



介绍

本文档将逐一深入介绍所有脚本，适用于教育工作者或想要了解Unity Playground更多信息的开发人员。

有关Playground的基本信息以及如何开始使用它，请查看入门指南（它是包含在Playground项目中的PDF文件，您也可以查阅[官方教程网站](#)，或在[这里](#)在线阅读）。

目录

| | |
|-------------------------------|-----------|
| 介绍 | 1 |
| 目录 | 2 |
| Movement 运动脚本 | 4 |
| Auto Move 自动移动 | 4 |
| Auto Rotate 自动旋转 | 6 |
| Camera Follow 摄像机跟随 | 7 |
| Follow Target 跟随目标 | 9 |
| Jump 跳 | 10 |
| Move 移动 | 11 |
| Patrol 巡逻 | 13 |
| Push 推 | 15 |
| Rotate 旋转 | 17 |
| Wander 漫步 | 18 |
| Gameplay 游戏机制脚本 | 19 |
| ObjectCreatorArea 对象生成区域 | 19 |
| ObjectShooter 对象发射器 | 20 |
| Projectile ID 子弹ID | 21 |
| PickUpAndHold 拾取和持有 | 22 |
| TimedSelfDestruct 定时自毁 | 23 |
| Attributes 特性 | 24 |
| BulletAttribute 子弹 | 24 |
| CollectableAttribute 可收集物品 | 25 |
| DestroyForPointsAttribut 击毁得分 | 26 |
| HealthSystemAttribute 生命值 | 27 |
| ModifyHealthAttribute 增减生命值 | 28 |
| ResourceAttribute 资源 | 29 |
| 定义资源类型 | 30 |
| Condition 条件 | 31 |
| 游戏行动脚本 | 31 |
| 自定义行动脚本 | 32 |

| | |
|--------------------------|----|
| ConditionArea 条件-区域 | 34 |
| ConditionCollision 条件-碰撞 | 35 |
| ConditionKeyPress 条件-按键 | 36 |
| ConditionRepeat 条件-重复 | 37 |

| | |
|-----------|-----------|
| 行动 | 38 |
|-----------|-----------|

| | |
|--------------------------|----|
| 添加和删除行动 | 38 |
| ConsumeResourceAction 消费 | 39 |
| CreateObjectAction 生成 | 40 |
| DestroyAction 销毁 | 41 |
| DialogueBalloonAction 对话 | 42 |
| LoadLevelAction 加载场景 | 44 |
| OnOffAction 激活和屏蔽 | 45 |
| TeleportAction 瞬间移动 | 46 |

运动脚本

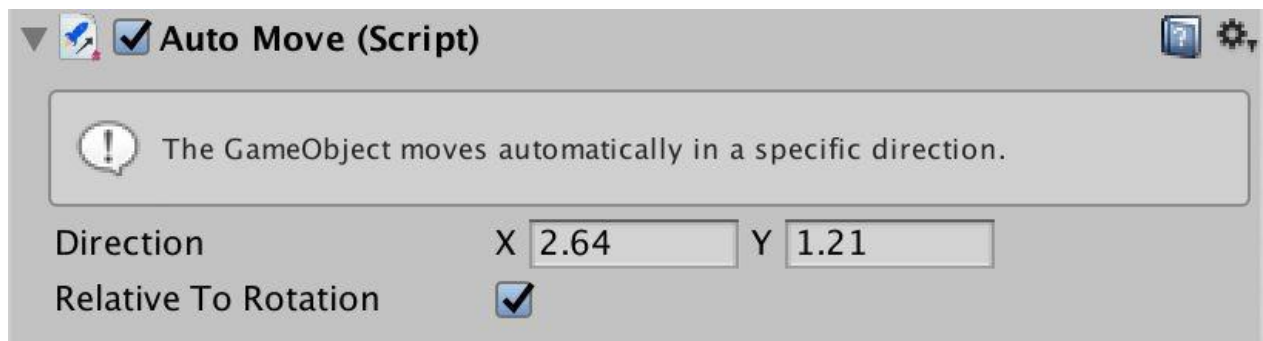
这类脚本都是用来移动游戏对象的，无论是玩家，威胁还是非玩家角色。

Playground的运动是基于物理引擎的，它们几乎都需要Rigidbody2D来产生移动，如果你想让对象能够与其他对象以碰撞的方式交互，还需要添加某种类型的Collider2D。



Auto Move 自动移动

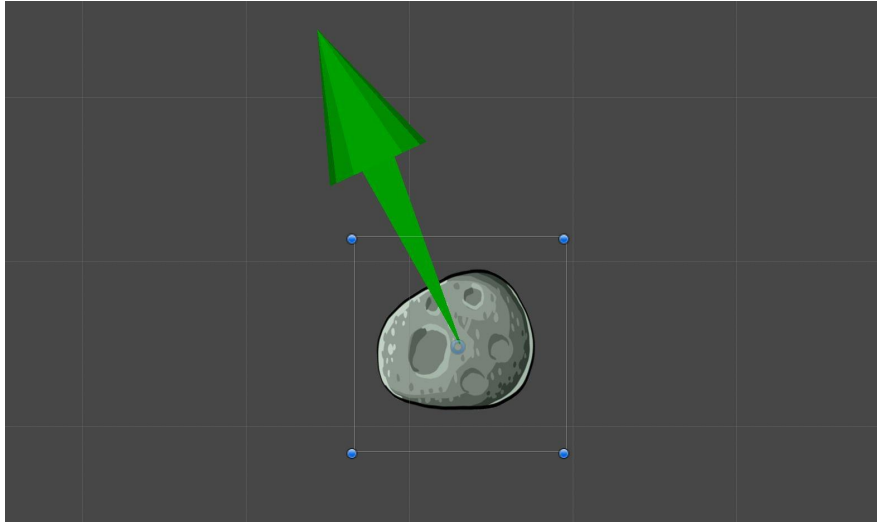
需要组件：Rigidbody2D



AutoMove对GameObject施加连续的力。适用于火箭，弓箭和其他自行式物体。

移动方向通过Vector2表示，向量长度代表强度，可以是绝对坐标或相对坐标。

在“场景”视图中，绿色箭头Gizmo表示推动的方向，而其大小表示强度。

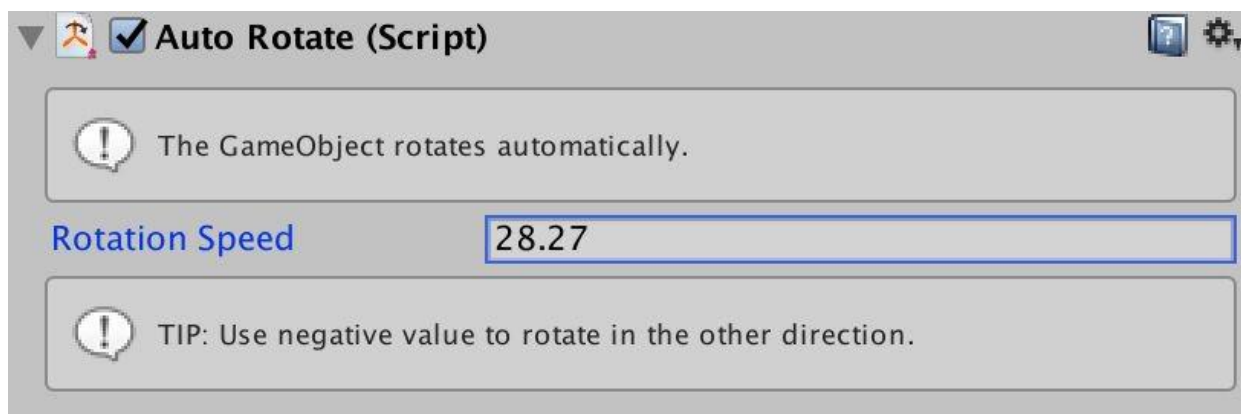


注意：如果您创建的对象是使用Object Shooter脚本创建的，就不一定需要添加自动移动脚本，因为Object Shooter在创建对象时已经给对象施加了力。如果是非自行推进物体（如弹射岩石），则不需要添加自动移动脚本。



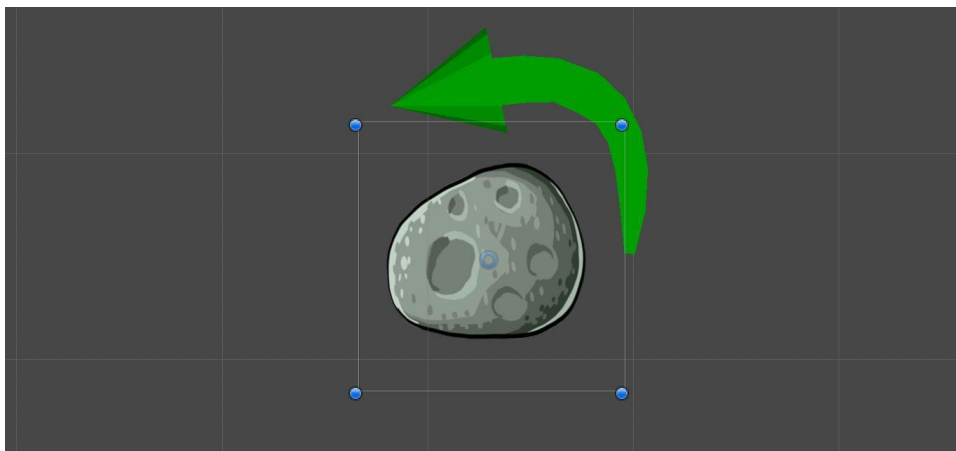
Auto Rotate 自动旋转

需要组件: Rigidbody 2D



AutoRotation使GameObject在Z轴上连续旋转。它可以让装饰性的对象显得更动感，也可以与ConditionCollision脚本配合使用来创建旋转的障碍物。您可以指定旋转速度，设置负数会使对象逆时针旋转。

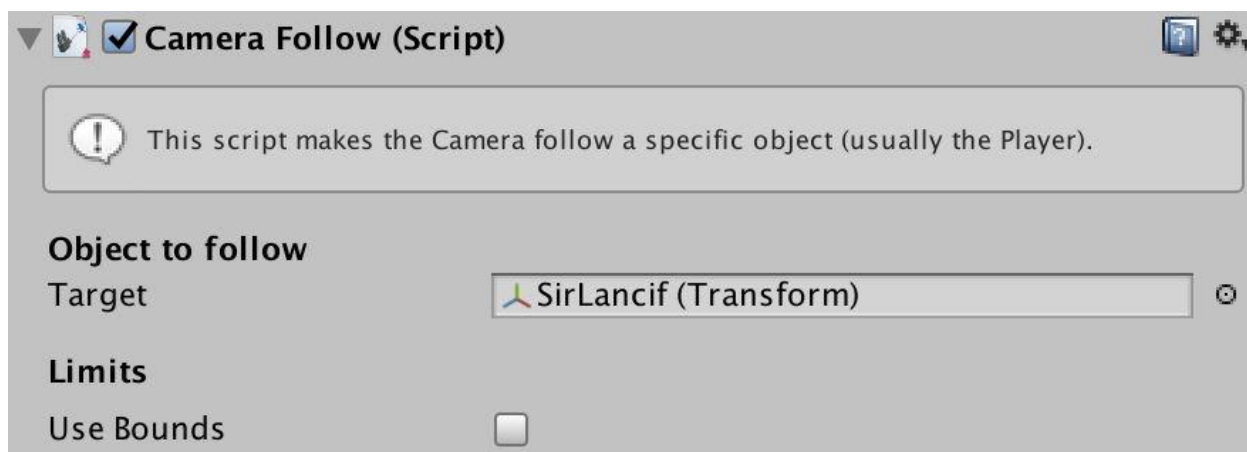
在“场景”视图中，绿色箭头Gizmo表示旋转方向。





Camera Follow 摄像机跟随

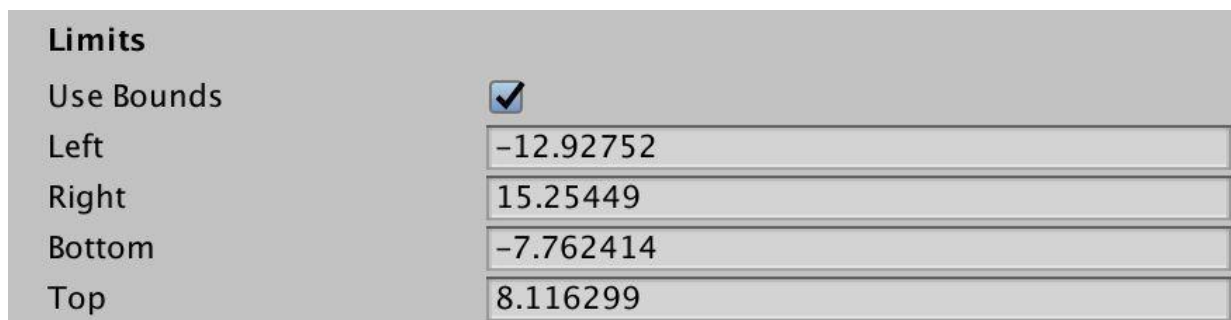
需要组件: Camera



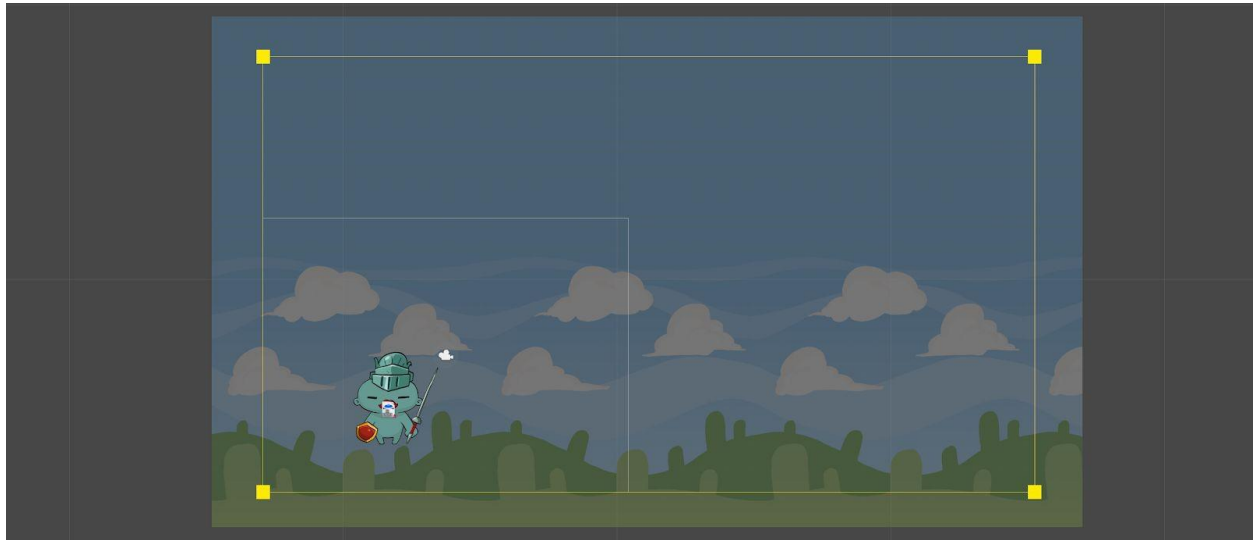
在具有Camera组件的GameObject上使用CameraFollow。这对于需要让玩家位于画面中心的动作冒险游戏非常有用。使用时，把需要摄像机跟随的GameObject 拖动到Target变量上。

注意：这个脚本是添加到摄像机上的，不是添加到被跟随的对象上！

如果勾选属性“Use Bounds”，则可以将摄像机的移动范围限制在一个矩形区域内。



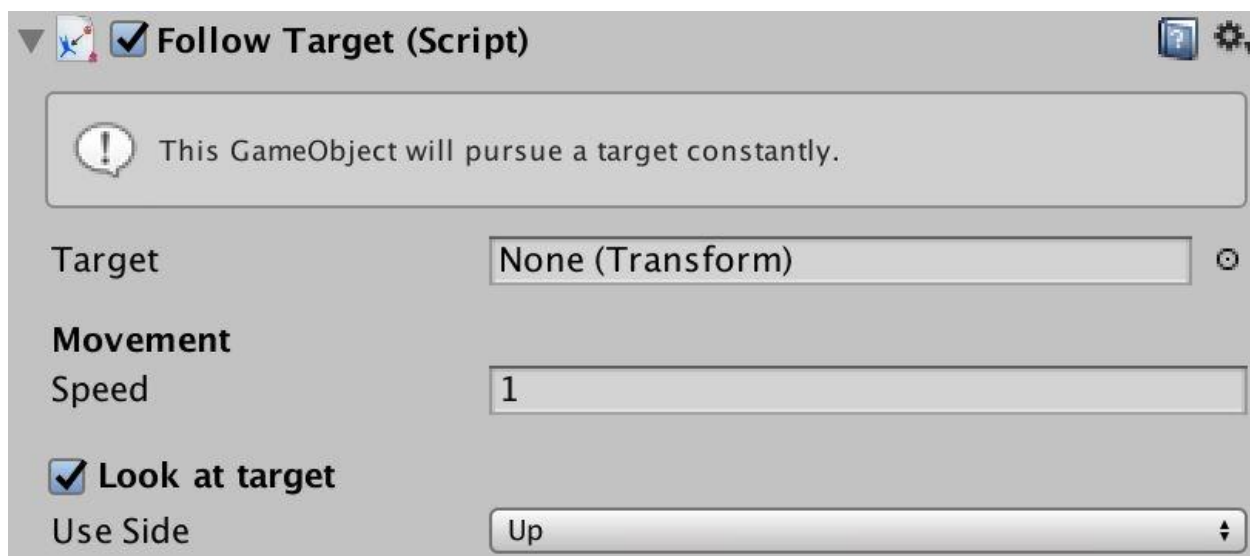
您可以通过Inspector或在场景视图中使用黄色矩形Gizmo调整边界的值：





Follow Target 跟随目标

需要组件：Rigidbody2D



FollowTarget强制GameObject无限期地跟踪指定的目标。

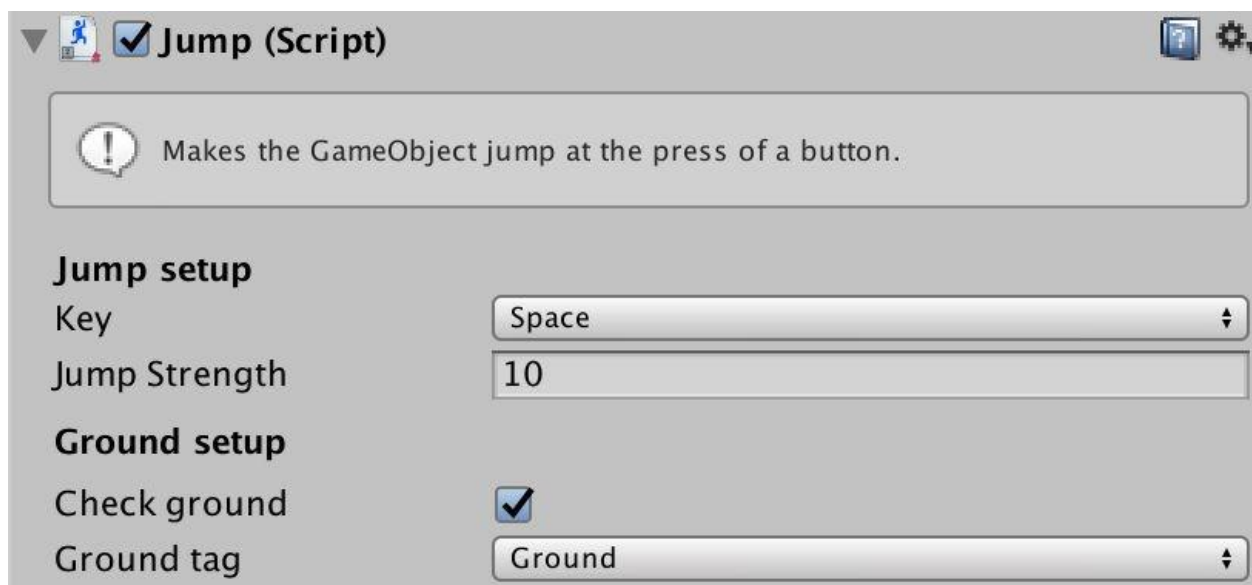
Look at target选项让对象将自己的正面朝向它跟随的目标。有关本选项的更多信息，请参阅本文档的Move脚本部分。

提示：您可以将此脚本分配给敌人并使用玩家作为跟随目标来给玩家持续造成威胁，您也可以让多个角色带上Follow Target脚本，让它们一个跟着一个，排成一队。



Jump 跳

需要组件：Rigidbody2D（以及作为地面的碰撞体！）

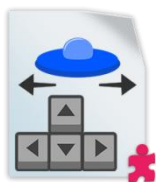


这是一个简单的脚本，当按下特定键时使角色向上跳跃。Key属性用来指定起跳的键盘按键。

要阻止玩家在空中跳跃，请选中“Check Ground”并选择一个标签。然后，您需要使用该标签来标记所有作为地面的游戏对象。一旦GameObject与“地面”发生碰撞，它就能再次跳跃。

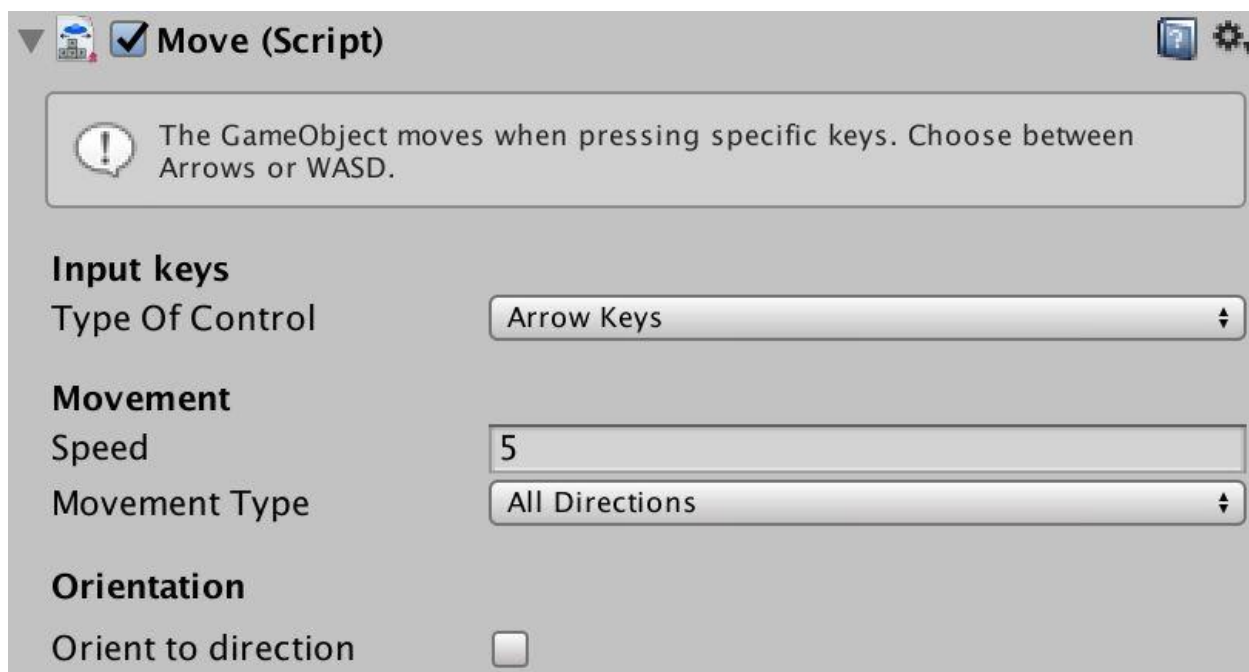
如果未选中“Check Ground”，则角色可以在空中多次跳跃。这可以用来制作像Flappy Bird里面的小鸟跳跃效果。

提示：为了达到想要的跳跃效果，您可能需要将“Jump Strength（跳跃强度）”属性与Rigidbody2D的“Friction（摩擦力）”属性配合调整。



Move 移动

需要组件：Rigidbody2D



这个脚本在两个轴上对GameObject施加恒定力，可以用箭头键或WASD来控制移动。

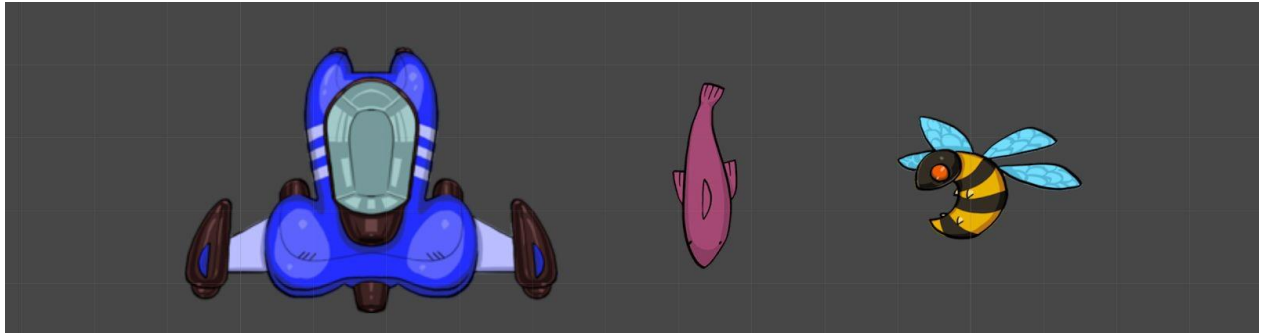
Type of Control属性指定要使用的按键。您可以为场景的两个玩家每人分配一个Move脚本，来创建在一个键盘上玩的多人游戏。

“Movement Type”属性可以让您把移动限制在一个方向轴上。这可以用来实现类似于在固定轨道上的移动，您也可以将其与其他移动脚本组合以创建更精细的移动。例如，您可以把移动限制在水平轴，配合Jump脚本，你就实现了一个能够左右移动并且跳跃的操作方式，这种操作很适合于平台游戏。

注意：请记住，即使您施加的力位于一个轴上，当对象发生碰撞时，它依然可能在另一个轴上移动，这可能会破坏您设想的游戏玩法。如果不希望物体在其他轴上发生移动，请勾选Rigidbody2D上相应轴的“Freeze Position”选项。

Orientation 属性使您可以控制对象是否应该旋转以面向行进方向。这对于载具（宇宙飞船，汽车，船只等）以及俯视角度的Sprite来说非常有用。

如果**Orientation** 被启用，您可以进一步通过设置**Use Side**属性选择使用哪一侧作为使用侧的前进方向。这取决于Sprite是怎么画的。








例如，在上面的图像中，飞船可以把Use Side设为Up，鱼可以把Use Side设置为Down。蜜蜂则不启用**Orientation**，因为它是一张侧面图而不是俯视图，如果让它旋转将看起来很奇怪。



Patrol 巡逻

需要组件：Rigidbody2D

 **Patrol (Script)**  



 The object moves through a series of positions. This can be used for patrolling characters.

Movement

Speed

Stops

| | | | |
|-----|---------------------------------------|---|---------------------------------------|
| = X | <input type="text" value="18.27585"/> | Y | <input type="text" value="13.0009"/> |
| = X | <input type="text" value="24.71309"/> | Y | <input type="text" value="12.50714"/> |

Reset Waypoints

Orientation

Orient to direction ☒

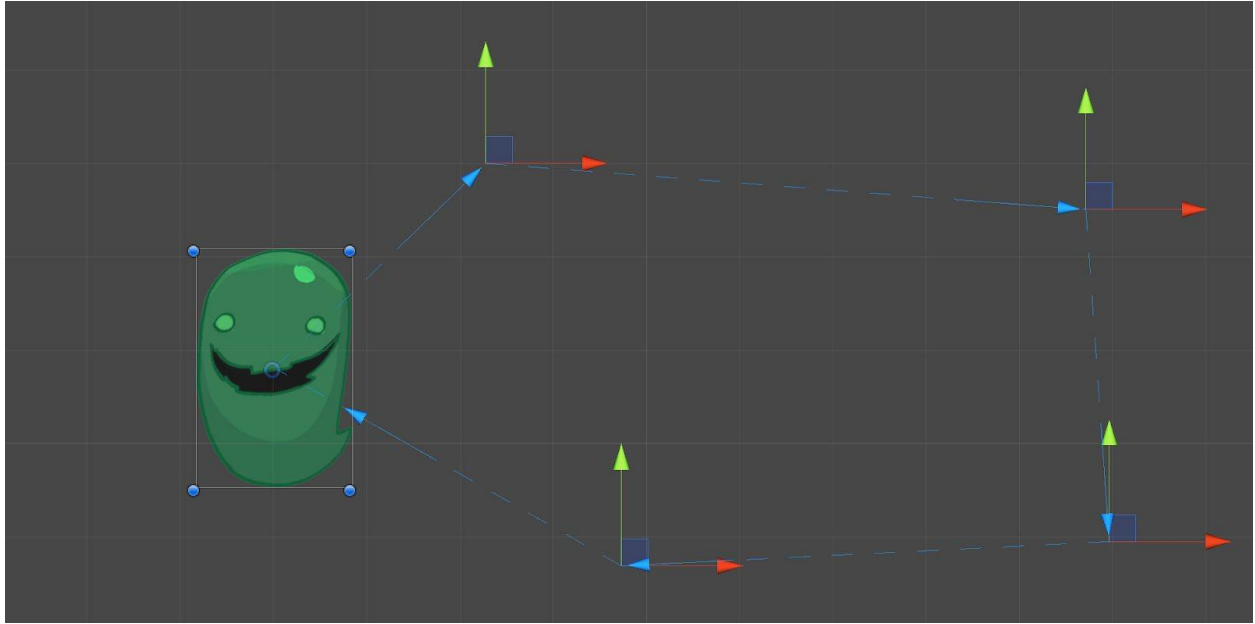
Look Axis

Patrol脚本允许您沿着由导航点构成的路径移动对象。导航点会在Stops 列表中依次列出，您可以轻松添加，删除或重新排序。“Reset Waypoints”按钮清除列表，然后仅添加一个导航点。

一旦所有导航点走完，GameObject就会返回起点，然后在路径上重新启动。

与其他Movement脚本一样，Orientation可让您控制Sprite在移动时的朝向。请参阅Move脚本的说明以了解更多信息。

在创建导航点时，它们在“场景视图”中可视化为“平移”坐标轴。您可以拖动坐标轴或通过Inspector更改位置值来移动它们。

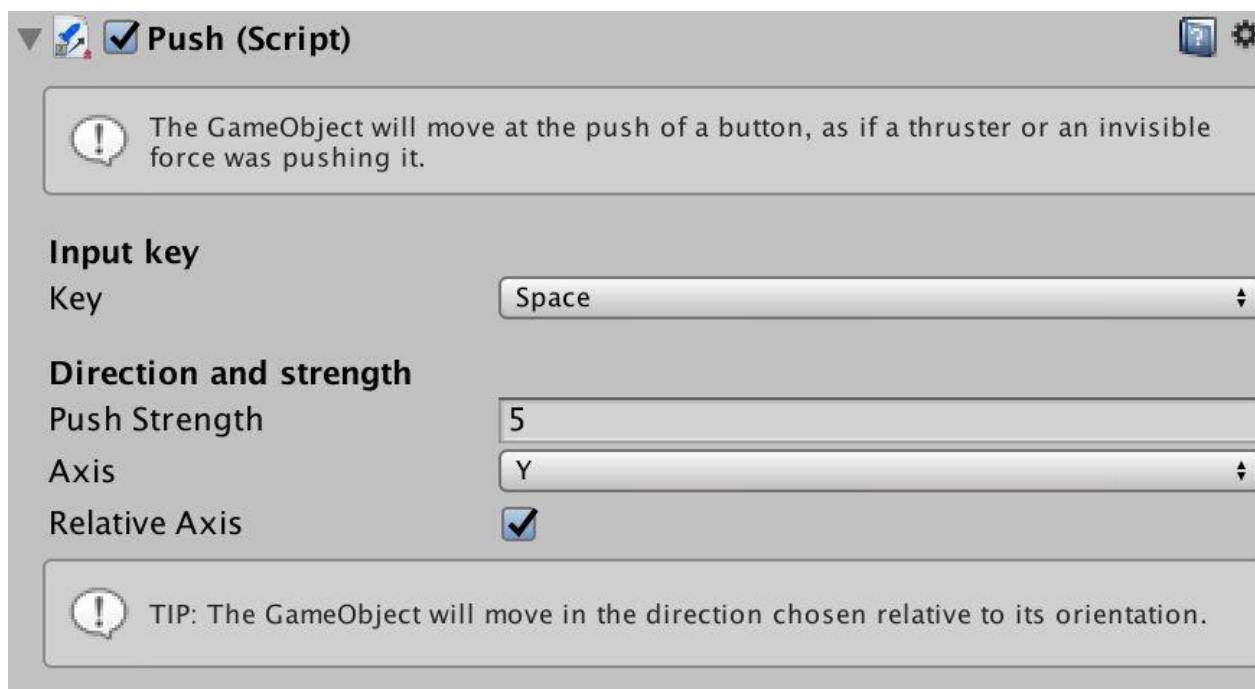


一个小蓝色箭头标志着运动的方向。



Push 推动

需要组件：Rigidbody2D



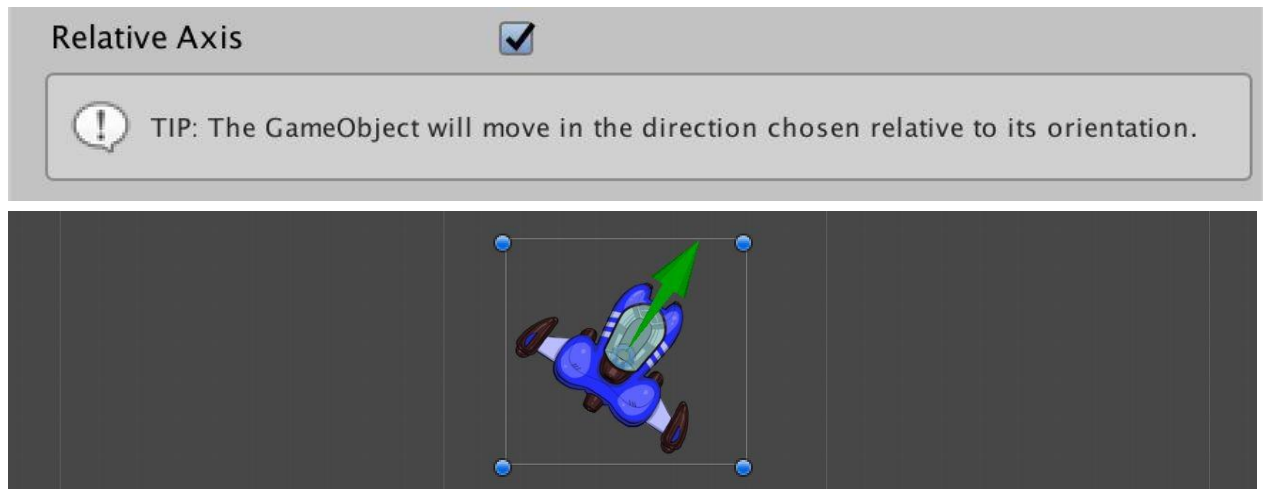
当按住键盘上的特定键时，Push会在一个方向上施加连续的力。它可以用于控制车辆，火箭等，您可以将其与旋转脚本结合使用以实现转向。

在场景视图中，您将看到一个绿色箭头Gizmo，显示力的方向和强度（见下文）。

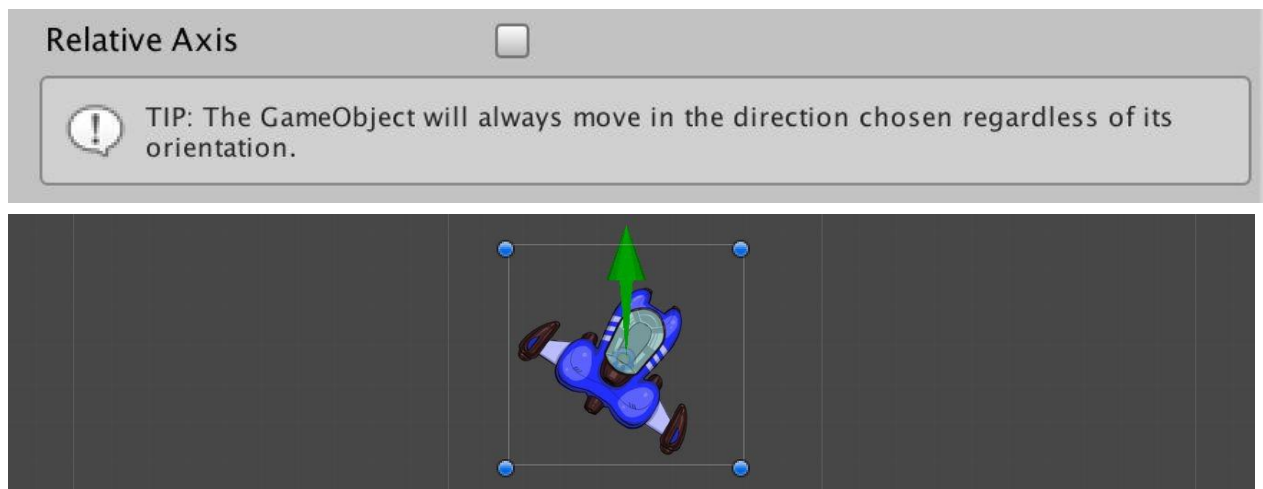
Axis属性用于控制推动的方向（Y表示向上，X表示向右）。要实现相反方向的推动，只需将“推力”设置为负值即可。

Relative Axis属性决定了推力方向是相对方向还是绝对方向。如果对象的旋转角度为0，从Gizmo上将看不到两者的区别。要了解其中的差异，请看下面的图像。

在**Relative Axis**打开的情况下，旋转对象意味着方向随之旋转（在局部空间中）：



当**Relative Axis**关闭时，方向是绝对的（在世界空间中）：

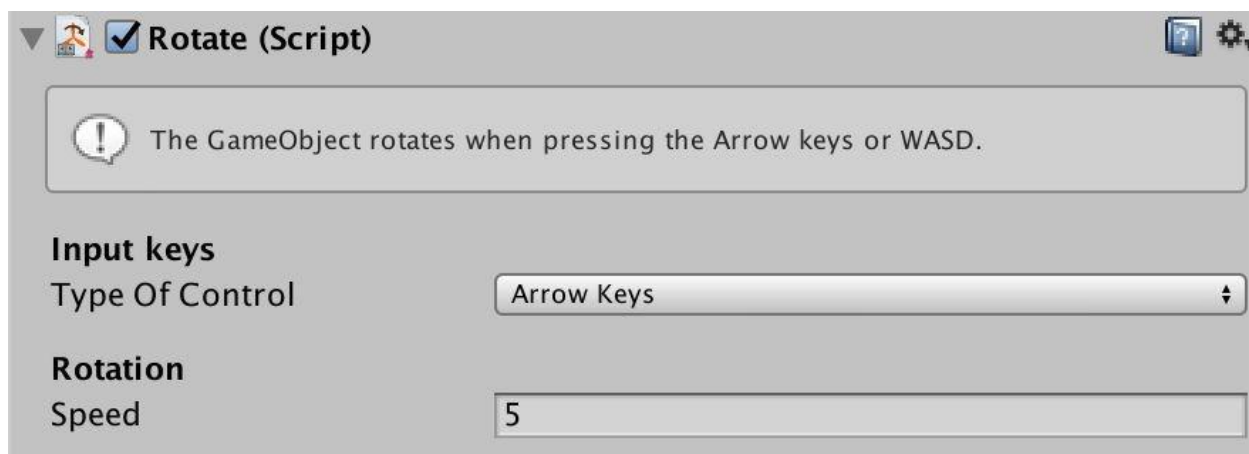


更改选项时，选项下方的工具提示会随之改变。多数时候**Relative Axis**应该被打开，以便载具向它的前方移动。



Rotate 旋转

需要组件：Rigidbody2D



旋转是一个施加扭矩的脚本 - 让物体在Z轴上的旋转。与Move类似，它由左/右箭头或AD键控制。您可以将它与Push脚本一起使用，以创建适用于载具的操作方式，控制载具的转向，以及让它朝自己的正前方移动。

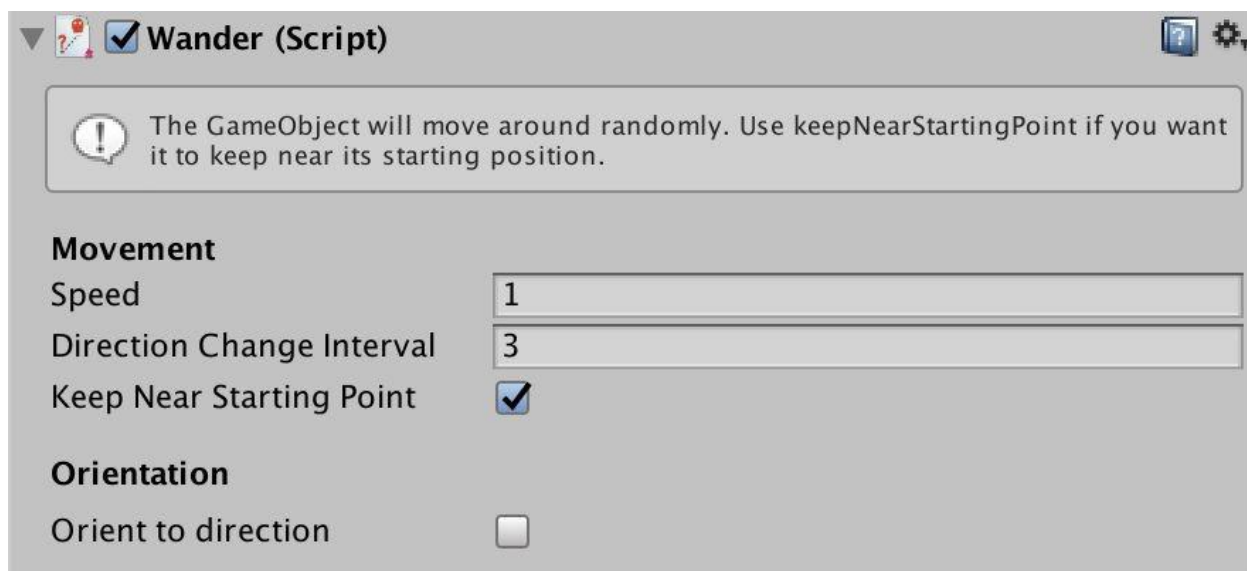
提示：如果您想改变对象的旋转中心位置，可以让这个对象成为另一个GameObject的子对象，调整它和父对象的相对位置，将Rotate脚本应用于它的父对象。例如，想一下自行车，其中旋转中心位于后轮上。

通过这种方式，您可以更好地控制中心（Gizmo出现的位置），而无需更改Sprite。



Wander 游荡

需要组件：Rigidbody2D



Wander游荡组件让GameObject不断进行随机移动。**Speed**属性控制移动速度，通常需要与Rigidbody2D中的**Friction**属性配合调整，才能达到理想效果。

Direction Change Interval属性用于设置对象改变移动方向的时间间隔（以秒为单位）。设置一个非常低的数值会使对象作出很多短暂的突然动作。

Keep Near Starting Point意味着GameObject始终在初始位置附近移动，不会走得太远。

注意：如果您将**Speed**属性设置得太高或Rigidbody2D上没有足够大的**Friction**，则物体可能仍然可以在很远的地方徘徊！

与其他Movement脚本一样，**Orientation**属性可让您控制在移动时如何定位Sprite。详细请参阅移动脚本中的说明。

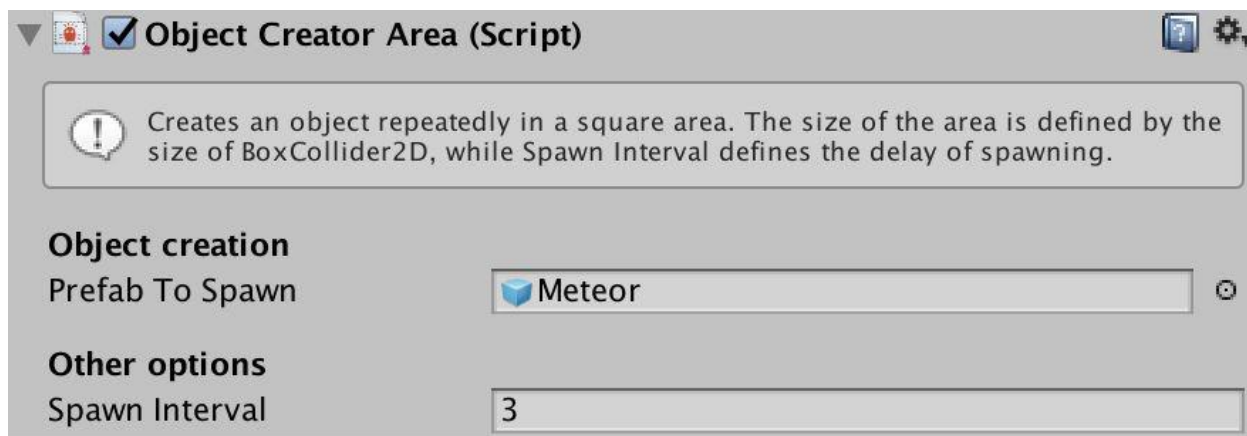
Gameplay 玩法机制脚本

Gameplay脚本是产生游戏效果的各种脚本。它们独立工作，不需要条件来激活它们。



ObjectCreatorArea 对象创建区域

需要组件： BoxCollider2D



ObjectCreatorArea脚本在矩形区域中生成新对象。要使用它需要BoxCollider2D（会被自动添加）来定义实例出现的区域。




Prefab to Spawn属性可以接受场景中的对象，但最好是给它一个预制件。如果从场景中分配普通GameObject，则组件会显示警告。


Spawn Interval属性确定生成对象的时间间隔，以秒为单位表示。




ObjectShooter 对象发射器

需要组件：没有

 **Object Shooter (Script)**  

 Spawns an object at the press of a button and it applies a force to it in the direction chosen.

Object creation
Prefab To Spawn 
Key To Press

Other options
Creation Rate
Shoot Speed
Shoot Direction X Y
Relative To Rotation ☒

ObjectShooter用于在按键时发射物体（默认按键是空格键）。你可以用它来制造发射子弹或激光的武器，或者网球发球机，或任何一种可以反复发射某些东西的物品。它可以与另一个脚本BulletAttribute一起使用（见下文）。

此脚本一般会被分配给空对象，让这个空对象作为一个角色的子对象，用作子弹生成点。通过调整这个空对象的位置，您可以精确控制子弹的射出位置。

Prefab to Spawn属性可以接受场景中的对象，但最好是给它一个预制件。如果给它的不是预制件，而是场景中的GameObject，则组件会显示警告。

Creation Rate属性控制发射的时间间隔，以秒为单位。**Shot Speed**决定速度，**Shot Direction**是指示发射方向的Vector2二维向量。

Relative to Rotation用于指定发射方向是相对方向还是绝对方向，如果您控制一艘能够旋转的太空船，并希望子弹朝飞船的正前方发射，那您应该打开这个选项。关闭这个选项将代表射击方向是在世界空间定义的绝对方向。

应用此脚本时，将在“场景视图”中显示绿色箭头Gizmo。



箭头的大小与发射的强度无关。

注意：您也可以将发射速度设置为零。这样玩家可以在移动时在原地留下物品。不要被“射击”的脚本名称所迷惑：生成的物体不一定需要被发射出去！

子弹ID

ObjectShooter还能够为发射出去的子弹分配玩家ID。为了让子弹带上ID，您需要在被发射的子弹预制件上添加BulletAttribute脚本。如果带有玩家ID的子弹击中了一个带有击毁得分脚本（DestroyForPointsAttribute）的对象，发射该子弹的玩家就能得分。

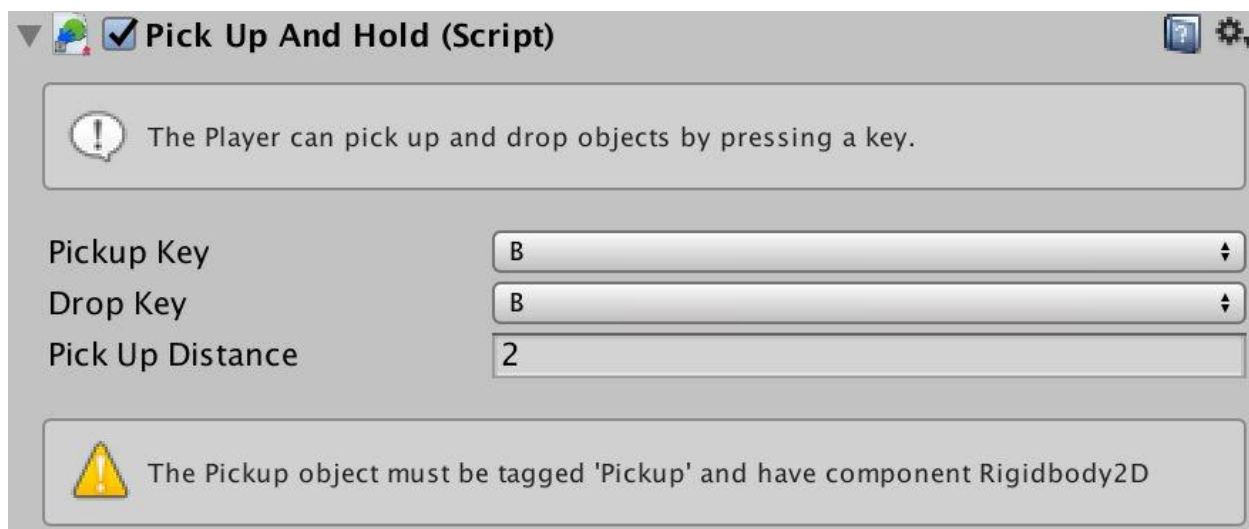
为了确保ObjectShooter分配正确的ID，您需要将玩家对应的GameObject的标签设为Player或Player2。

注意：如果您没有给GameObject设置玩家标签，则子弹被视为来自Player1，因此对于单人游戏，您无需担心标签问题。



PickUpAndHold 拾取和持有

需要组件：没有



此脚本用于为角色提供拾取（和丢弃）某些物体的功能，例如道具或者体育游戏中的球。再配合使用ConditionArea脚本，您可以创建夺旗回到基地得分的游戏玩法。

要使对象可以被拾取，您需要为其分配Pickup标签并为其添加任何类型的Collider2D。将碰撞体设置成触发器会使它比较容易被拾取。如果该对象具有Rigidbody2D，则它在被拾取后将成为拾取它的角色的子对象，Rigidbody2D组件的Is Kinematic选项也会被打开。

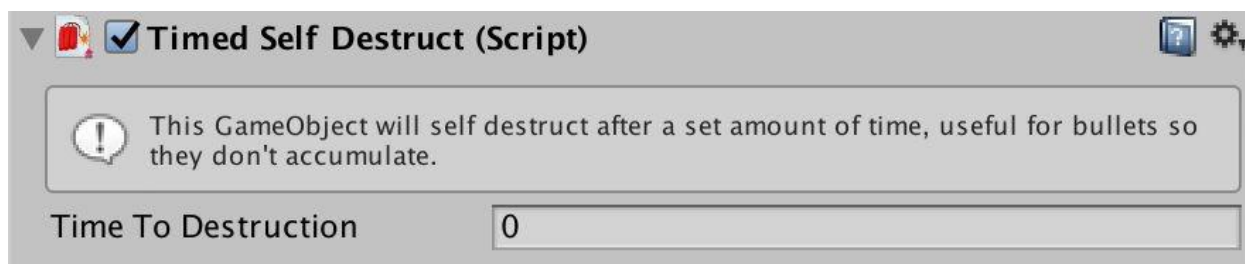
Pickup Key和**Drop Key**定义需要按哪个键来拾取和放下该物品。默认情况下拾取和放下都是同一个按键，但您也可以把它们设成不同的按键。

Pick Up Distance属性定义了玩家能拾取这个物体的最大距离。如果存在多个拾取对象，则拾取最近的拾取对象。



TimedSelfDestruct 定时自毁

需要组件：没有



TimedSelfDestruct用于删除一段时间后场景中不再需要的对象。**Time to Destruction**是指物体经过多少秒后会消失。

您应该在子弹和任何不需要一直存在于场景中的物体上使用它，以确保场景不会混乱。此外，如果您的游戏生成了大量对象，那么它可能会变慢。将TimedSelfDestruct用在这些对象上 - 哪怕销毁的时间间隔设置得很长 - 也会有助于游戏更流畅地运行。

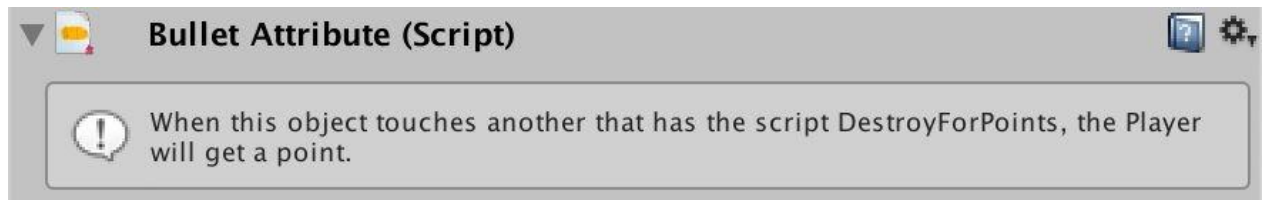
Attributes 特性

Attribute特性脚本不会自己执行操作：它们只是用于定义对象具有的数值参数，然后其他一些脚本将根据游戏对象带有的Attribute脚本来进行操作。它们的作用类似于Tags，但作为脚本，它们可以带有额外的数据。



BulletAttribute 子弹特性

需要组件：任何形状的Collider2D



BulletAttribute脚本没有自己的功能，但它包含了发射子弹的玩家的ID。这个ID（代码中的变量名是playerID）可以是0（玩家1）或1（玩家2）。

ObjectShooter脚本会在发射子弹的时候自动设置子弹的ID。

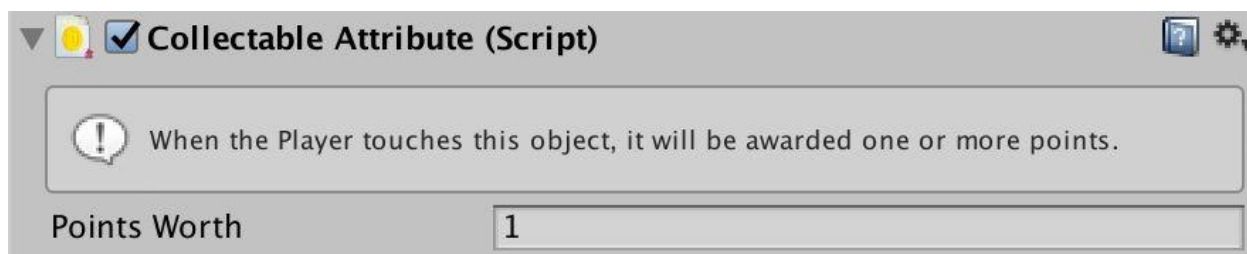
注意：如果您需要在Inspector中设置playerID属性，请打开BulletAttribute脚本并从属性中删除属性[HideInInspector]。

如果您的子弹不是用ObjectShooter脚本生成的，就可能需要这样自己设置。



CollectableAttribute 可收集特性

需要组件：任何形状的Collider2D



CollectableAttribute奖励任何接触该物体的玩家。因此，它需要Collider2D，并且需要将碰撞体标记为触发器以避免发生实体碰撞。

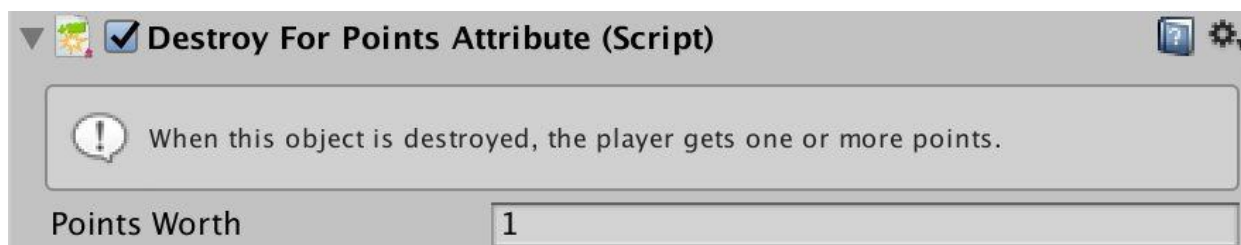
Points Worth属性设定吃到这个物体的得分，可以为每个对象分配不同的分数。

注意：要在屏幕上查看总分，需要在场景中显示UI预制件。有关UI预制件的更多信息，请参阅“入门”手册。



DestroyForPointAttribute 属性

需要组件：没有



DestroyForPointsAttribute适用于射击游戏中的目标和敌人。它会在与另一个对象发生碰撞时销毁GameObject，前提是它具有BulletAttribute脚本。此外，它还奖励最初射击子弹的玩家。

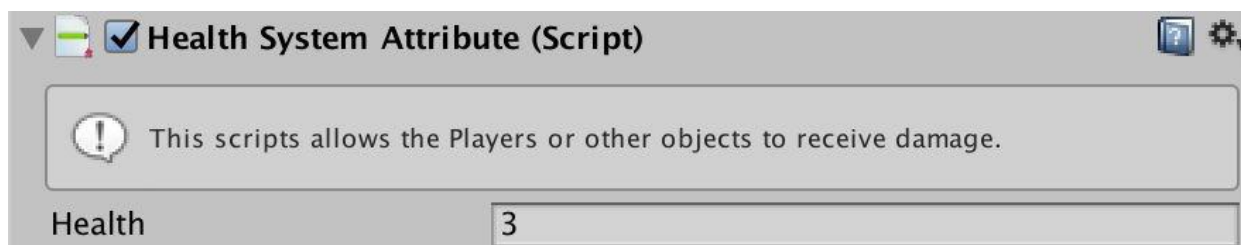
有关如何将ID分配给项目符号的详细信息，请参阅BulletAttribute和ObjectShooter脚本。

注意：要在屏幕上查看总分，需要在场景中显示UI预制件。有关UI预制件的更多信息，请参阅“入门”手册。



HealthSystemAttribute 生命值特性

需要组件：没有



HealthSystemAttribute可以添加到角色，敌人或对象中。它允许他们受到伤害，如果一个对象的生命值变为0，它通常会被从游戏中移除 - 玩家生命值为0时游戏就结束了。

这个脚本与ModifyHealthAttribute脚本结合使用。ModifyHealthAttribute脚本负责增减生命值，通常会被放在子弹对象上。具有ModifyHealthAttribute的对象不会影响没有HealthSystemAttribute的对象。

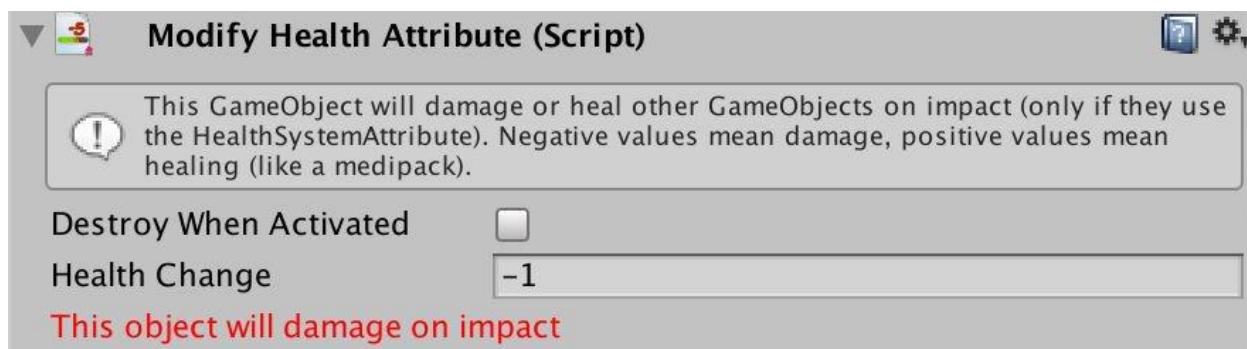
HealthSystemAttribute不要求游戏对象上必须存在Collider2D组件，但如果您依靠碰撞检测来对您的玩家扣血，建议您在玩家身上添加一个碰撞体。

注意：要在屏幕上查看玩家的生命值，需要在场景中显示UI预制件。有关UI预制件的更多信息，请参阅“入门”手册。



ModifyHealthAttribute 增减生命值

需要组件：任何形状的Collider2D



此脚本可以使具有HealthSystemAttribute的任何对象的生命值增加或减少。它可以用来制作子弹，危险区域等，但也适用于回血物品，如药品，食品等。

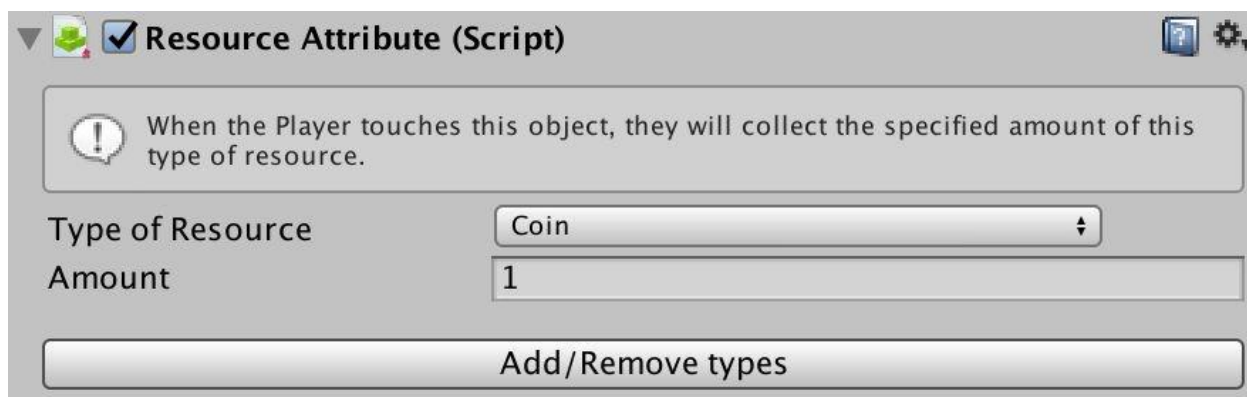
Destroy When Activated是一个属性，在选中时，对象会在第一次产生效果时被删除。这个选项可以用于子弹，消耗品和任何只能用一次的东西。

Health Change属性指示此对象会造成生命值增减。如果它是负数，那就是减血。如果是正数，那就是回血。如果是减血，检视面板中的最后一行会显示红色，否则显示蓝色。



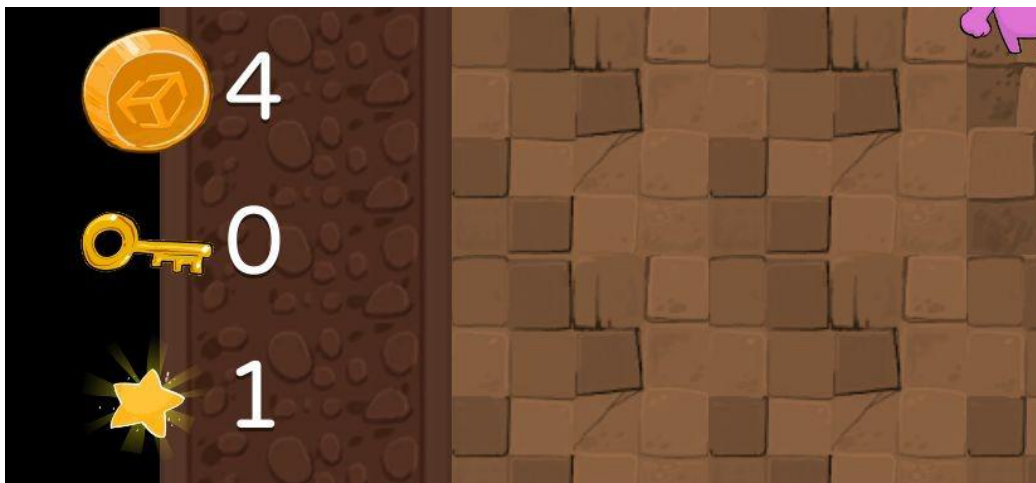
ResourceAttribute 资源特性

需要组件：任何形状的Collider2D和SpriteRenderer



ResourceAttributes与ConsumeResourceAction配合使用，就可以实现资源和道具背包的功能。当玩家触碰到具有此脚本的对象时会将其拾取，并将其添加到UI显示的清单中。

该脚本需要一个SpriteRenderer，Sprite用于在左下角显示一个图标，以及一个表示该玩家拥有的资源量的数字：



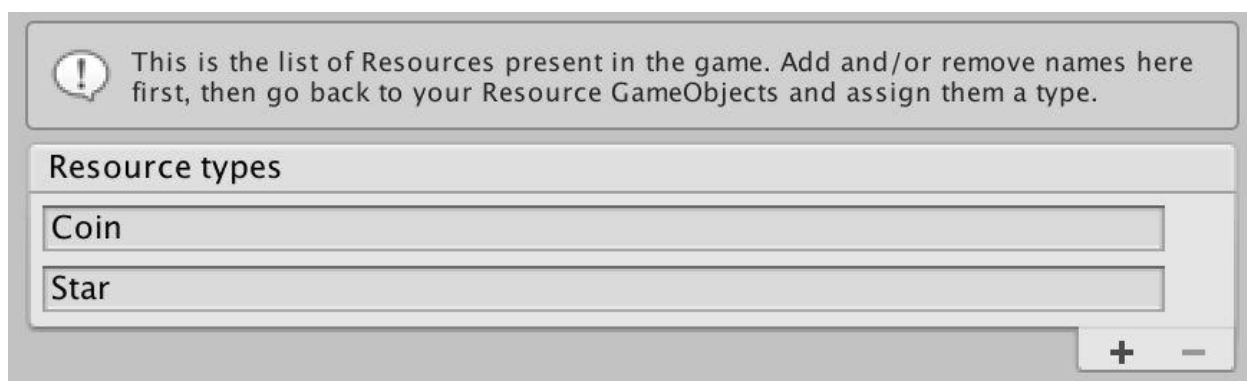
获取更多相同类型的资源会增加数量。然后，ConsumeResourceAction脚本可以请求和使用它们。有关详细信息，请参阅其说明。

使用资源，您可以创建一个物品合成系统（收集资源，然后使用它们来“支付”并获得一个新物品作为回报）或者一个门/钥匙系统，每个门需要正确的钥匙才能打开。

您可以使用“Resource of Type”属性定义此资源所属的类型（请参见下文），也可以设定这个物体的资源数量（可用于指定拾取的金钱数量）。

定义资源类型

可以通过单击“Add/Remove types”按钮来定义资源类型。单击按钮后编辑器将打开一个名为“InventoryResources”的ScriptableObject上。在此对象上，有一个定义了游戏中所有可用的资源类型的字符串列表。



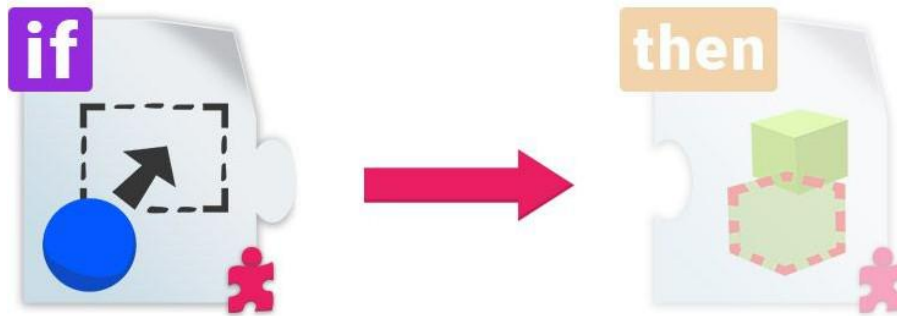
在示例项目中已添加了几个资源类型，但您可以添加和删除更多以创建更多自定义游戏。添加资源类型后，请务必返回具有ResourceAttribute脚本的对象并进行分配。

注意：资源类型在所有场景之间共享，因此如果删除已经定义好的基本类型（Coin，Star等），一些示例场景将无法正确运行（例如Roguelike）。

把“InventoryResources”对象移动到项目的其他位置也会导致出错。

Conditions 条件

条件与编程中的If语句非常相似，这意味着它们充当了其他行为的入口。如果条件成立，则执行该条件附带的Action行动脚本。请参阅本文档的行动脚本部分了解有哪些行动脚本可以使用。条件脚本在其图标的左上角有一个紫色的“if”标记。



所有条件脚本都具有一些共通的属性，如下所述。

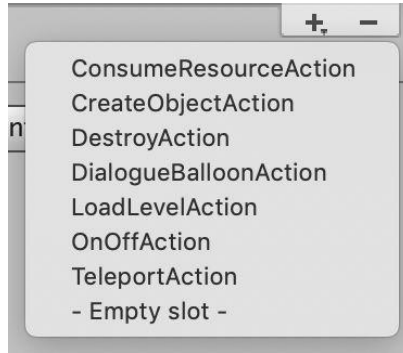
Gameplay Actions 行动列表

Gameplay Actions是一个Action脚本列表，它列出了在此Condition脚本的条件成立时执行的所有Action。一旦条件成立，Unity将执行此列表中的所有Action，直到其中一个失败（只有少数Action会出现执行失败的情况）。如果Action没有失败，则继续执行下一个。Gameplay Actions执行完毕后，将执行**Custom Actions**指定的自定义操作。

一个空白的Action列表如下所示：



按加号图标会显示可添加的Action行动列表：



它们与Action类别中的脚本完全对应。如果选择其中任何一个，脚本将作为组件添加并自动关联到列表。

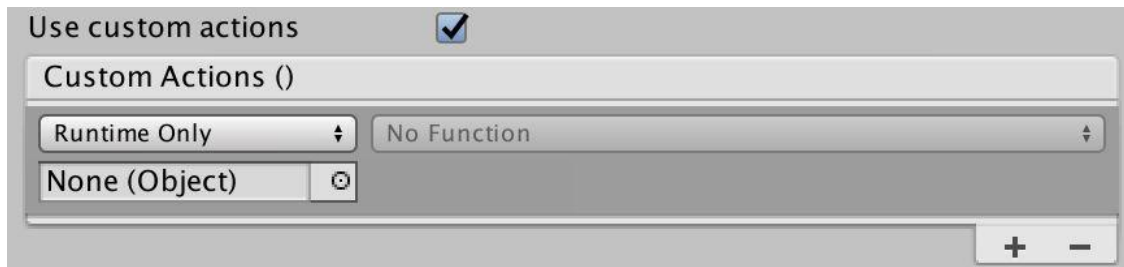
注意：只有在Gameplay Actions列表中被关联的Action行动脚本才会被执行。被添加到GameObject 的Action脚本不会自动执行。

提示：在下拉列表的末尾，您还可以选择添加空插槽，用于连接另一个GameObject上的Action。通过这种方式，您可以将逻辑划分为多个部分，从而使其更易于管理。
一个实际的例子，就是Roguelike示例游戏的InvisibleTrigger对象。可以发现它上面有2个Action是位于它的两个独立的子对象上的。

Custom Actions 自定义操作

与Gameplay Actions类似，自定义操作会在Condition脚本的条件成立时被执行。**Use Custom Actions**属性决定是否执行自定义操作，如果未选中，则不会执行自定义操作。

选中该选项会显示UnityEvent 事件列表，您可以在此处连接需要调用的UnityEvent。



对自定义操作的支持使您能够对游戏性做极大扩展：如果您了解编程，则可以编写一个简单的脚本并暴露一个公共函数，然后将它连接到自定义操作。这样，您可以通过Playground的自带Condition脚本来调用您自己的脚本。这样教师可以在上课期间快速扩展Playground功能。

另一个用途是在这里连接标准的Playground公共函数。UI脚本及其GameWon和GameOver函数就是一个很好的例子。通过将这两个函数连接到条件脚本的自定义操作列表中，您可以创建自定义的胜利条件，例如在对象碰到其他物体，或进入某个区域时获胜。



ConditionArea 条件-区域

需要组件： 设置为Trigger的Collider2D



ConditionArea需要将Collider2D设置为触发器。**Event Type**事件类型属性确定事件在何种情况发生：进入区域，离开区域，还是留在内部（如果选择留在内部，将显示频率参数以确定事件发生的频率）。

您可能希望只有在特定类别的对象进入该区域时才触发事件，Filter by Tag可以限制能够触发事件的对象标签。常用的标签是Player，但您可以使用任何其他标签。

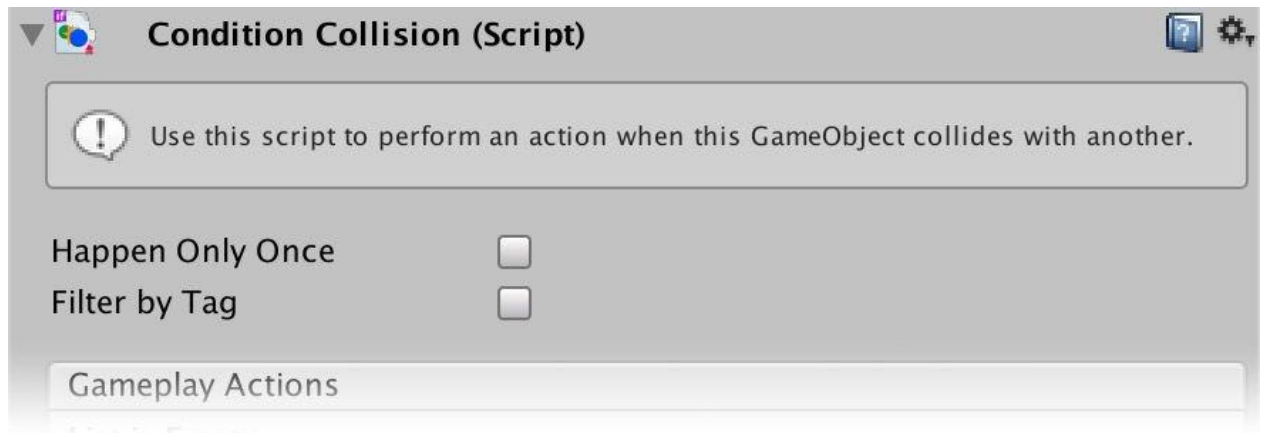
Happen Only Once让事件只能被触发一次。

Gameplay Actions和**Custom Actions**在所有条件脚本上的用法都是一样的，详细说明请参阅上文。



ConditionCollision 条件-冲突

需要组件：没有



ConditionCollision是一个简单的条件，当一个对象与具有此脚本的对象发生碰撞时，可以触发事件。与其他条件一样，您可以通过在“**Filter by Tag**”中设置标签来筛选能够触发事件的对象类型。

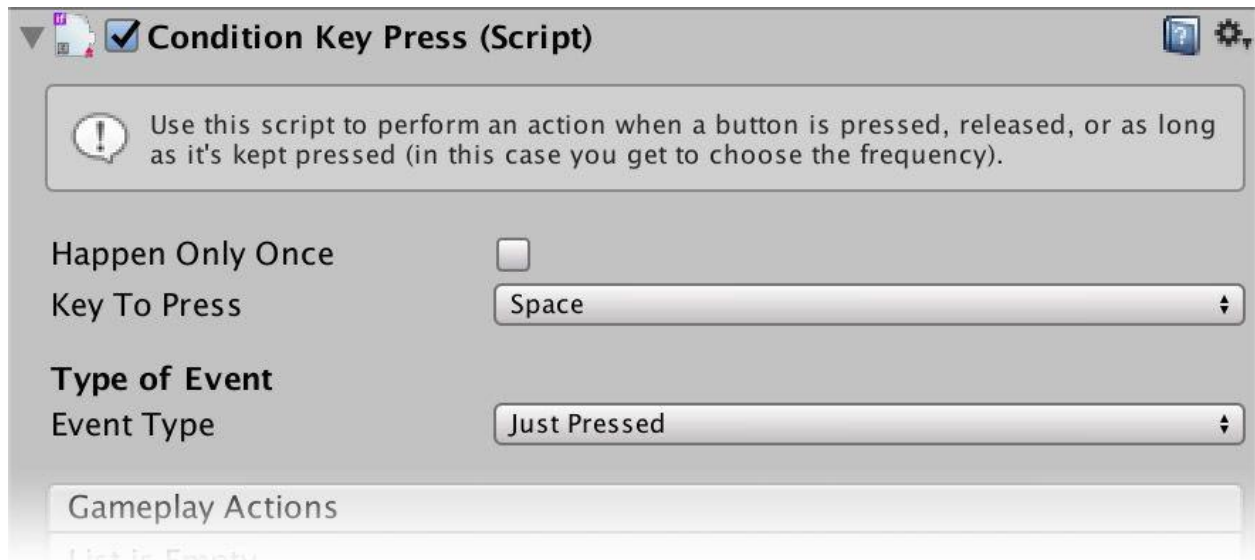
Happen Only Once让事件只能被触发一次。

Gameplay Actions和**Custom Actions**在所有条件脚本上的用法都是一样的，详细说明请参阅上文。



ConditionKeyPress 条件-按键

需求组件：没有



ConditionKeyPress是将Action绑定到按键的通用方法。**Key to Press**可以选择要绑定的按键，**Event Type**可以选择按键事件的类型：Just Pressed（类似于GetKeyDown），Released（GetKeyUp）或Kept Pressed（GetKey）。与其他连续动作一样，Kept Pressed模式具有Frequency频率属性来定义两次事件之间的时间间隔。

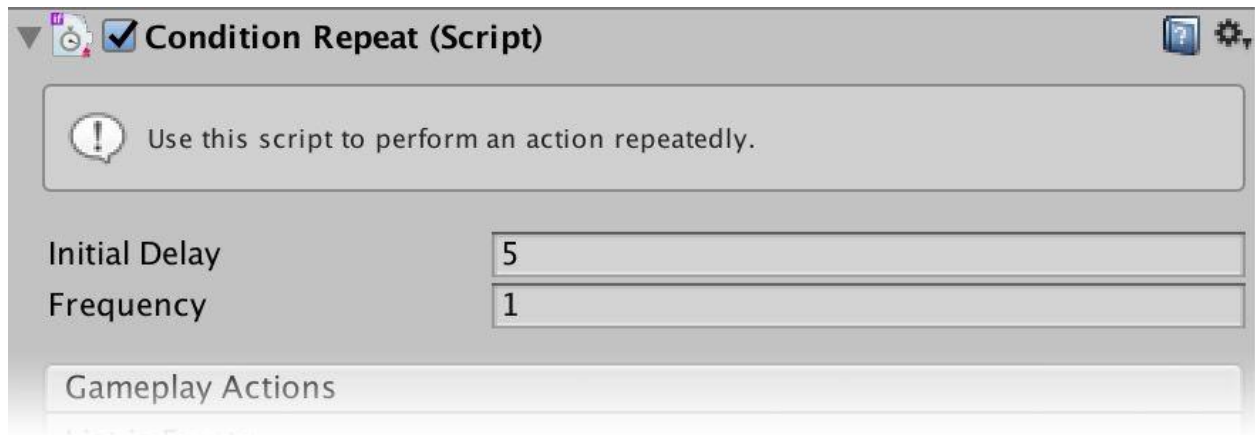
Happen Only Once让事件只能被触发一次。

Gameplay Actions和**Custom Actions**在所有条件脚本上的用法都是一样的，详细说明请参阅上文。



ConditionRepeat 条件-重复

需要组件：没有

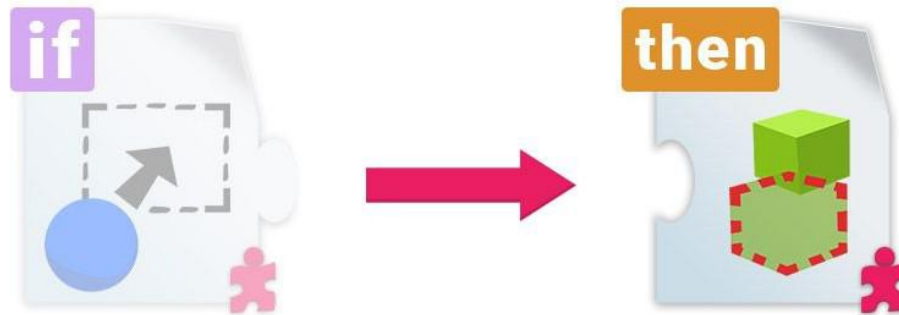


如果用编程语言来做类比，ConditionRepeat更像是WHILE而不是IF。它不需要用户输入或交互就能重复执行Actions列表。它提供**Frequency**频率和**Initial Delay**初始延迟两个属性。

Gameplay Actions和**Custom Actions**在所有条件脚本上的用法都是一样的，详细说明请参阅上文。

Actions 行动

Action行动脚本不能单独工作，它们需要由条件脚本执行。仅在条件成立时，才会执行操作。Action脚本图标左上角有一个黄色的“then”标记。



Action脚本都具有“成功”的概念。这意味着某些Action可能会失败，如果某个Action失败，正在执行这个Action的条件脚本将不再执行剩余的Action。

添加和删除Action

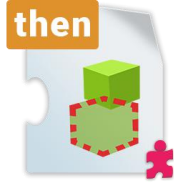
可以像任何其他常规组件一样添加Action，但是他们不会自动运行

- Action 总是需要连接到条件脚本。

因此，最好不要用添加组件的方式来添加Action，而是使用条件脚本的**Gameplay Actions**列表底部的下拉菜单。使用下拉菜单可以同时添加Action组件并将其连接到列表。类似地，减号图标既删除列表中的项目，也从GameObject中删除组件，干净利落。



如果要将Action连接到另一个对象的Condition上，只需将Action添加为普通组件，然后转到另一个对象的Condition列表并使用最后一个选项“Empty Slot”。最后，您需要将具有Action的GameObject拖到刚刚添加的列表槽中。



ConsumeResourceAction 消费资源

需要组件：没有

▼ ☒ Consume Resource Action (Script)

Use this script to check if the player has enough of a specific resource. If they have it, it will be removed from the Player's inventory.

Type of Resource

Amount Needed

ConsumeResourceAction是仅在以下条件成立时才起作用的Action：玩家的物品栏中必须存放着由**Type of Resource**属性指定的资源，并且数量不少于**Amount Needed**属性指定的所需数量。

上述条件成立时，则消耗该数量的资源并执行后续的动作。

如果上述条件不成立，则不消耗任何资源，并且停止执行后续的动作。

与ResourceAttribute脚本中一样，“**Add/Remove types**”按钮允许您定义资源类型。有关如何定义资源类型以及资源概念的详细信息，请参阅ResourceAttribute脚本部分的说明。



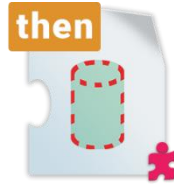
CreateObjectAction 创建对象

需要组件：没有



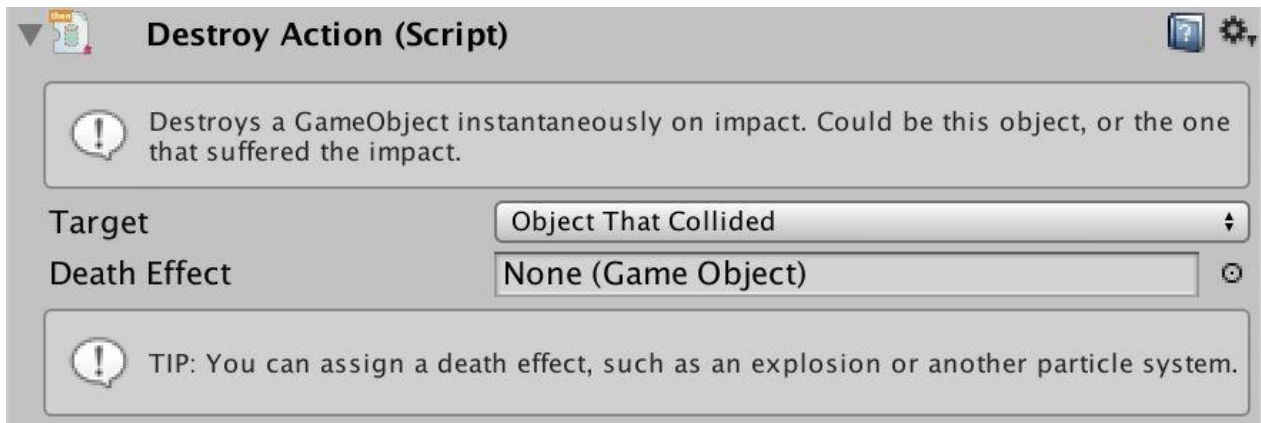
CreateObjectAction从预制件 (**Prefab to Create**) 生成新对象。

New Position属性用于确定新对象的创建位置，默认使用世界坐标（意味着0,0是场景的坐标原点）。**Relative to this Object**被选中时，使用局部坐标。



DestroyAction 销毁操作

需要组件：没有



DestroyAction可用于从游戏中删除对象。

Target属性可以有两个值：This Object（此对象）和Object That Collided（碰撞的对象）。使用后者时，此Action需要连接到ConditionArea或ConditionCollision，否则它将失败。

Death Effect用于选择死亡特效，这是在销毁**Target**对象时生成的另一个对象。它可以是粒子系统或其他物体（如碎片，被销毁物体的破碎版本等）。



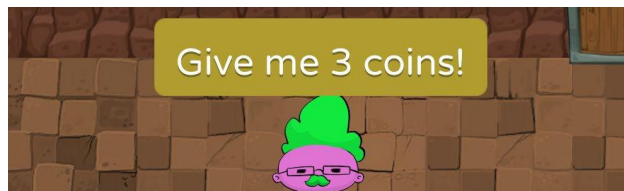
DialogueBalloonAction 显示对话框

需要组件：没有

The screenshot shows the 'Dialogue Balloon Action (Script)' configuration window. It has a title bar with a small icon and a question mark. Below the title bar is a warning icon and the text: 'Use this script to create a dialogue balloon on a character's head.' The window is divided into three sections: 'Contents', 'Options', and 'Continue dialogue'. In the 'Contents' section, 'Text To Display' is set to 'Hey!', 'Background Color' is set to blue, and 'Text Color' is set to white. In the 'Options' section, 'Target Object' is set to 'None (Transform)', 'Disappear Mode' is set to 'Button Press', and 'Key To Press' is set to 'Return'. In the 'Continue dialogue' section, 'Following Text' is set to 'None (Dialogue Balloon Action)'.

| Section | Property | Value |
|-------------------|------------------|--------------------------------|
| Contents | Text To Display | Hey! |
| | Background Color | Blue |
| | Text Color | White |
| Options | Target Object | None (Transform) |
| | Disappear Mode | Button Press |
| | Key To Press | Return |
| Continue dialogue | Following Text | None (Dialogue Balloon Action) |

DialogueBalloon脚本允许在游戏中放置简单的对话框。您可以在Roguelike示例场景中看到它的示例。



第一个属性块，**Text to Display**设置要显示的文本，**Background Color**设置背景颜色，**Text Color**设置文本颜色。

Target Object目标对象（如果已设置）允许文本显示在角色或对象上方。如果未设置，则文本将显示在屏幕中间。

Disappear Mode消失模式允许两个值：Button Press要求用户按下一个键（按键）来移除对话框，Time则让对话框在**Time to Disappear**指定的秒数后消失。

无论使用何种方式移除对话框，您都可以在最后一个插槽“**Following Text**”中连接另一个DialogueBalloonAction，以创建连续对话。

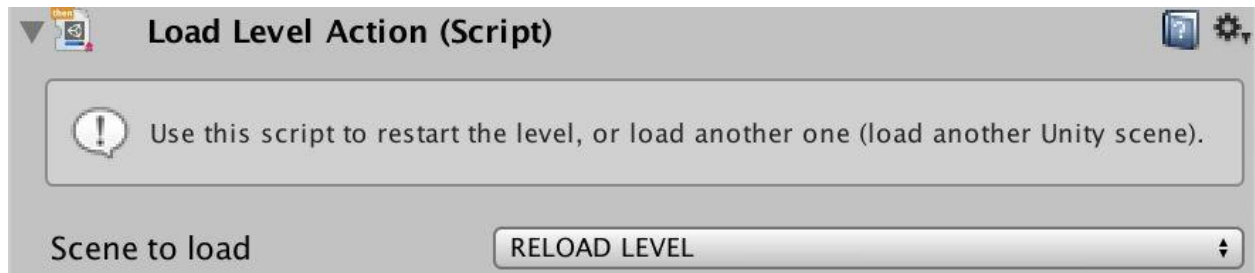
提示：使用没有指定**Target**的多个依次连接的DialogueBalloonAction脚本，并将**Disappear Mode**设置为Button Press，就可以为您的游戏创建小教程。

或者，通过链接多个DialogueBalloonActions并使用**Target Object**属性让它们显示在不同的角色上，您可以在两个或多个角色之间创建对话。



LoadLevelAction 加载场景操作

需要组件：没有



LoadLevelAction用于在条件成立时加载Unity场景。

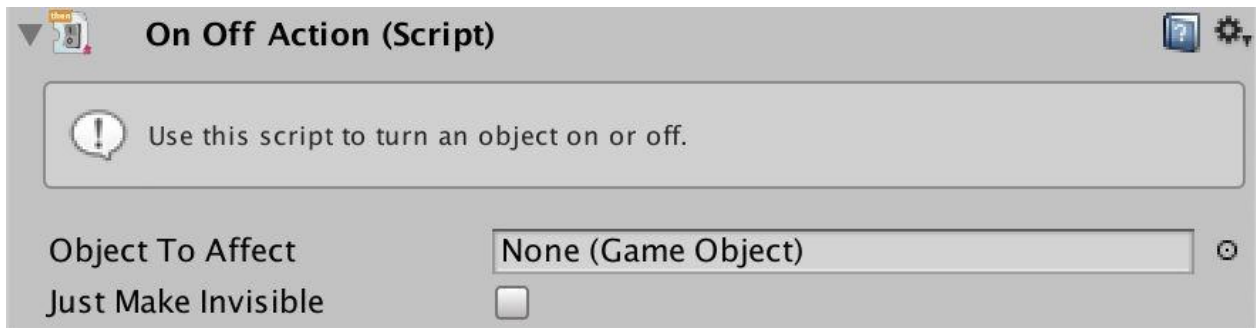
“**Scene to Load**”属性显示一个下拉菜单，其中包含已添加到“Build Settings”菜单中的所有场景（File > Build Settings...）。只有已经添加到Build Settings列表中并启用的场景才能被加载。

第一项“RELOAD LEVEL”只是重新加载当前场景，可用于在游戏结束后重新开始游戏。



OnOffAction 激活和屏蔽对象

需要组件：没有

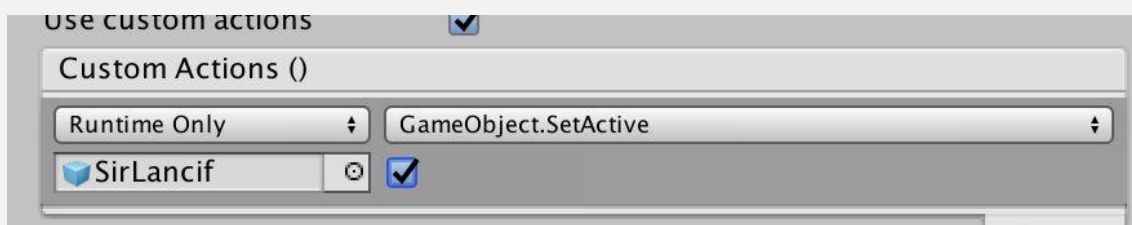


OnOffAction用于激活和屏蔽游戏对象，这意味着将游戏对象的Active标志设置为true或false。您需要在**Object to Affect**中选择目标才能使其生效。

Just Make Invisible允许您打开/关闭SpriteRenderer，这意味着该对象仅仅不能被看见，但仍然会和其他对象发生碰撞和互动。

OnOffAction每次执行都会“翻转开关”，将游戏对象的Active标志设置为与之前相反的值。这意味着每在同一对象上执行两次该Action时，它会恢复其先前的状态，依此类推。

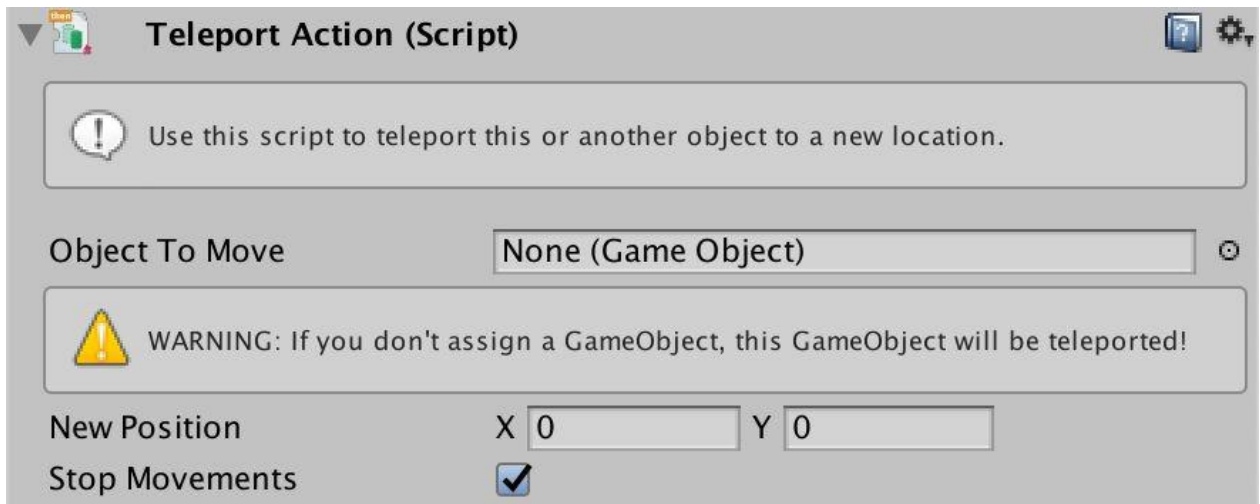
提示：如果要以绝对方式设置对象的开/关（不会每次翻转），您只需要使用一个常规UnityEvent，启用自定义操作并选择目标GameObject上的SetActive函数：





TeleportAction 瞬间移动

需要组件：没有



TeleportAction将对象立即移动到新位置。如果Object to Move属性没有指定任何对象，则会传送该脚本的所在对象。**New Position**属性使用世界坐标。

对于具有Rigidbody2D的对象，启用**Stop Movements**意味着除了被传送之外，它们也会停止移动，这意味着它们的速度和扭矩被归零 - 这有利于重置游戏状态（例如，在体育游戏中得分之后归位）。