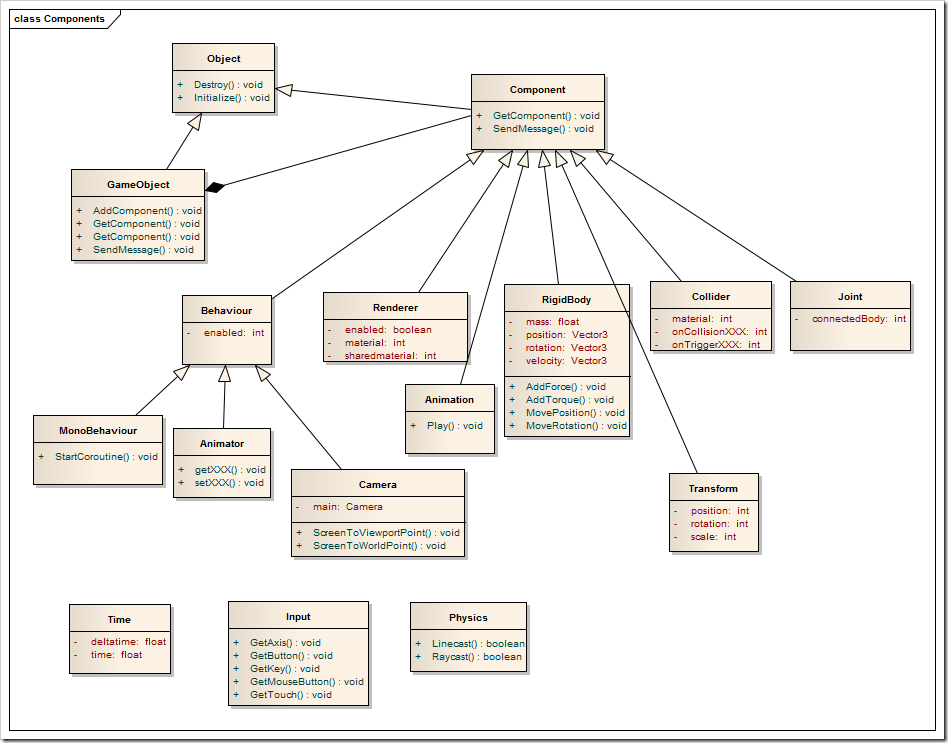
# Unity学习

## 一. unity3D的重要类

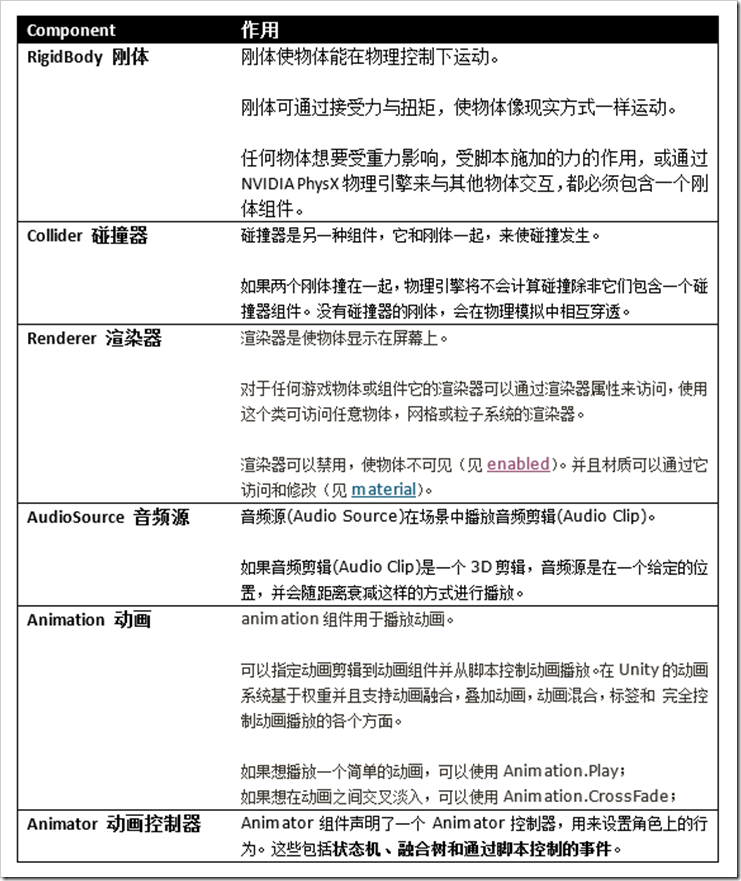


**Figure 1 unity的重要类**

Unity是一个Component-Based的游戏引擎，所以游戏中所有的物体都是一个GameObject，为了给这个GameObject附加上各种各样的属性，所以我们引入了Component这个概念。

GameObject是由Component组合成的，Component的生命周期和GameObject息息相关。一旦GameObject的Destroy方法，它的子对象和对应的所有Component都会被销毁，同时，我们也可以一次只销毁一个单独的Component。

### 1. Component有如下这些种类:



组件附属于游戏物体.把一个 [Renderer](http://game.ceeger.com/Script/Renderer/Renderer.html)(渲染器)组件附到游戏对象,可以使游戏对象显示到场景,附一个 [Camera](http://game.ceeger.com/Script/Camera/Camera.html)(摄像机)可以把物体变成一个摄像机物体.所有脚本都是组件,因此都能附到游戏对象上.

常用的组件可以通过简单的成员变量取得:

附在游戏对象上的组件或脚本可以通过GetComponent获取.如下代码示例：

using UnityEngine;

using System.Collections;

public class example : MonoBehaviour {

void Awake() {

transform.Translate(0, 1, 0);

GetComponent<Transform>().Translate(0, 1, 0);

}

}

### 2. Input和InputManager

Unity支持，键盘，操纵杆和游戏手柄输入。

在**输入管理器（Input Manager）**可以创建虚拟轴和按钮，并终端用户可以在屏幕配置对话框配置键盘输入。

如果想添加新的虚拟轴，选择菜单Edit->Project Settings->Input menu。这里可以改变每个轴的设置。即可进入Input Manager的配置界面。

从脚本，所有虚拟轴通过它们的名字（name）来访问。

默认值：

**Horizontal** and **Vertical** ：

水平和垂直被映射到w, a, s, d键和方向键

**Fire1**, **Fire2**, **Fire3** ：

Fire1, Fire2, Fire3被分别映射到Ctrl，Option（Alt）和Command键

**Mouse X** and **Mouse Y** ：

Mouse X 和 Mouse Y被映射到鼠标移动增量

**Window Shake X** and **Window Shake Y**

Window Shake X 和 Window Shake Y 被映射到窗口的移动

### 3. Time

Time类是Unity中的一个全局变量，它记载了和游戏相关的时间，帧数等数据。

[Time](http://game.ceeger.com/Script/Time/Time.html)类包含一个非常重要的变量叫deltaTime.这个变量包含从上次调用Update 或FixedUpdate到现在的时间(根据你是放在Update函数还是FixedUpdate函数中).(另注: Update每帧调用一次)

依照上面的例子，使得物体在一个匀速的速度下旋转，不依赖帧的速率，如下：

using UnityEngine;

using System.Collections;

public class example : MonoBehaviour {

void Update() {

transform.Rotate(0, 5 \* **Time.deltaTime**, 0);

}

}

### 4. Physics和Transform

Physics类是Unity重的一个工具函数类，它主要提供了Linecast和Raycast两种射线投射方式。

**Linecast：**

以投射的起始位置和终止位置为参数，来判断这个投射有没有和某个Collider发生了碰撞。

**Raycast：**

以投射的起始位置和投射方向为参数，来判断这个投射有没有和某个Collider发生了碰撞。

using UnityEngine;

using System.Collections;

public class Example : MonoBehaviour {

void Update() {

// 使用Raycast

Vector3 fwd = transform.TransformDirection(Vector3.forward);

if (Physics.Raycast(transform.position, fwd, 10))

print("There is something in front of the object!");

// 使用Linecast

Transform target;

if (!Physics.Linecast(transform.position, target.position))

ProcessData.AndDoSomeCalculations();

}

}

在Physics这个模块中，有三个Component是至关重要的，分别是RigidBody，Collision，Joint。在新的版本中，又引入了RigidBody2D，Collision2D，Joint2D这些Component来处理2D中的Physics事件。

这三个类都是处理物理相关的事件的，那么它们有什么区别呢？

**RgidBody**

是作为一个受力物体而存在的，所以可以向一个RigidBody施加Force（力），Drag（阻力）。同时RigidBody还有 velocity （速度），mass（质量），position（位置），旋转（rotation）等等。

**Collider**

是为了处理物理中的碰撞事件而出现的类，就像上面表格中所说的，如果没有Collider，两个RigidBody之间是无法发生碰撞的。对同一个GameObject可以绑定多个Collider构建更加复杂的碰撞体结构。Collider另外一个很值得注意的就是我们可以为Collider设置material，即Collider的物理材质。 物理材质用于调整摩擦力和碰撞单位之间的反弹效果。

当发生碰撞时，会触发回调函数***OnCollisionEnter，OnCollisionStay，OnCollisionExit***等等。这几个函数与OnTriggerXXX的区别会在接下来的博客中提到。

**Joint**用于连接两个RigidBody，当Joint断掉的时候会触发***OnJointBreak***的回调函数。

***注意：中Time，Input，Physics都是Unity中的全局变量。***

## 二．MonoBehaviour:

### 1. gameObject 是 unity 里真实存在的物件

GameObject是由Component组合而成的，GameObject本身必须有Transform的Component，这也加深了我们对GameObject的理解，即GameObject是游戏场景中真实存在，而且有位置的一个物件。

Monobehaviour是用来操作 GameObject 的脚本。

### 2. MonoBehaviour的生命周期

MonoBehaviour是Unity中所有脚本的基类，如果你使用JS的话，脚本会自动继承MonoBehaviour。如果使用***C#的话，你需要显式继承MonoBehaviour。***

在我们使用MonoBehaviour的时候，尤其需要注意的是它有哪些可重写函数，这些可重写函数会在游戏中发生某些事件的时候被调用。我们在Unity中最常用到的几个可重写函数是这几个：

**Awake：**

当一个脚本实例被载入时Awake被调用。我们大多在这个类中完成成员变量的初始化

**Start：**

仅在Update函数第一次被调用前调用。因为它是在Awake之后被调用的，我们可以把一些需要依赖Awake的变量放在Start里面初始化。 同时我们还大多在这个类中执行StartCoroutine进行一些协程的触发。要注意在用C#写脚本时，必须使用StartCoroutine开始一个协程，但是如果使用的是JavaScript，则不需要这么做。

**Update：**

当MonoBehaviour启用时，其Update在每一帧被调用。

**FixedUpdate：**

当MonoBehaviour启用时，其 [FixedUpdate](http://game.ceeger.com/Script/MonoBehaviour/MonoBehaviour.FixedUpdate.html)在每一固定帧被调用。

**OnEnable：**

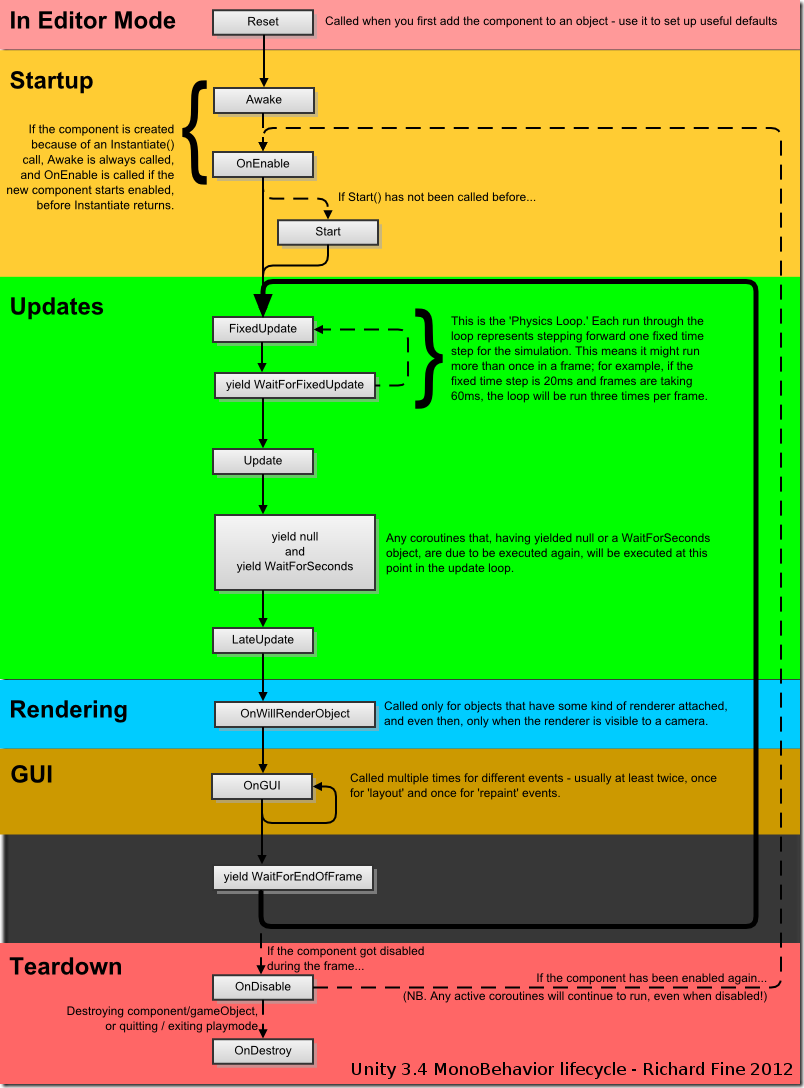
当对象变为可用或激活状态时此函数被调用。

**OnDisable**：

当对象变为不可用或非激活状态时此函数被调用。

**OnDestroy**：

当MonoBehaviour将被销毁时，这个函数被调用。



**Figure 2 MonoBehaviour的生命周期**

### 3. 注意事项

私有（private）和保护（protected）变量只能在专家模式中显示.属性不被序列化或显示在检视面板.

不要使用命名空间（namespace）

记得使用 **缓存组件查找，**即在MonoBehaviour的长远方法中经常被访问的组件最好在把它当作一个私有成员变量存储起来。

在游戏里经常出现需要检测敌人和我方距离的问题，这时如果要寻找所有的敌人，显然要消耗的运算量太大了，所以最好的办法是将攻击范围使用Collider表示，然后将Collider的 isTrigger设置为True。最后使用OnTriggerEnter来做攻击范围内的距离检测，这样会极大提升程序性能。

## 2. 关于碰撞的设置：

静态的物体，选择静态碰撞(不选 isTragger)

动态的物体(选择 isTragger)，选择添加Rigidbody, （use gravity, is Kinematic）

## 3.灯光：

## 4.摄像机：

Camera:可以和player 绑定，来达到同时移动的目的。

法1：最简单的办法：将camera设置为player的子类

法2：设置camera与player的关联：创建cameraController脚本给camera (for example:)

using UnityEngine;

using System.Collections;

public class CameraController : MonoBehaviour {

public GameObject player;

private Vector3 offset;

void Start ()

{

offset = transform.position - player.transform.position;

}

void LateUpdate () // lateUpdate() 指每一帧当别的都加载完了之后再加载

{

transform.position = player.transform.position + offset;

}

}

## 5.对撞机：

## 6.触发器：

## 7.刚体：