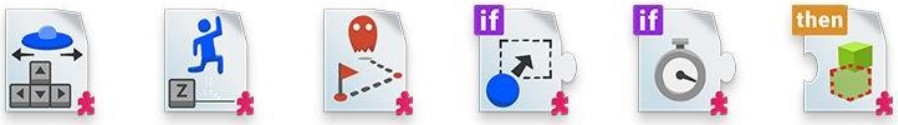
欢迎

无论您是有抱负的创客还是游戏开发的教育者：欢迎来到Unity！
Unity Playground是一个用于创建基于物理的2D游戏的框架，它无需编码就可以让初学游戏开发人员在Unity中制作游戏。它还可以用于介绍游戏设计或关卡设计。

Unity Playground通过提供一系列易于使用和组合的单任务组件，消除了对代码的需求。通过将它们组合在一起，您可以创建多种类型的基于物理的2D游戏。



一些脚本的图标

享受使用Unity Playground！

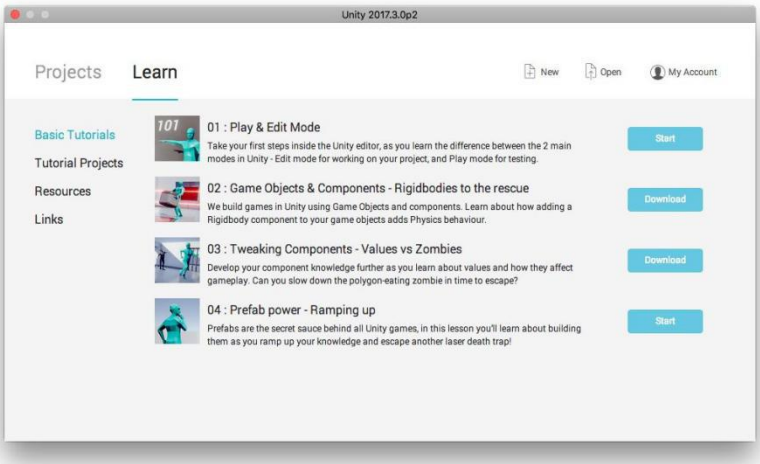
目录

入门

Playground的目的非常简单，您可以在不到30分钟的时间内熟悉基础知识。您只需要了解项目结构（请参阅项目结构）和Playground实现的概念（一般概念）。然后，您可以开始查看脚本，最好是从“运动”类别中的脚本开始。
如果您想了解有关特定脚本的更多信息，请参考指南（学习网站，项目中包含的PDF或在线文档 这里）。

先决条件

在使用Playground之前，您应该已经掌握了Unity的核心概念。官方教程是一个非常好的资源，您可以在编辑器启动窗口或Unity Hub中的Learn选项卡下找到它。您也可以搜索官方教程的接口＆要点部分。



启动窗口中的4个教程

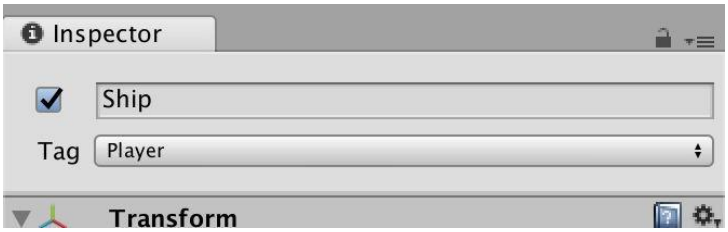
如果您是一名教育工作者，并且您希望将Playground用于研讨会或课程，您可以在教学部分找到有趣的指示。

制作你的第一个游戏

使用Playground做游戏非常容易！让我们做一个非常简单的。

创建玩家

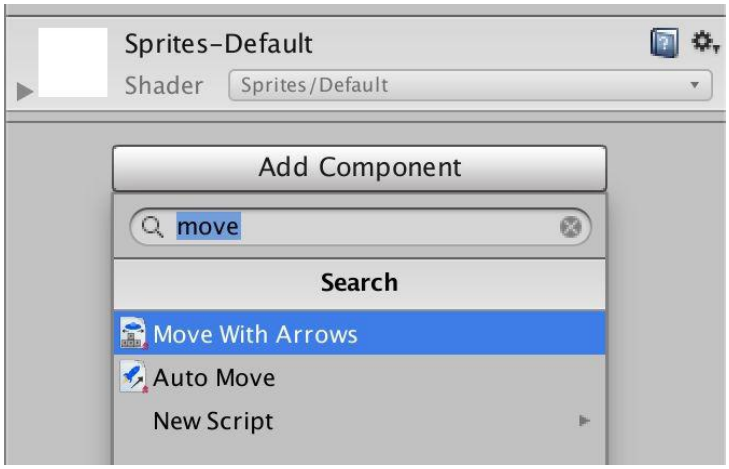
首先将一个宇宙飞船Sprite从/ Images / Spaceships文件夹直接拖到Scene视图中。因为它是一个Sprite，所以Unity会为你创建一个GameObject。这将是玩家，所以让我们将这个GameObject标记为Inspector窗口顶部的“Player”。



该对象现在标记为Player

让我们现在移动飞船。我们需要两个组件：**Move**（移动）以提供交互性，以及一个**Rigidbody2D**以确保船遵循物理定律。让我们将一个**MoveWithArrows**脚本从/ Scripts / Movement文件夹拖到船的Inspector上。

如果拖动有问题，您还可以直接使用Inspector底部的“**Add Component**（添加组件）”下拉菜单，然后键入“Move”。该脚本应作为第一个结果出现。



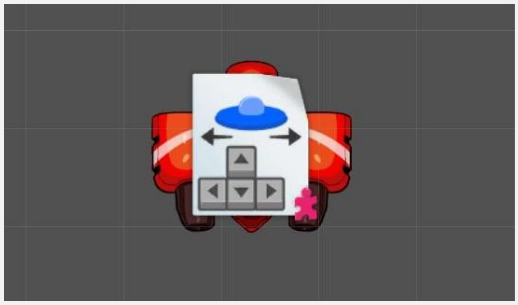
Inspector 中的 “Add Component” 下拉列表

您会注意到，只要添加Move脚本，Rigidbody2D组件就会被自动添加。这是因为Move需要Rigidbody2D才能工作。

注意：

场景中的组件图标 Gizmos

Playground脚本的图标可能很大，覆盖了您的图形。



Move图标覆盖了整艘船！

如果发生这种情况，您可以通过在“场景”视图中使用下拉Gizmos来缩小其大小。将3D Icon滑块向左拖动，直到图标尺寸合适。

“3D Icon”滑块控制场景中 Gizmos 的大小

首先，在**RigidBody2D**组件上，我们要将**Gravity**（重力）修改为0，让飞船不会掉下来。也可以随意选择首选的控制方法（箭头或WASD），如果您需要指定飞船图片的正前方向，请选择Orient to direction，然后选择方向轴。现在按编辑器顶部的Play来测试游戏。

您可能会注意到飞船漂移得很厉害，使其难以控制。调整Rigidbody2D上的**Friction**（摩擦力），使其达到10。

注意：

游戏模式
如果您在游戏模式下编辑组件值，则一旦游戏停止，您将丢失更改。请记住，应该在游戏没有运行的时候编辑组件值！除非您在做临时测试。

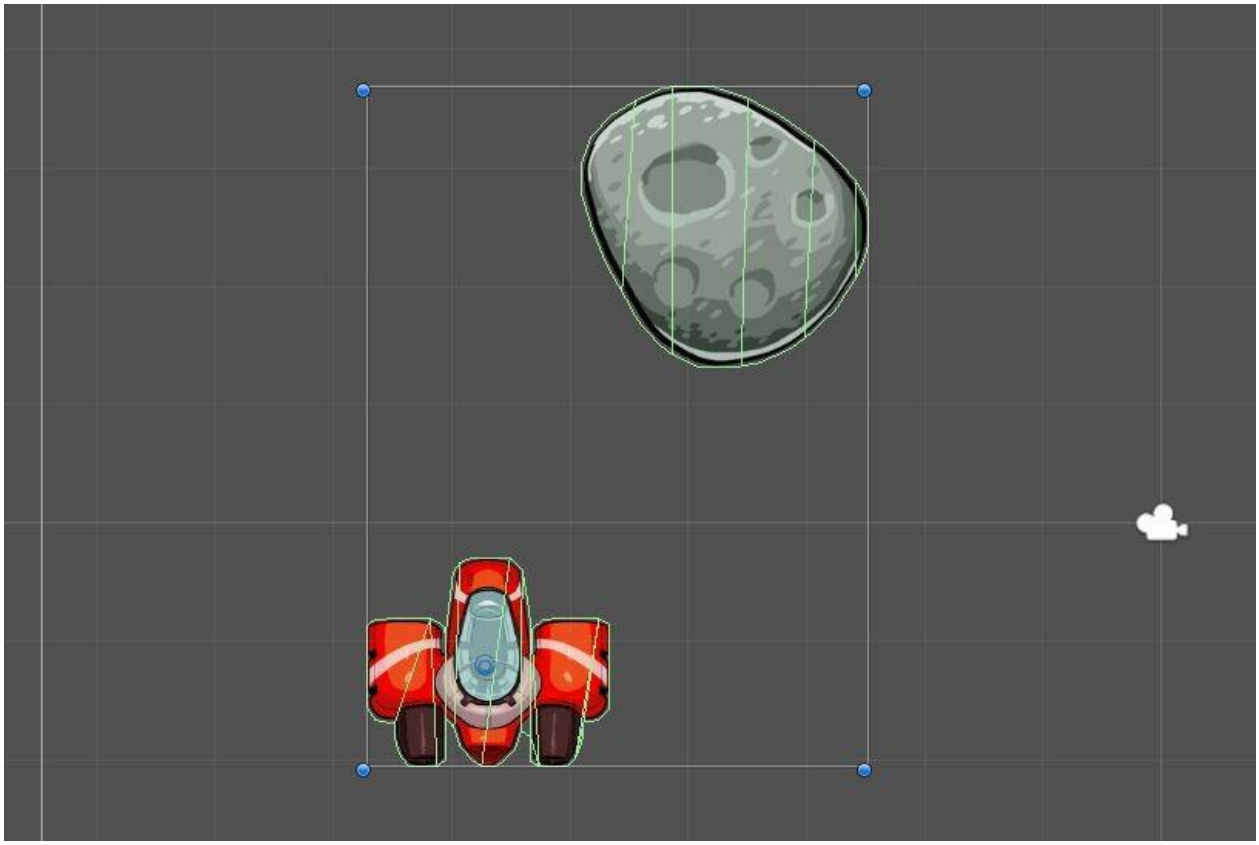
如果你现在玩，你会发现漂移消失了，但物体也慢了很多。这是因为移动它的力现在被摩擦抵消了。

因此，我们还需要在Move组件中调整**Speed**参数。将它调到8，再次按Play。您将看到现在船更容易控制。我们刚刚进行了第一次游戏调整！恭喜：**您已经是游戏设计师了！**

增加障碍和碰撞体

我们有一些可玩的东西了，但它不是真正的游戏。让我们从/ Images文件夹拖动小行星Sprite到场景里，来给航行的飞船制造一些障碍。和之前一样，Unity会为我们创建一个GameObject。

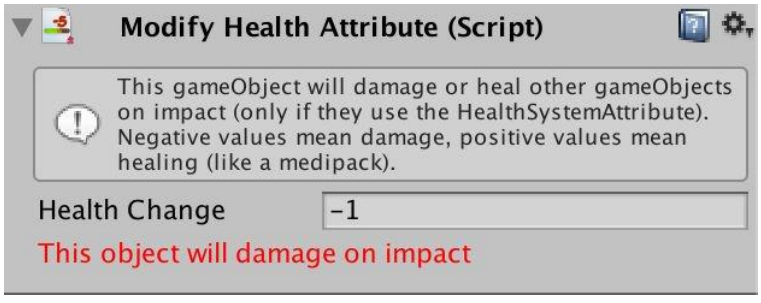
我们需要添加两个组件：**Rigidbody2D**和**PolygonCollider2D**。**PolygonCollider2D**将使它能够碰撞（接触）其他对象。我们也需要将**PolygonCollider2D**组件添加到飞船上。



飞船和小行星显示的绿色边框：碰撞形状

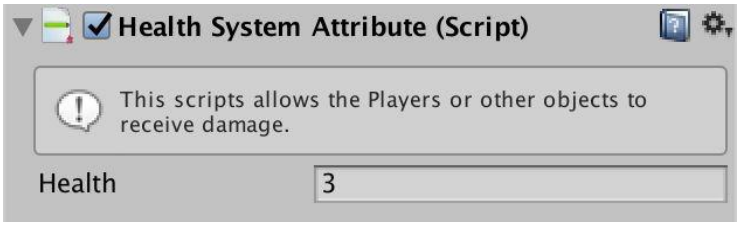
添加碰撞体后，请尝试按“Play”。你会注意到这艘船现在可以推动小行星。不要忘记将小行星的**Gravity**（重力）调整到0，否则它会掉下来！此外，您可以随意调整小行星的参数：**Friction**（摩擦力），**Angular Friction**（角度摩擦力）和**Mass**（质量），以使其按照您想要的方式运行。让我们将**Mass**（质量）设置为10，这样它变得更重，在碰到飞船时就不会被撞飞。小行星的体积越大，就应该越重！

我们现在让这个小行星成为威胁。您需要向小行星添加**ModifyHealthAttribute**脚本，这个脚本位于/ Scripts / Attributes下。



ModifyHealthAttribute的 Inspector 界面

然后，我们需要飞船能够检测到自己受损了。为此，还有一个名为**HealthSystemAttribute**的脚本（也是在/ Scripts / Attributes下）需要添加到太空船。



HealthSystemAttribute 的 Inspector 界面

这不仅可以让玩家具有生命值，还可以让我们设定初始生命值。最后，复制小行星（快捷键：Windows的Ctrl + D，Mac上的Command + D）并在飞船周围创建一个小行星地带。

预制件 Prefab

在复制小行星之前，您可能希望将其制作成预制件，以便稍后可以轻松编辑所有小行星。

如果你不知道预制件是什么，想要了解更多，你可以阅读手册，或观看视频教程

如果您要制作大型游戏或在团队中工作，预制件将是Unity中必不可少的的基本概念，但是现在您也可以将它们放在一边。您可以先专注于您的第一个游戏，有空的时候再回头学习。

现在当玩家被击中时，我们没有任何反馈。所以让我们添加一个UI来可视化玩家的生命值。将**UserInterface** 预制体从/ Prefabs文件夹拖到场景中。在游戏视图中就会出现一个带有分数和生命值显示的UI。

如果你现在玩游戏，你会发现撞上一颗小行星会减少一点生命值！如果你撞上小行星的次数太多，就Game Over了！

现在您已经拥有了一个完整的小行星地带，现在是时候再次测试是否可以在场景里飞行并且不会磕碰太多。游戏必须有难度，但不能太难！反复调整小行星的位置并测试游戏，您现在已经在调整游戏平衡性并进行关卡设计了。

添加游戏目标

那么这个游戏是关于什么的？假设我们希望这艘飞船收集一些星星而不会撞到小行星。一旦收集了所有星星，就赢得游戏。但如果你撞到小行星的次数太多，那就输了！

让我们将/ Images文件夹里的星星图片拖到场景中来添加星星物体。从/ Scripts / Attribute文件夹添加Collectable脚本。这将使得星星成为可收集的物品，每收集一个星星将得到一分。

但是我们如何检测这颗星星是否已被收集？我们将再次使用碰撞。添加一个PolygonCollider2D，但这次我们想让它成为一个触发器，所以启用Is Trigger属性。

注意：

Trigger 触发器
触发器是一种特殊类型的碰撞体。它们使物体变得无形，所以看起来它不能碰到其他物体。但Unity会检测到两个对象何时相互接触，以便您在它们接触的时候执行操作。

例如，触发器可以检测玩家是否已达到关卡的末尾。将一个触发器放在终点之前，当玩家接触它时，显示一个You Win消息。触发器不会阻挡玩家前进，但可以提供检测获胜条件的逻辑。

如果你现在玩游戏，你会发现这颗星被飞船收集了。此时，您可能还想让星星成为预制件（请参阅上文有关预制件的说明），然后根据您的需要将其复制多次，例如5次。现在我们想要四处摆放它们，让一些星星很容易获得，而另外一些则没那么容易。这样我们的小游戏就有了一个逐渐提升的难度。

最后，您要选择UI GameObject，并在UI脚本上确保游戏类型为Score，并且所需得分为5。如果所需得分比游戏中的星星多，那么游戏就没法通关了！

再次按“播放”，检查您是否可以赢得游戏。如果你得到所有5星，你应该看到一条消息说“玩家1获胜！”。

恭喜：你刚刚完成了你的第一个游戏。做得好！继续调整它，直到您对控件，难度和关卡布局感到满意为止。

下一步

现在您已经初步掌握了Unity和Unity Playground，您可以继续使用它制作更复杂的游戏。作为第一步，您可能想要了解Playground的一般概念以更好地理解它。

请记住：Playground是非常灵活的，你可以设计自己的原创游戏类型或复制一些现有的游戏！

注意：
在学习如何制作游戏时，复制80年代旧游戏的游戏玩法（如乒乓，打砖块，太空侵略者，沙罗曼蛇等等.....）通常是一个好主意，因为它们很简单。然后，您可以添加更多的细节。

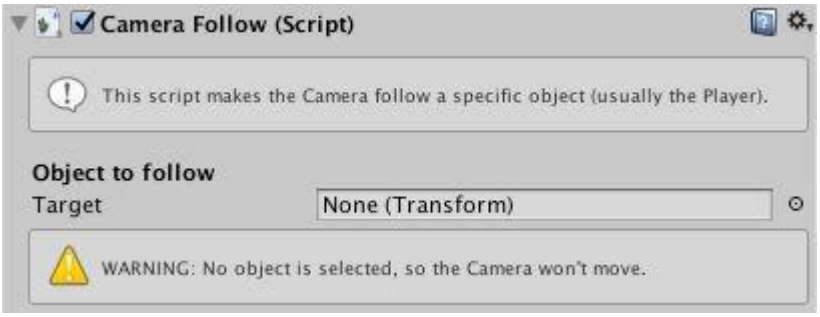
如果您需要灵感，请打开Examples文件夹并启动其中一个游戏。看看里面的GameObject是如何制作的，然后尝试创建类似的东西。

一般概念

在本节中，我们将非常快速地介绍Playground的一些基本概念。了解这些概念以后，您可以使用它们来创建更复杂的游戏。在您阅读这些内容之前，我们建议您完成上文的教程。

信息和警告

Playground中的所有自定义脚本顶部都有一个小信息框，简要说明了脚本的功能。



摄像机跟随脚本中的信息和警告

类似地，许多脚本都有几条警告消息（带有黄色危险标志的消息），当设置不正确时会出现这些消息。留意这些，他们可能会给你一个很好的指引，说明脚本无法正常工作的原因。

碰撞体和触发器

Playground中的几乎所有逻辑脚本都使用碰撞来创建游戏玩法。这意味着当具有碰撞体的2个对象彼此接触时，或者当具有碰撞体的对象进入另一个对象（它也具有碰撞体，但标记为触发器）时，可以触发游戏里的事情。

例如，你可以在与敌人碰撞时造成伤害，在接触某个物体时通关，走到金币或者强化道具身上来收集它们，或者让已经捡到钥匙的角色冲到门前来打开门。
类似地，当角色进入另一个角色面前的区域时（例如在Roguelike示例场景中），您可以触发一行对话。

当某些东西不能正常工作时，先问问自己：碰撞体加对了吗？

标签

标签允许我们将对象划分为不同类别，以便脚本只有在接触正确的对象时才能执行操作。没有标签，就没有办法区分对象。

与碰撞一样，许多脚本要求目标对象被打上正确的标签才能正常工作。如果您的脚本没有按预期执行操作而您不清楚原因所在，请阅读参考指南以检查您是否遗漏了一些重要的标签设置。

导入自定义图形

Playground包含很多Sprite图片，位于Images文件夹下。但是，您也可以自由使用任何2D图形！

要在Playground中使用图像，只需将其拖动到Assets文件夹中的某个位置即可。图像将作为Sprite导入，然后就可以使用了。只需将其拖动到场景或Hierarchy层次结构视图，就会自动创建一个新的GameObject，然后您就可以在游戏中使用它。

请记住，要保证图像效果，非正方形的图像需要具有透明度，否则它们将显示白色背景。允许透明度的建议格式是.png或.gif，而.jpg不包括透明度。还要记住，Unity不会播放GIF动画。

高级概念

如果你是老师或者教育家，或者是希望更好地了解Playground的高级用户，建议阅读这一部分。

备忘单

乍看之下，可能很难理解Playground中有多少脚本。为此，Playground附带了一系列备忘单，共有6页，其中包含所有脚本的图标和单行描述。它们按类别（运动，游戏机制等）进行组织，并用不同颜色标出。

一些页面的预览

备忘单位于项目文件夹，包含PDF和JPG两个版本，分别适用于打印，以及屏幕演示。如果学员对Unity还不熟悉，将这些资料打印出来发给他们，可以帮助他们了解学习范围以及各个组件可能实现的功能。

除此之外，第七页还有一些不同性质的额外挑战。您可以使用这些来鼓励参与者尝试新的东西，或者给予他们约束（有点像Game Jam的主题）。

项目结构

要在课程中使用Playground项目，您可以让人们从资源商店下载（免费），如果无法访问互联网，您可以自行将Assets和ProjectSettings文件夹分发给学生。

资产文件夹

Documentation文件夹包含PDF文档。您可以在阅读完这份入门文档以后，打开Reference Guide参考指南深入了解每个脚本的详细信息。备忘单可以被打印出来并分发给学习者，以便他们快速浏览脚本清单及其作用，以获得灵感。

Images and Particles 文件夹包含可用作角色，敌人或组成场景的图形资源，但开发人员可以根据需要自由导入新图形。

Playground的核心是脚本，它们位于Scripts文件夹中。它们中的大多数应该开箱即用，尽管有些需要以给对象设置特定标签才能工作（在“标签”部分中阅读更多相关信息）。

还有一个名为Examples的文件夹，你可以在其中找到一些已经制作的小游戏。您可以将它们用作学习材料，或作为自定义的起点。

项目中有一个名为_INTERNAL_的“特殊”文件夹。顾名思义，除非你想修改Playground的内部运作，否则不应该触及它。它包含基本脚本，字体，Gizmos以及学习者不需要了解的其他内容。它需要在项目中，让Playground正常工作。

本节将更深入地介绍Playground，以便您更好地理解它的内部工作原理。那些必须帮助学生解决游戏中问题的教育工作者最好阅读一下本节内容。

标签

某些脚本使用标签来筛选对象并决定它们是否应该发挥作用。某些脚本会在OnCollisionEnter2D 或 OnTriggerEnter2D 方法中筛选标签，而另一些脚本（如HealthSystemAttribute）则根据对象是否标记为玩家而表现不同。

如果您在开始时导入ProjectSettings文件夹，则已定义了一些额外的标记。特别：

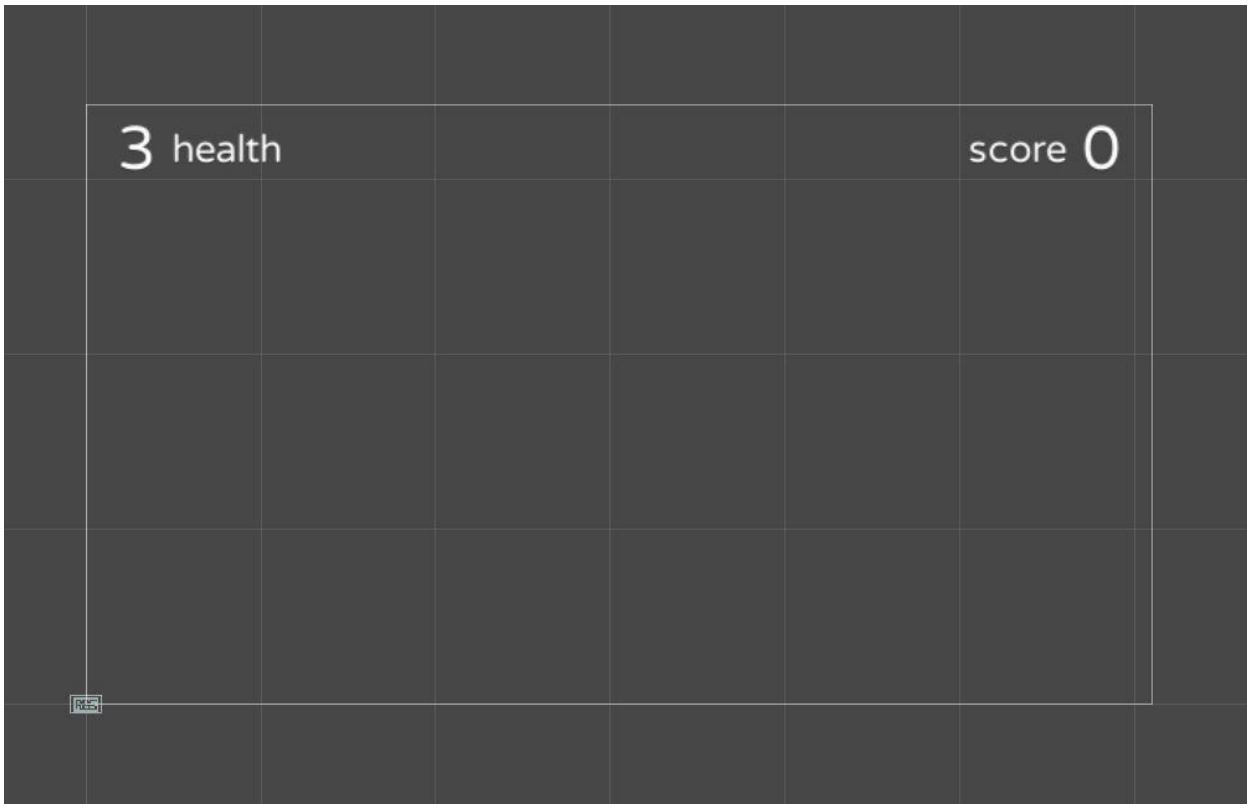
- 双人游戏中，两个玩家必须分别被标记为Player和Player2才能使UI、得分等正常工作。
- Enemy 目前没有在任何脚本中使用，可用于定义那些物体能被子弹和武器伤害。
- Bullet 是用于子弹的标签。
- Ground 用于检查玩家跳跃时接地的地面。

除了Player标签之外，许多脚本允许您定义要查找的标签。例如，Jump脚本会要求您定义什么是地面，因此您不一定需要选择“Ground”标记。预设标签仅是一种建议。

您可以在Condition 条件脚本中筛选标签，这样只有当您与带有某个标签的对象发生碰撞时才会发生某些事情。标签列表会列出所有标签供您选择，不用打字，不用担心拼写错误。

用户界面

UI被做成了一个预制件，包含在Prefabs文件夹中。只需将UI拖动到场景中，您就可以自动获得玩家1的健康和分数，但您也可以选择该游戏适用于2个玩家，UI会根据游戏规则为两个玩家显示分数或健康状况。



空场景上的 UI画布

UI允许在Score，Life和Endless之间选择游戏规则。达到胜利或失败条件时，将显示Game Over或You Win屏幕：

- 在Score 分数模式下，如果玩家达到规定分数，将显示“胜利”屏幕。当两个玩家在场时，UI将仅显示分数，第一个达到规定分数的玩家获胜。
- 在Life 生命值模式中，如果玩家生命值降到0，将显示游戏结束。有两个玩家时，会显示两者的生命值。这种模式没有胜利画面。
- 在Endless 无尽模式下，不会显示结束游戏屏幕，也没有赢或输。

自定义游戏类型

使用Condition脚本时，可以将UIScript的GameWon和GameOver函数连接到一个自定义的UnityEvent。这样，您可以通过利用碰撞和其他事件来创建自定义的胜利或失败条件。

同样，您可以将AddOnePoint和RemoveOnePoint函数连接到一个自定义的UnityEvent。创建自定义的加分和减分操作。

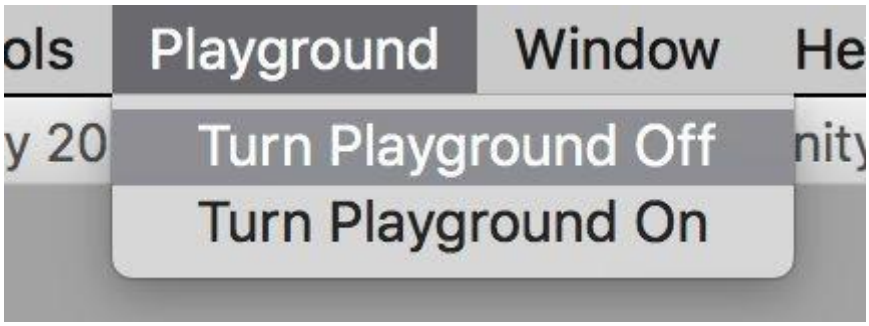
自定义检视面板

Unity Playground大量使用自定义检视面板，包括Playground自己的脚本（Move，Jump等）和默认的Unity组件（Transform，Collider，SpriteRenderer等）。这样做的目的是简化Unity UI，让新用户更易上手。

因此，一些变量名也被更改（即Rigidbody2D中的Drag已更改为“Friction”）。

禁用 Playground 检视面板

您可以从顶部菜单栏打开和关闭自定义检视面板（如下图）。这允许您恢复Playground脚本隐藏的Unity默认界面。您可以随时回到默认界面作更改，然后重新打开Playground。由于Playground只是定制了检视面板的外观，来回切换不会影响正常工作。



当讲师需要暂时回到Unity默认界面修改被Playground隐藏的参数时（例如，编辑任何类型的Collider的形状），可以使用此方法。

Tilemaps 地图块

图形素材中包含一系列用于构建示例游戏的方块Sprites（草，鹅卵石，木材），您可以使用Unity的Tilemap 功能构建自己的关卡。

如果您想了解更多关于Tilemap的信息，可以在Unity手册中找到详细的指南。
学习网站也有很多关于它的好教程：我们建议从Intro to 2D World Building 视频开始，或者您可以在2D Gamekit系列教程的Painting a Level部分找到简短的文字说明。

附录

参与开发

如有问题，建议，请随时发送电子邮件给西罗·孔迪西奥（或通过推特联系）。如果您想为项目做出贡献，请在Github上查看，创建分支，并提交合并请求。

注意

我们将考虑所有建议，但请记住，Playground的目标不是拥有尽可能多的脚本和功能。拥有合理数量的内容使其更有针对性，更容易学习。我们相信，一旦学习者熟悉了整个Playground 并想要使用更多功能时，他们更应该尝试从头开始创建自己的游戏。

制作人员

Unity Playground是由西罗·孔迪西奥开发的。Stefano Guglielmana制作了美术资源。

特别感谢

Unity Technologies，特别是布莱顿内容团队，他们使 Playground 在资源商店成功上线。

Kenney.nl免费提供了用于 Playground 初次迭代中使用的图片素材。

迪奥塞林·冈萨雷斯, 约翰西茨玛相信 Playground 的潜力并首先将它引入课程。

斯丁·凯罗博尔提供了一些宝贵的早期反馈，以及标志设计的建议。
尼科琳·赫和内维·埃伦德热情提供了用户体验的早期反馈。
Phil Jean提出了一个重要的修改建议，这成为了项目的转折点。

Github存储库的众多贡献者：索菲亚克拉克, 伊桑布鲁斯, 马克苏特, 吉姆 “吉博”皮克顿, “凯旋门山”，以及所有不能一一致谢的其他人！

Coderdojo Brighton的孩子们和Unity的新员工入职培训的参与者，他们是首批测试用户。
最后，感谢使用Playground 的所有教师和教育工作者！