

8.1　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。

8.2　流程

第九章　寻路

9.1　简介

NPC 完成自动寻路的功能。

9.2　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

第十章　UGUI

第十一章　着色器渲染

第十二章　资源打包