4.4 Vector3类(向量)

3D游戏开发中经常需要用到向量和向量的运算，Unity中提供了完整的向量以及向量操纵方法，分别为表示二维向量的Vector2类和表示三维向量的Vector3类。二维和三维其实都差不多，这里我们只学习Vector3类就可以了。

创建一个Vector3的实例用以下方式：

(1)

```
Vector3 vec1 = new Vector3();
vec1.x = 1;
vec1.y = 2;
vec1.z = 3;
```

(2)

```
Vector3 vec2 = new Vector3(1,2,3);
```

Vector3类中也定义了一些常量，从Mono编辑器中可以跳进去看到如下，

```
public static Vector3 back...

public static Vector3 down...

public static Vector3 forward...

[Obsolete ("Use Vector3.forward instead.")]
public static Vector3 fwd...

public static Vector3 left...

public static Vector3 one...

public static Vector3 right...

public static Vector3 up...

public static Vector3 zero...
```

通过脚本，我们可以打印这些常量的值。

关于向量操纵，还有很多，像我们高中数学中向量的乘积、+、-、*、/等。常用的方法有Distance求点的距离、Max/Min返回两个向量中最长/最短的向量。具体可以看课堂上给的中文API文档，仅供参考。

4.5 成员变量和静态成员变量

一般情况下，定义在方法体外的变量是成员变量，如果这里变量的访问权限是public，就可以在检视视图中查看到，这样就可以在检视视图中修改，它的值会一并自动保存。

如果声明的是一个组件类型的变量，需要在检视视图中给这个变量赋值一个对象。具体操作，前面的例子已经包含，不再赘述。

private和protected是不能再检视视图中查看到的。

C#脚本中可以通过static关键字来创建全局变量，这样就可以在不同脚本间调用这个变量。如下，

public static int test;

如果想从另外一个脚本调用变量”test”，可以通过”脚本名.变量名”的方法来调用。

4.6 实例化游戏对象

第三章提到过动态创建对象的方法Instantiate，以下就给出一段代码，不再赘述。

```
using UnityEngine;
using System.Collections;
public class InstantiateDemo : MonoBehaviour {
    protected GameObject obj;
    // Use this for initialization
    void Start () {
        GameObject objT = GameObject.Find("Cube");
```

```
        obj =
Instantiate(objT,objT.transform.position+new
Vector3(1,1,1),objT.transform.rotation) as
GameObject;//as 是强制类型转换
    }
    // Update is called once per frame
    void Update () {
        obj.transform.Rotate(10,0,0);
}
```

4.7 Unity中一些比较重要的类

(1)MonoBehavior类

MonoBehavior类是每个脚本的基类，其继承自Behavior类。都继承自Object类。MonoBehavior类中的一些方法可以重写，这些方法会被系统在固定的时间回调。常用的可以重写的方法有很多。

方法	说明
Update	脚本挂载到对象，每帧都会调用一次
FixedUpdate	脚本挂载到对象，会在固定的物理时间调用一次
Awake	脚本实例被载入时该方法被调用
Start	仅在Update方法第一次被调用前调用
OnCollisionEnter	当刚体撞击或碰撞体撞击刚体时该方法被调用
OnEnable	对象变为可用或激活状态时调用
OnDisable	对象变为不可用或非激活状态时该方法被调用
OnDestroy	对象被销毁时调用
OnGUI	渲染和处理GUI事件时被调用

同样。MonoBehavior类中有许多被子类继承的成员变量，这些变量可以在脚本中直接使用。例如之前用的transform变量，其就是继承自MonoBehavior类，不用通过获取Transform组件也能得到对象的transform属性。在Unity5.x之后，很多的成员变量已不再存在，需要通过获取组件才能获取相关属性。具体需要去Mono中查看。不过建议除了transform属性其他都通过获取组件来获取属性。

MonoBehavior类中也包含了很多的成员方法，这些方法在子类也同样可以使用。

成员方法	说明
GetComponent	返回游戏对象上指定名称的组件
GetComponentsInChildren	返回游戏对象及其子对象上指定类型的所有组件
GetComponentInChildren	返回游戏对象及其子对象上指定类型的第一个找到的组件
GetComponents	返回游戏对象上指定名称的全部组件
SendMessage	挂载在游戏对象的每一个脚本中调用指定名称的方法
Instantiate	动态的实例化一个对象
Destroy	销毁一个游戏对象、组件、或资源
DestroyImmediate	立即销毁物体
FindObjectsOfType	返回指定的游戏对象的名称的且被激活的所有物体列表
FindObjectOfType	返回指定的游戏对象的名称的且被激活的第一个物体

(2)Transform类

场景中的每一个物体都有一个”Transform”组件，它就是Transform实例化的对象，用于储存并操控物体的位置、旋转、缩放。每一个Transform可以有一个父级，允许分层次应用位置、旋转、缩放(既参考系不同)。层次关系可以在层级视图中查看到。打开Mono可以进入到Transform中查看到它其中的成员函数和变量。这里需要提一下LookAt可以被用作相机跟随物体的移动，可以用于那种第三人称的游戏。

4.8 输入对象

Unity引擎为开发人员提供了两个输入对象——-Touch 与Input。开发人员通过Touch与Input输入对象中的方法以及参数可以非常方便地获取用户输入的各种参数，包括触摸位置、相位、手指按下后在屏幕的移动距离，Input还能处理键盘的输入等。

变量名	含义	变量名	含义
fingerID	手指索引	position	手指触摸位置
deltaPosition	距离上次改变的距离增量	deltaTime	自上次改变的时间增量
tapCount	点击次数	phase	触摸相位

(1)Touch输入对象

Touch结构体对象中有很多的静态变量，通过这些变量可以轻松获取手指在可触摸设备上的位置等信息，以上是Touch中变量和说明。

关于Touch的游戏只能在真机上才能测试。

(2)Input输入对象

Input相对于Touch而言，Input是Unity中单独的一个类，可以通过Input.touchCount、Input.touches以及Input.GetTouch配合触摸使用，而且Input还可以处理键盘的敲击以及鼠标的点击事件，以下是Input对象的主要变量。

变量名	含义	变量名	含义
mousePosition	当前鼠标的像素坐标	anyKey	当前是否有按键按住，若有则true
anyKeyDown	用户点击任何键盘或鼠标按钮，第一祯返回true	inputString	返回键盘输入的字符串
acceleration	加速度传感器的值	touches	返回当前所有触摸列表

(I.)mousePosition变量

变量mousePosition是一个三维的坐标，用于获取当前鼠标的像素坐标。像素坐标的坐标系是以屏幕左下角为(0,0)点，右上角为(Screen.width,Screen.height)计算的。具体看如下代码，

```
void Update()
    {
        if(Input.GetButtonDown("Fire1")){
            Debug.Log(Input.mousePosition);
        }
}
```

(II.)anyKey变量与anyKeyDown

anyKey用监听当前是否有按键按下，不管什么按键都会触发，若按住键不放则一直触发。

anyKeyDown也能监听当前是否有按键按下，不管不同的，若按住键不放只会在按住的时候触发一次，之后就不会触发，直到松开再次按下便能触发。

(III.)inputString变量

变量inputString返回键盘在这一祯中输入的字符串，在返回的字符串中只包含有ASCII码的字符串。若没有则为一个空串。可用如下代码进行测试，

```
 void Update()
    {

        if(Input.inputString !="")
        {
            Debug.Log(Input.inputString);
        }
}
```

(IV.)acceleration变量

变量acceleration可用获取设备在当前三维空间中的线性加速度，常用于3D游戏的重力感应的游戏。当具有加速度传感器的设备倾斜时，就会回传一个代表设备的倾斜加速度三维向量，使用Input.acceleration变量可用获取该值。

(VI.)touches变量

变量touches可用于获取当前在屏幕上所有触控的引用(Touch[]类型)，可用根据索引可以轻易地获取各个触控点的信息。如下代码所示，

```
void Update()
    {
        int fingerCount = 0;//触摸计数器
        foreach(Touch touch in Input.touches){//遍历触摸的点
```

```
            if(touch.phase != TouchPhase.Ended &&
touch.phase != TouchPhase.Canceled){//当前触摸没结束或者
取消
                fingerCount++; //触摸计数器自加1
}
        }
        if(fingerCount > 0)Debug.LogFormat("User has
{0} fingers touching the Screen");
```

Input除了以上几个常用变量以外，还有很多实用的方法，如下所示，

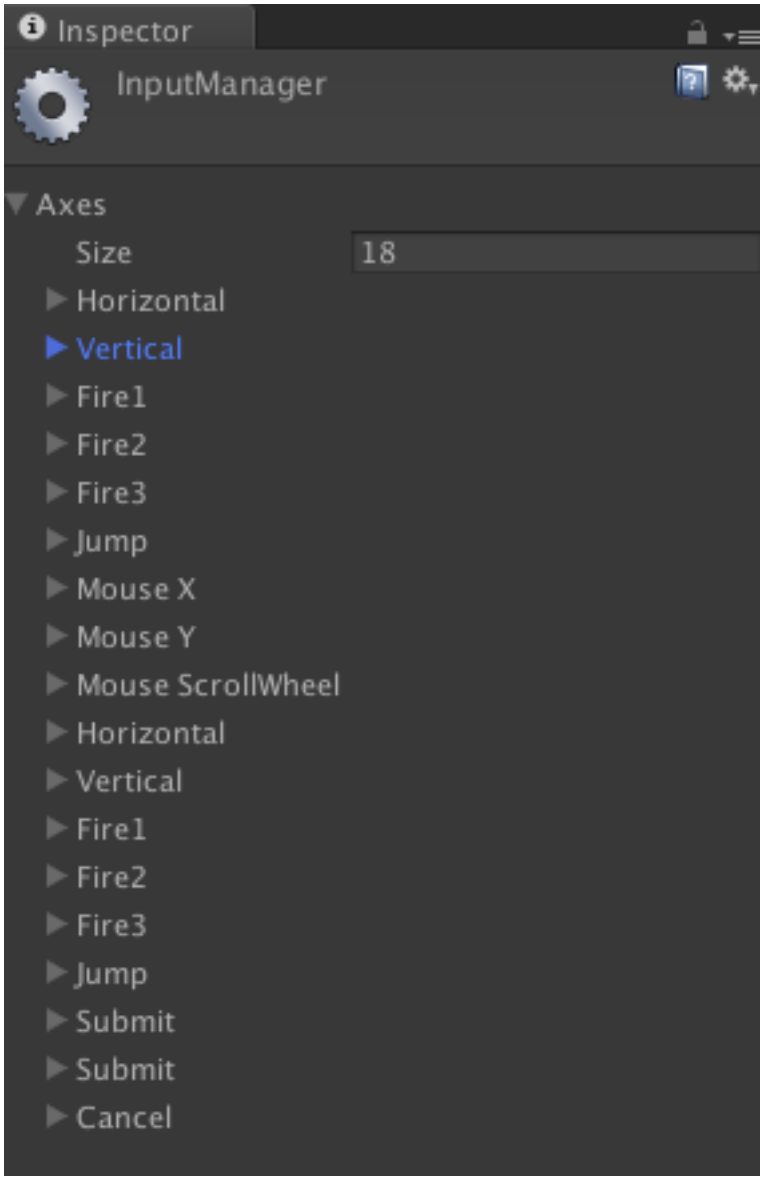
变量名	含义
GetAxis	返回被标识的虚拟轴的值
GetAxisRaw	返回没有经过平滑处理的虚拟轴的值
GetButton	虚拟按钮被按下，则该方法返回true
GetButtonDown	虚拟按钮被按下的一帧，该方法返回true
GetButtonUp	虚拟按钮被按下后被抬起的一帧，该方法返回true
GetKey	按下指定按钮时返回true
GetKeyDown	按下指定按钮的一帧时返回true
GetKeyUp	按下指定按钮然后抬起的一帧时返回true
GetMouseButton	指定的鼠标按键按下时返回true
GetMouseButtonDown	指定的鼠标按键按下时的一帧返回true
GetMouseButtonUp	指定的鼠标按键按下然后抬起时的一帧返回true
GetTouch	返回当前触摸(Touch对象)

(VII.)GetAxis方法和GetAxisRaw方法

GetAxis方法和GetAxisRaw方法都是获取虚拟轴的值得方法。在游戏的开发过程中，经常会在屏幕中添加一些2D的虚拟轴，可以通过触控或者鼠标事件改变虚拟轴的值来控制场景中的游戏对象。如下代码所示。

```
using UnityEngine;
using System.Collections;
public class TouchDemo : MonoBehaviour {
    // Use this for initialization
    void Start () {
    }
    // Update is called once per frame
    void Update () {
        Debug.Log(Input.GetAxis("Horizontal"));//-1~1
        Debug.Log(Input.GetAxis("Vertical"));//-1~1
    }
}
```

如果以上GetAxis换成GetAxisRaw就只有-1、0、1，而上面的值都是在-1~1之间变化。关于Axis可以在Unity中查看，如下图所示，



(VIII.)GetButton方法、GetButtonDown方法与GetButtonUp方法

这3种方法用监听虚拟按钮的按下状态，包括按钮按下时、按钮按下过程、按钮抬起时3个状态。虚拟按钮如上图所示。测试代码如下。

```
using UnityEngine;
using System.Collections;
public class TouchDemo : MonoBehaviour {
    // Use this for initialization
    void Start () {
```

```
    }
    // Update is called once per frame
    void Update () {
        if(Input.GetButton("Fire1")){
            Debug.Log("Fire1");
        }
        if(Input.GetButtonDown("Fire1")){
            Debug.Log("Fire1 Down");
        }
        if(Input.GetButtonUp("Fire1")){
            Debug.Log("Fire1 Up");
        }
}
```

(IX.)GetKey方法、GetKeyDown方法、GetKeyUp方法

这3种方法用于监听键盘上的按键的状态，其他跟上面差不多，代码如下。

```
void Update () {
        if(Input.GetKey("up")){
            Debug.Log("up key");
        }
        if(Input.GetKeyDown(KeyCode.UpArrow)){//可以用键码 键名"up"的键码就是UpArrow
            Debug.Log("up key down");
        }
```

```
        if(Input.GetKeyUp(KeyCode.UpArrow)){
            Debug.Log("up key up ");
        }
}
```

(X.)GetMouseButton方法、GetMouseButtonDown方法、GetMouseButtonUp方法

Input类还提供了监听鼠标点击的事件。其参数为0，1，2，其中0代表鼠标左键，1代表右键，2代表中间滚轮。

(XI.)GetTouch方法

GetTouch用于获取指定索引的Touch对象，因此提供了索引不能大于触摸点个数。为安全起见，一般可以与Input.touches来使用。

4.9 综合案例

见课上。