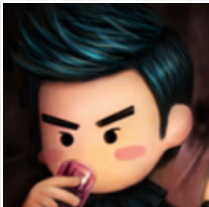


活到老，学到老！
居高声自远，非是籍秋风.....

目录视图 摘要视图 RSS 订阅

个人资料



nobcaup



访问：46130次
积分：977
等级：BLOG > 3
排名：千里之外

原创：51篇 转载：2篇
译文：0篇 评论：9条

文章搜索

文章分类

- 虚拟现实技术 (30)
- 开发平台 (2)
- 协议标准 (0)
- 操作系统 (0)
- 编程语言 (9)
- 通信技术 (0)
- 网络技术 (0)
- 算法实现 (1)
- 编译原理 (0)
- 计算机基础 (5)
- 【语言提升】英语|汉语 (0)
- 【行者无疆】旅游|摄影|美食 (0)
- 【行业远瞻】科技前沿|未来趋势 (2)

文章存档

- 2016年09月 (3)
- 2016年06月 (14)
- 2016年05月 (14)
- 2015年10月 (1)
- 2015年09月 (1)

展开

CSDN日报20170303——《百亿互金平台救火故事》 程序员2月书讯 社区有奖问答--一起舞动酷炫的iOS动画

【Unity3D】常用API学习笔记

标签：unity3d 方法 类 API Coroutine

2016-05-25 23:00 1892人阅读 评论(0) 收藏 举报

分类：

虚拟现实技术 (29)

版权声明：本文为博主原创文章，未经博主允许不得转载。

【Unity3D】常用API学习笔记

1、MonoBehaviour类（UnityEngine命名空间中定义）：

Awake：最开始调用，做一些初始化工作。建议少用，此刻物体可能还没有实例化出来，会影响程序执行顺序。

Start：不是很紧急的初始化，一般放在Start里面来做。仅在Update函数第一次被调用前调用。

Reset：用户点击检视面板的Reset按钮或者首次添加该组件时被调用。此函数只在编辑模式下被调用。Reset最常用于在检视面板中给定一个最常用的默认值。

Update：每一帧调用一次，帧间隔时间有可能改变。

FixedUpdate：以相同时间间隔调用，用在力学更新效果中。执行在Update之前。

LateUpdate：在Update和FixedUpdate调用之后调用。一般人物的移动放在Update中，而摄像机的跟进变化放到FixedUpdate中。确保两个独立，一前一后，不产生错误。

On开头的方法，是由其他事件触发调用的。

OnDestory：物体被删除时调用。

OnEnable：物体启用时被调用。

OnDisable：物体被禁用时调用。

OnGUI：这个函数会每帧调用好几次（每个事件一次），GUI显示函数只能在OnGUI中调用。

下图是单个脚本内部方法的调用顺序：

关闭

程序员培训机构



阅读排行

- VS2015 C#访问MySQL (2835)
- 【虚拟现实】Unity3D+V (2798)
- win7下搭建nfs-server的 (2385)
- MySQL基础及MySQL C (1893)
- 【Unity3D】常用API学习 (1892)
- 2013年中国互联网创业 (1748)
- 七种常用排序算法 (1490)
- 【那些年学过的计算机基 (1339)
- 【Unity3D】常用设计模 (1328)
- 【C#】笔试知识点 (1204)

评论排行

- 常思已过，调整姿态，坚 (4)
- VS2015 C#访问MySQL (2)
- 【Unity3D】ShaderLab: (1)
- 【单目全景相机】项目介 (1)
- 【C#】笔试知识点 (1)



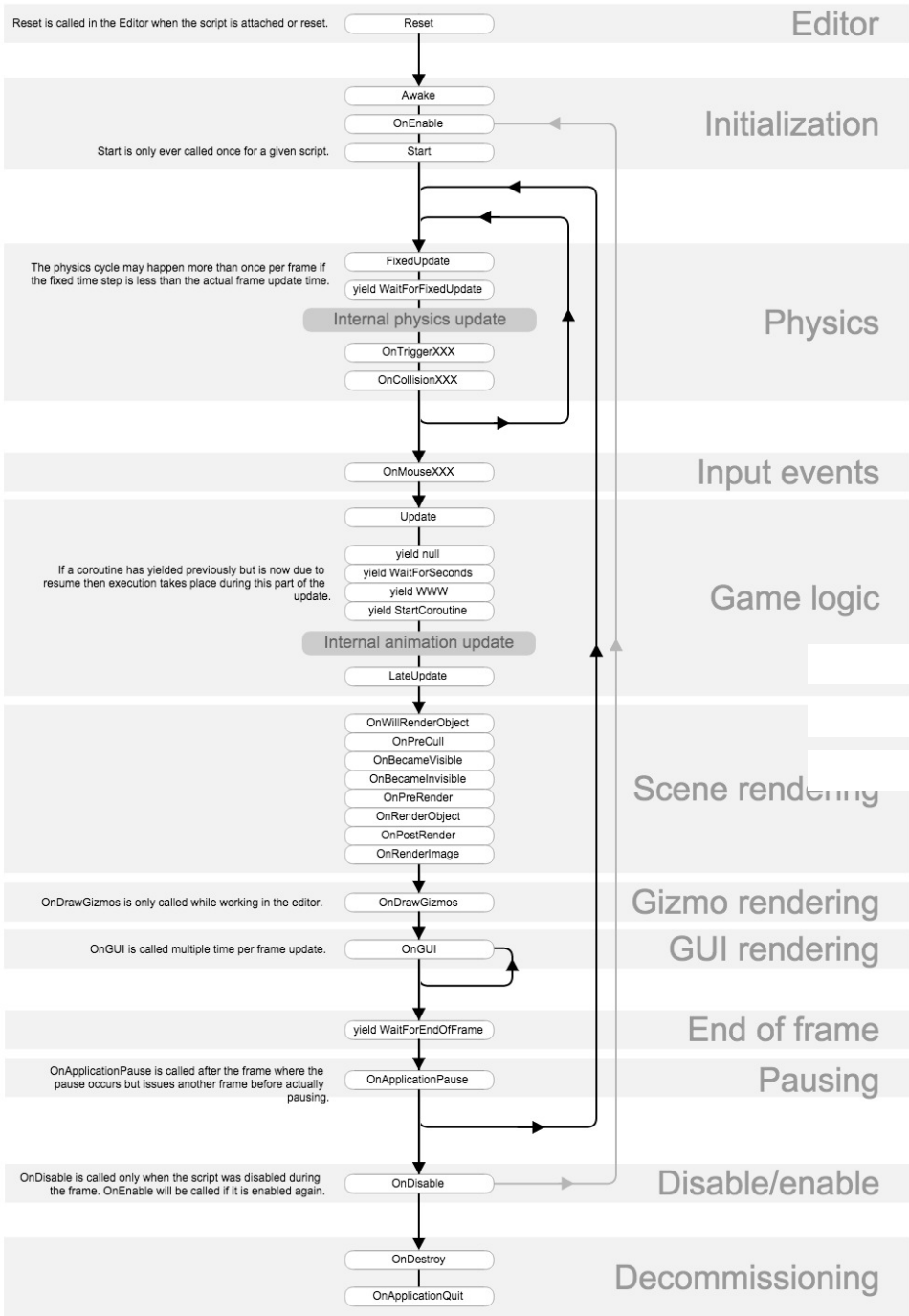
袖珍笔记本电脑



- * CSDN日报20170305——《谈谈学习方法》
- * 游戏跨服架构进化之路
- * 第一个PWA程序-聊天室
- * 安居客Android项目架构演进
- * 程序员转行为什么这么难

最新评论

- 【Unity3D】ShaderLab学习笔记 liujialin_art: 谢谢 分享这么好的学习平台
- 【单目全景相机】项目介绍 suhaiqiangxue: 最近也在搞这个，不知道可否分享一下预览的demo？万分感谢
- VS2015 C#访问MySQL数据库 qq_36714061: 请问能不能不使用datagridview控件直接取到数据库的信息呢，求大神帮忙解决这个问题
- VS2015 C#访问MySQL数据库 11ang: 按照你的方法无法运行啊，con.Open()处出现错误“MySql.Data.MySqlClien...
- 【C#】笔试知识点 a505050: 好熟悉的笔记内容
- 常思已过，调整姿态，坚定前行 nobcaup: @u010707157: 千万不能沦为好好先生，我现在觉得是一件很消极，很悲哀的事情。这段时间我会继...
- 常思已过，调整姿态，坚定前行 nobcaup: @u012377333: 谢谢！只有不断反思和改变，才会让自己更成熟。
- 常思已过，调整姿态，坚定前行 jiangedan: 看着你有种在照镜子的感觉，你的缺点就是我的缺点，我也要深刻剖析自己，找找原因，这样才能走得更远，加油...



Monobehaviour函数调用顺序

2、Input类：

getaxis、getkey、getbutton、getjoystick等函数。

3、Time类：

常用Time.deltaTime

在Update/LateUpdate中打印Time.deltaTime时间是不固定的，是一个随帧率变化的值。在FixedUpdate中打印Time.deltaTime时间是固定的。

4、单例模式singleton：

单例仅允许被实例一次，这就保证了他在各个程序模块间的唯一性。

```
[cpp]
01. private static ModelLocator instance;
02. public static ModelLocator getInstance{
03.     get{
04.         if(instance==null){
05.             instance=new ModelLocator();
```

程序员培训机构



常思己过，调整姿态，坚定前行
一枪尽骚、魂: 兄弟，加油

```
06.     }
07.     return instance;
08.     }
09. }
```

5、GameObject类：

gameObject (g小写) 代表当前脚本挂载的游戏对象本身。

GameObject (G大写) 代表游戏对象类。

查找GO并赋值Find族函数：

```
private GameObject go;
```

```
go = GameObject.Find("Cube");//根据名字查找对象
```

```
go = GameObject.FindGameObjectWithTag(string tag);//根据标签查找
```

```
go.activeSelf 游戏物体是否被激活 ( true or false )
```

```
go.activeInHierarchy 游戏物体所处层级的父级是否被激活 ( true or false )
```

6、Destroy方法：

```
Destroy(go);
```

```
Destroy(go, 3);//等待3s销毁
```

销毁一个游戏物体。

7、Transform对象：

位置transform.position(注意是小写t，是monobehaviour类中的默认字段，表示当前脚本挂在的游戏物体上的transform信息)

旋转transform.rotation

缩放transform.scale

向量及运算 Vector3

8、移动对象：

```
Transform.Translate
```

```
Transform.Rotate
```

插值运算：位置Vector3.Lerp 旋转Vector3.Slerp

```
Quaternion targetrotation = Quaternion.LookRotation(player.position - transform.position);//根据初始和目标位置计算出对应的旋转角度
```

9、Lerp插值运算：

插值运算不仅仅可以作为位置、旋转、缩放等计算，还可以做为灯光亮度等的差值计算，也就是说只要是从一个确定状态渐进过渡到另一个确定状态的计算，都可以用到插值运算。

位置插值：三维向量

```
Vector3 targetpostion = player.position + new Vector3(0, 2.42f, -2.42f);
```

```
transform.position = Vector3.Lerp(transform.position, targetpostion, speed * Time.deltaTime);
```

旋转插值：三维角度

```
Quaternion targetrotation = Quaternion.LookRotation(player.position - transform.position);
```

```
transform.rotation = Quaternion.Slerp(transform.rotation, targetrotation, speed * Time.deltaTime);
```

灯光亮度插值:浮点值

```
public float newIntensity = 5;
```

```
light.intensity = Mathf.Lerp(light.intensity, newIntensity, speed * Time.deltaTime);
```

值

颜色插值：

```
Color.Lerp(currentColor, newColor, speed * Time.deltaTime);
```

其他比如Material.Lerp、Mathf.InverseLerp等，可以通过查手册了解。

10、Instantiate实例化prefab对象：

所有的C#对象都是继承于object类，包括int、float这些函数类型，当然还有一些类。



关闭

程序员培训机构



```

[cpp]
01. static function Instantiate(original: Object, position: Vector3, rotation: Quaternion): Object;
02.
03. public GameObject Spawn()
04. {
05.     /* 生成prefab的实例化，因为默认是object类型，所以需要强转为GameObject */
06.     return GameObject.Instantiate(prefab, transform.position, transform.rotation) as GameObject;
07. }

```

11、其他方法：GameObject.GetComponent：通过游戏物体获取其组件CharacterController cc = this.GetComponent<CharacterController>();Animator animator = this.GetComponent<Animator>();Component.GetComponent：通过游戏物体的组件获取其其他组件Transform player = GameObject.FindGameObjectWithTag(Tags.player).transform;PlayerATKAndDamage playerAtkAndDamage = player.GetComponent<PlayerATKAndDamage>();//PlayerATKAndDamage是一个脚本AddForce：添加力AddTorque：添加扭矩

12、协同（协程）：一般用来在脚本中增加延时效果。因为在Start()或者Update()中是不能直接延时的（WaitForSeconds()）等待某个操作结束之后再执行代码字符串做为参数：

```

[cpp]
01. void Start ()
02. {
03.     StartCoroutine("DoSomething", 2.0);
04.     yield WaitForSeconds (1);//可以在任意位置使用yield语句。yield的返回值控制何时恢复执行。这里等待1s之后才会接着执行下面的语句。
05.     StopCoroutine("DoSomething");
06. }
07.
08. void DoSomething (float someParameter)
09. {
10.     while (true)
11.     {
12.         print("DoSomething Loop");
13.         // 停止协同程序的执行并返回到主循环直到下一帧。
14.         yield;
15.     }
16. }

```

IEnumerator做为参数：

```

[cpp]
01. IEnumerator Start()
02. {
03.     StartCoroutine("DoSomething", 2.0F); //StartCoroutine(DoSomething(2.0F)); 使用IEnumerator
    做参数不能用StopCoroutine停用。
04.     yield return new WaitForSeconds(1);
05.     StopCoroutine("DoSomething"); //请注意只有StartCoroutine使用一个字符串方法名时才能用
    StopCoroutine停用之。
06. }
07.
08. IEnumerator DoSomething(float someParameter)
09. {
10.     while (true) {
11.         print("DoSomething Loop");
12.         yield return null;
13.     }
14. }

```

开启协同：

StartCoroutine(string methodName)：字符串作为参数可以开启线程并多只能传递一个参数，并且性能消耗会更大一点

StartCoroutine(IEnumerator routine)：只能等待协程的结束而不能随时

中止协同：



关闭

程序员培训机构

StopCoroutine(string methodName)：中止一个协同，只能终止该MonoBehaviour中的协同程序

StopAllCoroutines()：中止所有协同，只能终止该MonoBehaviour中的协同程序

将协同程序所在gameObject的active属性设置为false，当再次设置active为ture时，协同程序并不会再开启。

13、yield：和协同密切相关的一个概念，一个协同程序在执行过程中,可以在任意位置使用yield语句。yield的返回值控制何时恢复协同程序向下执行。

yield不可单独使用

需要与return配合使用，例如：

1 yield return 0; //等0帧

2 yield return 1; //等1帧

3 yield return WaitForSeconds(3.0); //等待3秒

4 yield return null; //立即返回调用点

所有使用yield的函数必须将返回值类型设置为IEnumerator类型，例如：

```
IEnumerator DoSomethingInDelay() {...}
```

当然，你也可以把Start()返回值定义为IEnumerator类型，那么在Start里面也可以使用yield延时返回，但不推荐这样做：

```
[cpp]
01. IEnumerator Start()
02. {
03.     StartCoroutine("DoSomething", 2.0f); //StartCoroutine(DoSomething(2.0f)); 使用IEnumerator
    做参数不能用StopCoroutine停用。
04.     yield return new WaitForSeconds(1);
05.     StopCoroutine("DoSomething"); //请注意只有StartCoroutine使用一个字符串方法名时才能用
    StopCoroutine停用之。
06. }
```

你也可以把StartCoroutine和yield return结合起来使用，保证函数执行顺序，这样调用能保证，init1，init2，init3一个一个的执行，不至于出现后面执行的代码引用一个前面未初始化的变量：

```
[cpp]
01. IEnumerator Init()
02. {
03.     yield return StartCoroutine(init1());
04.     Debug.Log("init1 finish");
05.     yield return StartCoroutine(init2());
06.     Debug.Log("init2 finish");
07.     yield return StartCoroutine(init3());
08.     Debug.Log("init3 finish");
09. }
10.
11. IEnumerator init1()
12. {
13.     // 模拟初始化
14.     yield return new WaitForSeconds(2); //
15. }
16. IEnumerator init2()
17. {
18.     // do something..
19.     yield return new WaitForSeconds(2); //
20. }
21. IEnumerator init2()
22. {
23.     // do something..
24.     yield return new WaitForSeconds(2); //
25. }
```

好了，就先介绍这些。



程序员培训机构



上一篇 [【Unity3D】基础知识学习笔记](#)
下一篇 [【Unity3D】Unity3D工具、Mono工具、内部脚本工作原理以及跨平台特性](#)

我的同类文章

虚拟现实技术 (29)

• [【单目全景相机】Unity3D...](#)

2016-09-16

阅读 419

• [【单目全景相机】项目介绍](#)

2016-09-03

阅读 361

• [U3D服务器端开发知识点总...](#)

2016-06-19

阅读 693

• [VS2015 C#访问MySQL数...](#)

2016-06-17

阅读 2820

• [MySQL基础及MySQL C A...](#)

2016-06-16

阅读 1892

• [【3D计算机图形学】变换...](#)

2016-06-14

阅读 726

• [【单目全景相机】友盟分享...](#)

2016-09-16

阅读 459

• [【C#】笔试知识点](#)

2016-06-21

阅读 1200

• [U3D编译Web PC IOS And...](#)

2016-06-19

阅读 932

• [U3D前后端开发知识体系](#)

2016-06-17

阅读 449

• [个人代码托管和版本控制](#)

2016-06-15

阅读 533

更多文章



袖珍笔记本电脑





程序培训班



程序开发培训



九寨沟五日游



软件工程师月薪



编程自学



参考知识库



Go知识库

1946 关注 | 868 收录



Unity3D知识库

3226 关注 | 507 收录

猜你在找

- 移动端游戏架构设计

unity3D一游戏 AR VR在线就业班 unity引擎

实战进阶学习Unity3d游戏开发

深入浅出Unity3D——第一篇

Unity3D着色器程序设计-CG版
- Unity3D 学习笔记8 UGUI控制和按钮的监听系统

unity3d动画插件Dotween使用学习笔记

Unity3d学习笔记1

unity3d学习笔记十四--NGUI用Sprite动画和屏幕自适

Unity3D学习笔记之六创建更多的Prefab




高级ui设计培训



眼睛近视怎么恢复



vr培训班



瘦脸隆鼻



学习电脑编程



程序培训班

查看评论

暂无评论

您还没有登录,请[登录](#)或[注册](#)

关闭

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS S...
OpenStack VPN Spark ERP IE10 Eclipse CRM JavaS...
WAP jQuery BI HTML5 Spring Apache .NET API F...
LBS Unity Splashtop UML components Windows Mobile
CloudStack FTC coremail OPhone CouchBase 云计算 iO...
SpringSide Maemo Compuware 大数据 aptech Perl Tom...
HBase Pure Solr Angular Cloud Foundry Redis Scala

程序员培训机构



☐



关闭

