#所有挂载到游戏对象上的脚本中包含的类都继承自MonoBehaviour类

#初始化脚本的代码必须置于Awake() 或Start() 方法中。不同之处在于 :

Awake() 方法是在加载场景时运行，在所有Start() 方法之前运行

Start() 方法是在第一次调用Update() 或FixedUpdate() 方法之前调用

#\*Unity脚本中协同程序(Coroutines)必须是IEnumerator返回类型，并且yield用yield return替代。如yield return new WaitForSeconds(2); //等待2s

#\*只有序列化的成员变量才能显示在属性查看器中，而private个protected类型的成员变量只能在专家模式中显示，而且，其属性不被序列化或显示在属性查看器，如果属性想在属性查看器中显示，必须是public的

#不要在构造函数中初始化任何变量，而是使用Awake() 和Start() 方法来实现。并且在单一模式下使用构造函数可能会引发类似随机的空引用异常

#Update()方法都是按帧为时间单位计算的

#FixedUpdate()方法是按固定的物理时间被系统回调执行的

#物体旋转是通过transform.Rotate()方法来实现的

#物体移动是通过transform.Translate()方法来实现的

#Time.deltaTime指的是从最近一次调用Update或者FixedUpdate方法到现在的时间，如果想均匀旋转一个物体，不考虑帧速率的情况下，可以使用

#可通过GetComponent<>()来获取一个对象组件

#transform.Find()来找到对象进行操作

#GameObject.FindWithTag().transform.Rotate()来通过tag找到对象 也可以同理操作上面

#FindObjectOfType<>()可以找到第一个有<>的组件 最后添加最先找到

#Instantiate()来初始化游戏对象

#可通过StartCoroutine()方法启动一个协同程序，所有的协同程序都是在主线程中运行的，还是一个单线程程序，StopAllCoroutine()终止所有可以终止的协同程序

#通过私有变量储存组件，使得程序不会在每一帧查找所需组件

#规定左上角坐标为屏幕原点

#绘制按钮需要重写OnGUI()

#普通文字按钮if (GUI.Button (new Rect (), "文字"))

#带有2D纹理的按钮if (GUI.Button (new Rect (), Texture))

#同理还有GUIContent和GUIStyle按钮，GUIStyle多一个按钮文字变量，if为真为按下后的回调执行

#Screen方法里有屏幕长度和宽度

#GUI.DrawTexture()用于绘制纹理，其中参数scaleMode有StretchToFill会对图片进行拉伸，使之沾满整个矩形。ScaleAndCrop将图片等比缩放充满矩形，多余会被裁剪。ScaleTofit将图片等比缩放使图片完全显示在矩形内

static function DrawTexture (position, image, scaleMode, alphaBlend, imageAspect) : void

alpgaBlend : 图片的混合模式，是否通道混合图片显示，默认为混合通道

imageAspect : 源图片的长宽比，如果为0，则使用图像的长宽比，通过宽/高获得所需的长宽比

#GUIStyle控件中的Alignment : 控件中文本的位置，Word Wrap : 勾选后控件中的文字会被限制在控件的矩形区域内，超出部分会换行

#文本输入框static function TextField (position : Rect, text : String, maxLength : int, style : GUIStyle) : String

#密码输入框 PasswordField(position : Rect, password : string, maskChar : char, …)

#同理有TextArea, Toggle控件

#按钮网格static function SelectionGrid (position : Rect, selected : int, texts/image : String[]/Texture[], xCount : int, style : GUIStyle) : int

selected 声名空int，获取返回索引

xCount 在水平方向上有多少元素，控件将缩放到适合宽度

#滑块HorizontalScrollbar (Rect, value, size, leftValue, rightValue, GUIStyle) : float

Size 为滑块的大小

#同理有VerticalScrollbar

#控件分组容器BeginGroup ()和EndGroup ()，夹在二者中间的控件以容器左上角为原点进行定位

#同理有BeginScrollView ()和EndScrollView ()，参数为(position : Rect, scrollPosition : Vector2, viewRect : Rect, alwaysShowHorizontal : bool, alwaysShowVertical : bool, GUIStyle, GUIStyle) : Vector2

scrollPosition用来显示滚动位置

viewRect滚动视图内使用的矩形大小

#Window控件用于创建浮动在普通GUI控件之上的弹出窗口，参数为(id : int, clientRect : Rect, func : WindowFunction, text/image/GUIContent+GUIStyle) : Rect

id设置每个窗口自己的ID

clientRect设置控件在屏幕上的位置

func在窗口创建GUI的函数，该函数必须被传入窗口的ID

#在Assets->Create->GUISkin中添加的skin，在OnGUI()方法中GUI.skin中设置此skin，可影响所有GUIStyle

#在OnGUI()函数内可设置后续控件全局颜色用GUI.color = Color.green;

#在OnGUI()函数内可用GUI.enabled = true/false;来控制是否启用下面的控件

#UI->text可以用obj.setActive([true/false]) 来让其是否显示

#UI->button中有OnClick()事件监听点击

#实现不规则碰撞检测，需重写Image类:

*PolygonCollider2D collider; //声名多边形碰撞器组件*

*void Awake() {*

*collider = GetComponent<PolygonCollider2D>(); //获取2D多边形碰撞器组件*

*}*

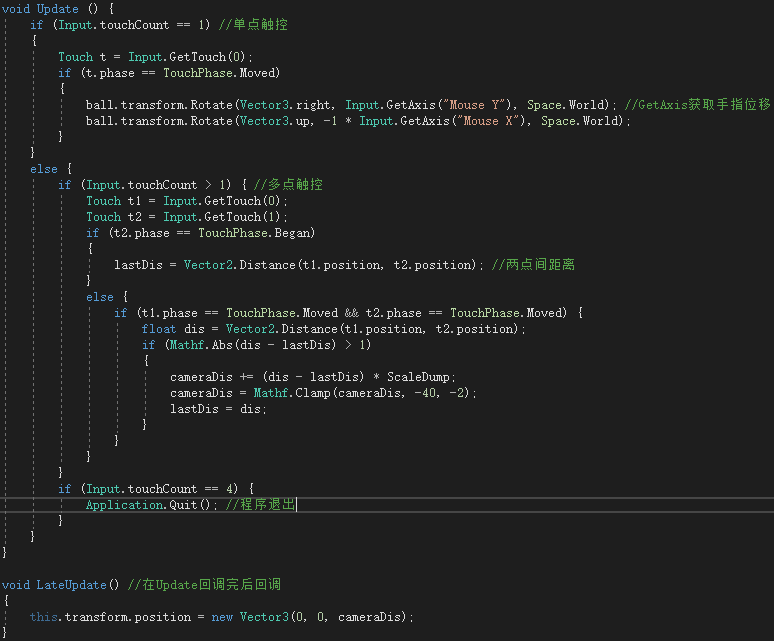
*public override bool IsRaycastLocationValid(Vector2 screenPoint, Camera eventCamera) {*

*bool inside = collider.OverlapPoint(screenPoint); //判断触摸是否在圈出的多边形内*

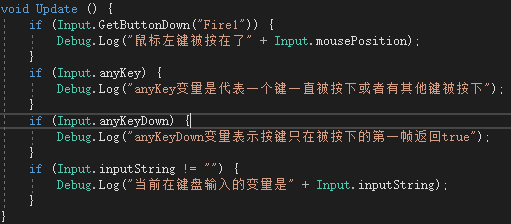
*return inside;*

*}*

#Touch



#Input对象的主要变量



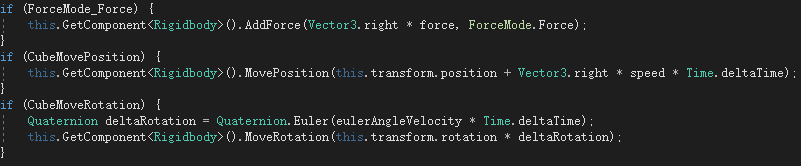
#function AddForce(force : Vector3, mode : ForceMode – ForceMode.Force) : void

Vector3来指定力的方向和大小

ForceMode有四种模式，Force、Impulse、Acceleration、VelocityChange，后两种分别表示对物体施加加速度和改变物体速度，它们都会忽略物体的质量，在不同刚体上使用它们产生的效果相同。

->Force。对物体施加一个持续的力。

->Impulse。对物体施加一个瞬间冲击力。







#利用脚本进行碰撞过滤

*Void Strat* *() {*

*Physics.IgnoreCollision (ballA.GetComponent<Collider>(), ballB.GetComponent<Collider>()); //控制AB两球不发生碰撞*

*}*

#角色控制器相关函数

function SimpleMove (speed : Vector3) : bool

function Move (motion : Vector3) : CollisionFlags

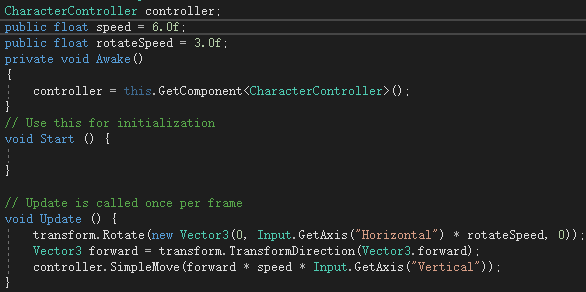
function OnControllerColliderHit (hit : ControllerColliderHit) : void

SimpleMove函数。该函数功能为将物体以一定的速度(speed)移动，并且会返回布尔值来判断物体是否着地。使用该函数物体在y轴上速度被忽略即无法实现物体的跳跃功能，速度以m/s为单位，重力被自动应用。

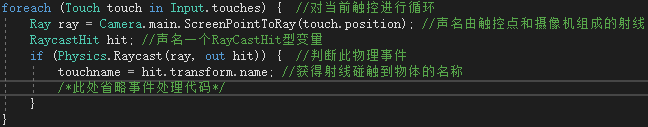
Move函数。这个函数比上个函数更复杂，SimpleMove函数是提供控制器速度来驱动物体，而Move函数是通过提供动力(motion)来驱动物体，使用该函数时，将不会自动应用重力，开发人员需要自行模拟重力，并且会返回角色与其他物体碰撞的信息。

OnControllerColliderHit函数。当角色碰到一个可以执行移动的碰撞器时，这个函数将会被调用。例如需要角色来推开一个带有刚体的物体，就可以将对角色碰到的刚体的控制代码写在这个函数中。

#角色移动



#3D拾取技术。原理: 当玩家用手指单击屏幕时生成一条由屏幕发射到游戏世界的射线，起点就是玩家手指触摸的地方。当射线与游戏世界中的物体发生碰撞之后则会返回被检测到的物体的具体信息，代码入下:



Touch: 用来记录一根手指触摸在屏幕上的状态，变量有position(手指触摸的位置)、taoCount(单击次数)、phase(描述触摸的相位)。片段中通过foreach将手指触摸在屏幕上的信息存储在Touch类型的变量中。

Ray: 表示射线，即一条从起点射出的能够达到无穷远的线。其中包含origin(起点)和director(方向)两个变量，代码的第二行使用ScreenPointToRay函数来创建一条射线，使用该函数时需要传递给它当前手指触摸的位置，这样该函数就会创建一条以手指触摸位置为起点并射向3D世界的射线。

RaycastHit: (光线投射碰撞): 用来获取从Raycast(光线投射)函数返回的信息。其中常用的变量有distance(射线起点到碰撞点达到距离)、collider(碰到的碰撞器)、transform(碰到的变换组件)。代码第五行通过其transform变量来获取射线触碰到的物体的名称。

Physics.Raycast(光线投射): Raycast的重载方法有很多。Raycast函数用来向3D世界投射射线，返回boolean，如果碰到了带有碰撞器的物体就返回true。

###函数

OnTriggerEnter(Collider) : 当Collider(碰撞体)进入trigger(触发器)时调用

OnTriggerExit() : 当Collider停止触发trigger时调用

OnTriggerStay() : 当碰撞体接触触发器时，OnTriggerStay将在每一帧被调用

OnCollisionEnter() : 当此Collider/rigidbody触发另一个rigidbody/collider时，将被调用

OnCollisionExit() : 当此collider/rigidbody停止触发另一个rigidbody/collider时，将被调用

OnCollisionStay() : 当此collider/rigidbody触发另一个rigidbody/collider时，将在每一帧被调用



this与gameObject等价