第四章

1 Unity脚本概述

上节我们介绍了如何去创建一个脚本，以及简单的介绍了一点Unity的脚本知识，这节将详细的介绍关于脚本的一些知识。

与其它常用的平台有所不同，Unity中的脚本程序如果要起作用，主要途径为将脚本挂载到特定的对象上。这样，脚本中的方法在特定的情况下就会被回调，实现特定的功能。

其实上节所说的几个回调方法一般是位于MonoBehavior类的子类中的，也就是说开发脚本代码时，主要是继承自MonoBehavior类并重写其中特定的方法。

2 Unity中C#脚本的注意事项

Unity中C#脚本的运行环境使用了Mono技术(Mono是指由Novell公司领导的，一个致力于.NET开源的工程)，可以在Unity脚本中使用.NET所有相关的类。但Unity中C#的使用和传统的C#有一些不同。

再次强调，C#中的类名必须手动编写，在创建脚本的时候，脚本的类名会与其文件的名称一样。因此不要轻易改变文件名或者类名，在创建脚本之前，应该先预先将类名设计好。

用于初始化脚本的代码必须置于Awake或者Start方法中。两者的不同之处在于，Awake方法是在加载时运行，Start方法是场景加载后第一次调用Update之前调用。因此Awake方法总是在Start方法前调用。不要在构造函数中初始化任何变量，要用Awake或者Start方法来实现。即便是在编辑模式，Unity扔会自动调用构造函数，这通常是在一个脚本编译之后，因为需要调用脚本的构造函数来取回脚本的默认值。无法预计何时调用构造函数，它或许会被预制件或未激活的游戏对象所调用。

而在单一(例)模式下使用构造函数可能会导致严重的后果，会带来类似随机的空引用异常。因此想实现单一模式就不要用构造函数，要用Awake或者Start方法。

通过观察上节的代码可以看出，只有是public类型的变量才能在检视视图中通过拖拽来完成赋值操作，而private或者protected类型的成员变量不能查看。

3 Unity脚本调试

Unity中C#代码的调试与传统的C#调试有所不同。Unity有一个控制台，Console，在这个控制台可以看到脚本中精确的错误，包括位置、行数。比如以下错误代码。

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class TestDebug : MonoBehaviour {
5      int a;
6      // Use this for initialization
7      void Start () {
8          a = 0.2;
9      }
10
11     // Update is called once per frame
12     void Update () {
13
14     }
15 }
16
```

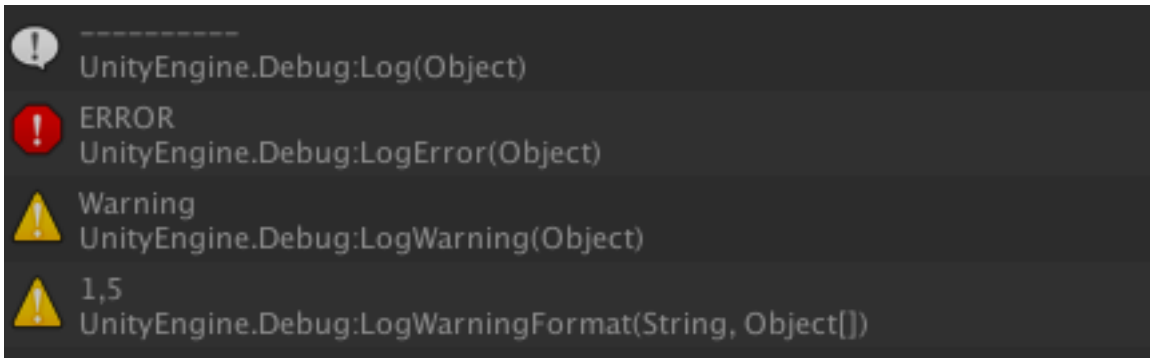
将一个浮点数据赋值到整型变量(注意这里没有什么隐式转换)，然后将该脚本挂载到游戏中的某个对象上，控制台输出如下。

双击错误的信息，Unity会直接将错误定位到脚本对应的错误的地方，非常的智能、方便。

Unity中，可以使用print()和Debug.Log()打印调试信息。但print()只能在Mono类中使用，所以一般情况最好使用Debug.Log()。同时也可以使用Debug.LogWarning()和Debug.LogError()打印警告和错误信息。Unity中通过Debug.Break()设置断点。如果想查看特定情况发生时对象属性的变化时，用断点可以快速的完成。例,

```
using UnityEngine;
using System.Collections;

public class TestDebug : MonoBehaviour {
    int a;

    // Use this for initialization
    void Start () {

//      Debug.Break();
        Debug.Log("----------");
        Debug.LogError("ERROR");
        Debug.LogWarning("Warning");
        Debug.LogWarningFormat("{0},{1}",1,5);
//      a = 0.2;
    }

    // Update is called once per frame
    void Update () {

    }
}
```

控制台信息:



4 Unity脚本的基础语法

Unity中很多对游戏对象的操作都是通过在脚本中修改对象的Transform与Rigidbody参数来实现的。例如，让物体延x轴顺时针旋转一定角度，则可以使用如下的C#代码片段来实现。

```
using UnityEngine;
using System.Collections;

public class TransformDemo : MonoBehaviour {
    public float angelS = 20;
    // Use this for initialization
    void Start () {
        this.transform.Rotate(20,0,0);//对象在原来的基础再延顺时针旋转20°
    }

    // Update is called once per frame
    void Update () {
        this.transform.Rotate(1,0,0);//对象每帧都旋转1°
    }
}
```

如果想让对象延某个方向做平移运动。可以在Start函数中添加如下代码，如果想让其延某个方向一直平移，可以将代码写入Update函数里。

```
this.transform.Translate(1,0,0);
```

用于旋转的Rotate方法和用于移动的Translate方法都有4个参数的重载形式。第四个参数为Space枚举类型，一个是Self，另外一个是

World，后者表示被应用于相对于世界坐标系统，前者则是自身。默认第四个参数不传的话是Self。

4.1 Time类

在Unity中记录时间的需要用Time类。Time类中比较重要的变量为deltaTime(only read),它指的是Update或者FixedUpdate调用的时间间隔。可以用Time.deltaTime获取该值。

问题:如何让一个对象匀速运动？匀速旋转?

4.2 访问游戏对象的组件

组件属于游戏对象，比如把一个Renderer(渲染器)组件附加到游戏对象上，可以使游戏对象显示到游戏场景中；把Camera(摄像机)组件附加到游戏游戏对象上可以使该对象具有摄像机的所有属性/功能。由于所有的脚本都是组件，因此一般脚本都可以附加到游戏对象上。

常用的组件可以通过简单的成员变量取得。

在Unity中，附加到游戏对象上的组件可以通过GetComponent方法获得，具体操作时可以使用如下C#代码片段来实现。

```
void Update () {
//
this.gameObject.transform.Rotate(60*Time.deltaTime,0,
0);//对象每帧都旋转20°
//      transform.Translate(0.01f,0,0);

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

同样的，也可以通过GetComponent方法获取其他的脚本。比如有一个HelloWorld脚本，里面有一个sayHello方法。HelloWorld脚本要与调用它的脚本附加在同一游戏对象上，具体操作时可以使用如下的C#代码。

```
using System.Collections;
public class TransformDemo : MonoBehaviour {
    public float angelS = 20;
    // Use this for initialization
    void Start () {
    }

    void Update () {
        HelloWorld hello =
GetComponent<HelloWorld>();
        hello.SayHello();
    }
}
```

关于组件的详细知识，我们后面会学习到，在本节只要知道在脚本当中用GetComponent泛型就能获取到组件。

4.3 访问其他游戏对象

(1)通过属性查看器(检视视图)指定参数

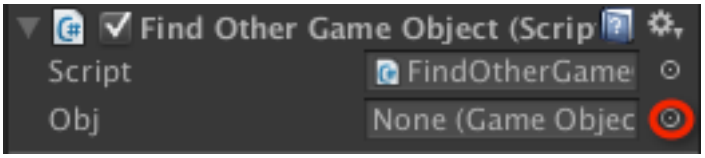
代码中声明为public类型的游戏对象的引用，在检视视图中就能看到。只要将该引用指向一个游戏对象，那么脚本中访问这个对象的引用就相当于访问其他游戏对象。例如下面，

代码段:

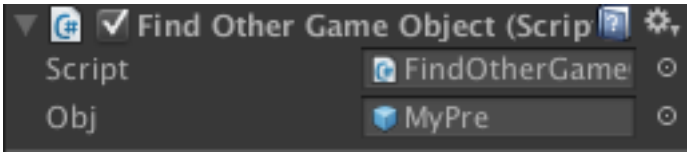
```
using UnityEngine;
using System.Collections;
public class FindOtherGameObject : MonoBehaviour {
    public GameObject obj;
    // Use this for initialization
    void Start () {
    }
    void Update () {
        obj.transform.Translate(1,0,0);
    }
```

}

在检视视图中，将该C#脚本组件挂载到游戏对象上，如图，



赋值后，



(2)确定对象的层次关系

游戏对象，在层级视图的对象列表中，存在有父子关系，在代码中可以通过获取Transform组件来找到对象或者父对象。

```
using UnityEngine;
using System.Collections;

public class Cengci : MonoBehaviour {
    private Transform objT;
    // Use this for initialization
    void Start () {
        objT = this.transform.Find(“Cube”);//
transform.Find获取的是名称为Cube的子对象

    }
    // Update is called once per frame
    void Update () {
        objT.Rotate(20*Time.deltaTime,0,0);
        objT.parent.Translate(1,0,0);//objT的父对象一直
移动
    }
}
```

一旦获取到”Cube”子对象，就可以通过GetComponent方法获取到”Cube”子对象的所有组件。

另外，Unity中可以通过遍历transform对象来获取其子对象，遍历代码如下，

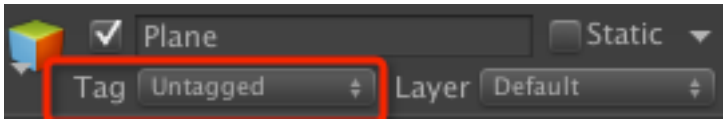
```
    foreach(Transform child in transform){
            child.Translate(0.1f,0.1f,0);
        }
```

(3)通过名字或标签获取游戏对象

Unity脚本中可以使用FindWithTag方法和Find方法来获取游戏对象，FindWithTag方法获取指定标签的游戏对象，Find方法获取指定名字的游戏对象。如下代码，

```
using UnityEngine;
using System.Collections;
public class FindObject : MonoBehaviour {
    protected GameObject nameT;
    // Use this for initialization
    void Start () {
        nameT = GameObject.Find("Plane");
    }
    // Update is called once per frame
    void Update () {
        nameT.transform.Translate(0,0,0.1f);
    }
}
```

前面说了，FindWithTag方法为获取指定标签的游戏对象，关于设置游戏对象的标签需要了解一下，在检视视图中，可以设置对象的标签，如下如所示，



红色区域展开就能设置Tag值了，既标签。如下，

```
using UnityEngine;
using System.Collections;
public class FindObjWithTag : MonoBehaviour {
    private GameObject objT;
    // Use this for initialization
    void Start () {
        objT = GameObject.FindWithTag("Test");
    }
    // Update is called once per frame
    void Update () {
        objT.transform.Translate(0.1f,0,0);
    }
}
```

留个问题？将多个游戏对象的标签都设置为一样，例如，“Test”，会有什么样的效果？

(4)通过传递参数来获取游戏对象

一些事件回调方法的参数中包含了特殊的游戏对象或组件信息，例如触发碰撞事件的Collider组件。在OnTriggerStay方法的参数中有一个碰撞体参数，可以通过这个参数获取到碰撞的刚体(这块内容需要我们学习了物理组件后才会明白)。

(5)通过组件名称获取游戏对象

Unity脚本中可以通过FindObjectOfType方法和FindObjectsOfType方法来找到挂载特定类型组件的对象。FindObjectsOfType方法可以获取所有挂载指定类型组件的对象，返

回的是一个数组。FindObjectOfType方法获取挂载指定类型组件的第一个游戏对象。具体操作如下所示。

```
using UnityEngine;
using System.Collections;
public class Test : MonoBehaviour {
    private int num1;
    // Use this for initialization
    void Start () {
        Test[] tests = FindObjectsOfType<Test>();
        Test t = FindObjectOfType<Test>();
        num1 = tests.Length;
        Debug.Log(num1);
        Debug.LogFormat("game{0} is
{1}",gameObject.name,t.name);
    }

    // Update is called once per frame
    void Update () {

    }
}
```