场景

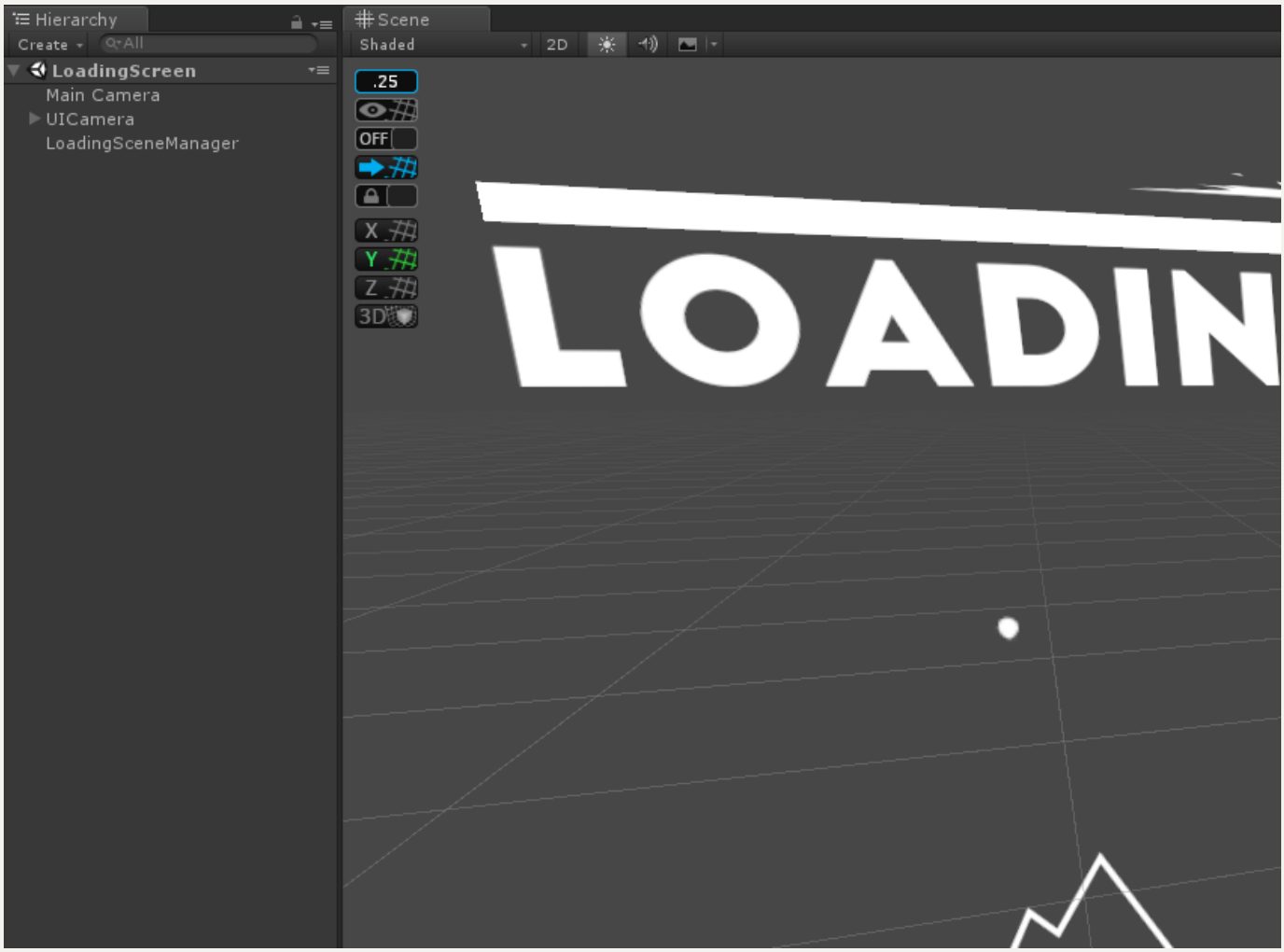
本页介绍了如何设置场景以及如何从一个场景切换到另一个场景。

介绍

您的游戏很可能由多个级别或部分组成。在自上而下引擎中，就像在 Unity 中一样，您可以通过多种方式构建关卡。最直接的方法是为每个关卡创建一个场景，并在关卡末尾设置一条通向下一关卡的门/大门/终点线，但您也可以在单个场景中创建多个部分。最重要的是，该引擎提供了链接关卡、保存数据、创建关卡选择屏幕和角色选择屏幕的方法。本页涵盖了所有这些。

使用加载场景管理器转到另一个场景

使用 Unity，通常当您想要转到另一个场景（例如在菜单中，或者从一个级别转到下一个级别）时，您会使用**SceneManager API**，并且可能使用 SceneManager.LoadScene() 方法。本机方法没有为玩家提供足够（或任何）视觉反馈，例如，移动设备上的场景加载可能需要几秒钟的时间，因此只有黑屏并没有真正好看。为了解决这个问题，引擎附带了自己的场景更改 API，如果您不喜欢它，您可以完全自由地不使用它。



场景视图中的 LoadingScene

如果您想为您的播放器提供更好的体验，您可以使用**MMSceneLoadingManager**：

- 它可以从**任何地方**调用，您的场景中不必有 MMSceneLoadingManager
- 它处理加载（顾名思义），显示**动画和进度条**
- 它是完全**可定制的**，只需编辑 ThirdParty/MoreMountains/MMTools/Tools/MMSceneLoading/LoadingScreens/LoadingScreen 场景的内容。您可以轻松添加自己的徽标、更改进度条的外观、正在播放的动画等。
- 这是很**简单的**来使用

要使用**MMSceneLoadingManager API**，当您想更改级别时，只需调用 MMSceneLoadingManager.LoadScene (string sceneToLoad) 方法。您传递的字符串参数当然必须与您尝试加载的场景的名称相匹配。因此，例如，如果您要加载 Koala Dungeon 级别，则可以使用：

```
MMSceneLoadingManager.LoadScene ("KoalaDungeon");
```

引擎会处理剩下的事情 

通过 LevelManager 加载场景

从 LevelManager 的检查器中，您可以定义 LoadingSceneMode。然后引擎中的一些类（如 FinishLevel 脚本或 LevelSelector）将使用此方法从一个场景转到另一个场景。三种模式可用：

- **UnityNative** : 将简单地使用 SceneManager API 加载一个新场景
- **MMSceneLoadingManager**，将使用 MMSceneLoading API 加载新场景，如上所述。在该模式下，您必须指定要用作加载场景的场景。开箱即用，您可以设置"LoadingScreen"以使用同名场景。确保将其添加到构建设置中。
- **MMAdditiveSceneLoadingManager**，将使用更高级的附加场景加载系统。您还必须指定加载场景名称，您可以选择位于第三方/MoreMountains/MMTools/Tools/MMSceneLoading/LoadingScreens/LoadingScreen 的任何附加场景名称。如果您不知道选择哪一个，请从 MMAdditiveLoadingScreen 开始。这种模式提供了许多选项，让您可以完全控制每一步的时间，包括淡入淡出、线程优先级、延迟等。

等级选择

该引擎带有一个内置的级别选择示例。它是一个包含显示每个级别信息的卡片的旋转木马，LevelSelection 场景。逻辑上相当简单，只需使用LoadingSceneManager就可以进入选中的关卡。

场景中的房间

在某些情况下，在更大的场景中创建不同的部分可能是个好主意。在这种情况下，您需要使用 Teleporter 组件。您可以在许多演示级别中看到它的运行情况，并在 Minimal2DDoors1 演示场景中重点关注它。它使用起来非常简单，您的场景中只需要其中两个（或更多），您可以通过它们的检查器链接它们，定义它们的行为，然后就可以开始了。

移动到另一个场景中的某个点

该引擎还允许您将不同的场景链接在一起，并根据您来自何处指定每个场景的开始位置。也许你走到场景 A 的顶部，那里有一扇门，把你送到场景 B。但也可能在场景 A 的底部有一扇门，把你送到场景 C。一旦你进入场景 B，你想回到 A 的顶部，等等。这正是 Minimal2DDoors1 和 Minimal2DDoors2 演示场景所展示的。他们使用**GoToLevelEntryPoint**组件。

它很容易使用，但需要一些设置。首先，您需要在目标关卡中创建入口点（如果您在场景 A 中并想转到场景 B，则您的目标关卡是场景 B）。要在场景 B 中创建入口点，只需创建和定位空对象，或选择现有对象（例如检查点）。然后选择你的场景 B 的 LevelManager，在它的检查器中，定义这个场景需要多少个入口点，并将它们一一绑定在那里。

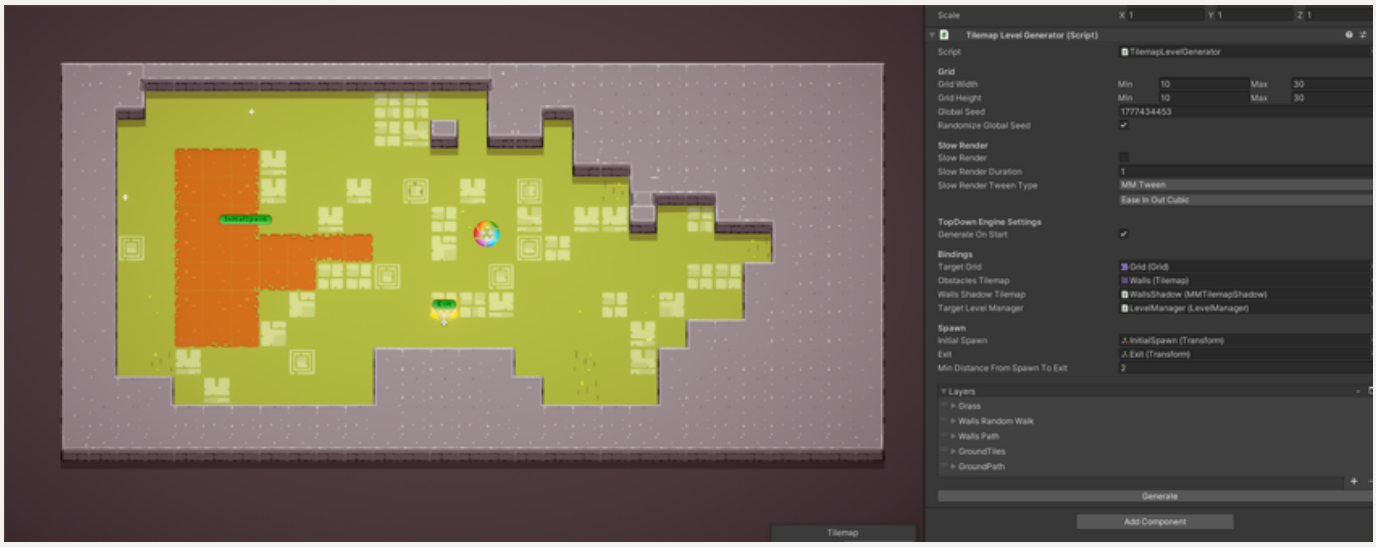
完成后，返回场景 A，创建一个带有**GoToLevelEntryPoint**脚本的门/对象。根据您的喜好设置各种激活条件和设置，相应地设置级别名称（在我们的示例中为场景 B），并为此对象设置入口索引。请记住，索引从 0 开始。因此，如果 SceneB 的 LevelManager 中有 3 个入口点，则索引 0 是该列表中的第一个点，索引 1 是第二个，依此类推。就是这样，你很高兴去！

级别的程序生成

虽然引擎中的大多数演示都具有手工制作的关卡，在场景开始时已经在编辑器中确定，但没有什么能阻止您按程序生成关卡。您只需要注意两件事：

- 确保您在 Awake 时生成关卡，以便它们可以在开始时使用。
- 确保您的关卡符合引擎的规格，并且生成的结果与您手动完成的结果相匹配（墙壁位于 Obstacles 图层上，您的角色有可以行走的地面，关卡管理器的边界已正确调整大小， 等等）。

您将在 KoalaProceduralTilemap 演示场景中找到它的示例。此场景使用 MMTilemapGenerator 的扩展版本，即**TilemapLevelGenerator**。该**TilemapLevelGenerator**是定制的自上而下的引擎制作，并不会只基于你所定义的规则的新tilemap的，也将自动生成墙上的影子，将在合理和可到达位置的入口和出口，和将调整级别经理。您可以按原样使用它，如果它符合您的需求，或者您可以参考它来实现您自己的程序生成系统。



TilemapLevelGenerator 的检查器和生成级别的示例

TilemapGenerator 通过生成一个由 0 和 1 组成的网格，并用它们绘制一个 Unity Tilemap（0 保持为空，1 变成一个完整的图块）来工作。它使用多种不同的算法来生成这些 0 和 1 数组，并允许您组合和自定义它们以获得非常独特的结果。

这是其检查器的细分，它可以让您生成各种级别：

- **Grid**：让您定义网格的大小。您可以为其宽度和高度设置最小值和最大值。这将在这些最小和最大边界之间生成一个随机大小，用于您不希望级别始终相同大小的情况。当然，如果您为宽度的最小值和最大值设置相同的值，它将保持不变。
- **全局种子**：这个生成器是种子的，这意味着如果你使用相同的种子，你将总是得到相同的结果。例如，如果您希望您的角色能够回到之前生成的关卡，这可能会很有用。您所要做的就是存储该种子并使用它重新生成一个级别。复选框还允许您自动生成新种子，但如果您愿意，您可以自行处理该生成。
- **慢渲染**：这只是为了展示，并且只在运行时有效。它可以让您逐块渲染关卡的生成。这看起来不错的样子。如果您要在测试目的之外进行操作，则不太实用（并且可能会导致问题）。
- **自上而下引擎设置**：一个复选框将让您决定是否应在唤醒时自动生成。绑定将让您定义要使用的网格、使用什么瓷砖地图作为障碍物、在什么样的瓷砖地图上绘制墙壁阴影，最后一个插槽将让您绑定您的 LevelManager。然后你可以绑定你的 InitialSpawn 和 Exit，以及你想在它们之间保持的距离。生成时，系统会生成一个新的关卡，然后将这两个放置在可到达的位置。
- 层：这就是魔法发生的地方。每个图层将由要绘制的目标瓦片图和定义如何绘制瓦片的设置组成。这些设置的细分如下：
- 图层：名称：纯粹用于您自己的组织目的，通常指定该图层将绘制的内容
- **Layers : Active** : 如果未选中此项，则在生成关卡时不会处理该分层

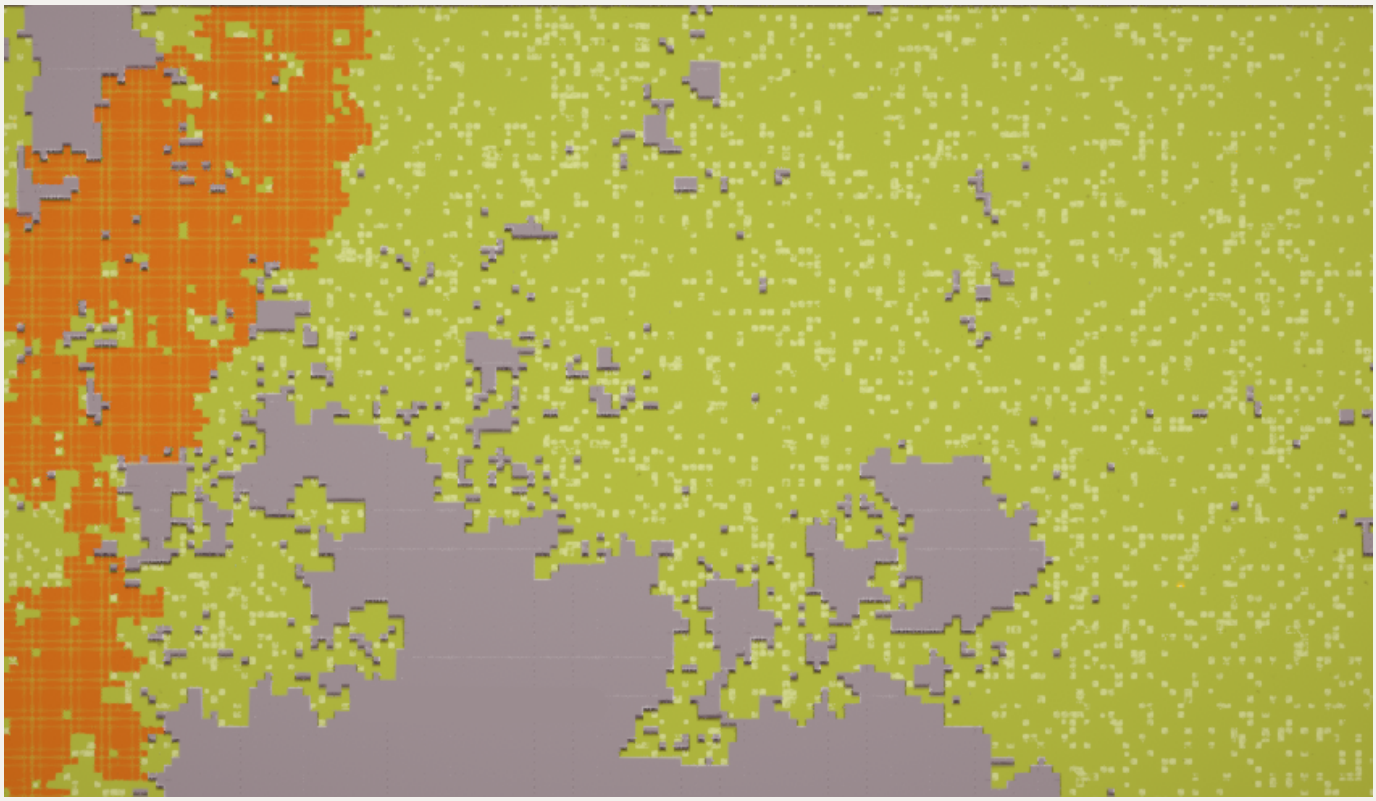
- **Layers : Tilemaps** : 允许你拖动一个目标 tilemap 来绘制，以及使用什么 Tile 来绘制它
- **Layers : Override Grid Size** : 默认情况下，每个层都将绘制在一个与生成器的常规网格大小设置大小相同的网格上。但是您也可以在每层基础上覆盖它。
- 层：后处理：平滑选项将对生成的网格应用平滑，衰减孤立点和各处的尖峰。反转网格会将所有 0 变为 1，反之亦然。如果您对某个形状感到满意，但又想"如果它与那个形状相反就好了"，这就是适合您的复选框。
- 图层：融合模式：这些融合模式将从上到下应用于图层（最后发言者获胜）。Normal 只是生成一个网格，默认模式。NormalNoClear 生成一个网格，但不会先清除它。Intersect： 在已经有内容的目标网格上绘画时，只会保留产生的交集。合并：将此网格的结果添加到现有目标。减去：从现有目标中删除此网格的结果。
- 层：生成方法：许多可用于生成模式的算法。Full : 填充整个网格。Perlin： 使用柏林噪声生成模式。PerlinGround： 同样的事情，但旨在在网格底部保持一个可步行的"地面"。Random： 随机绘制 0 和 1。RandomWalk： 选择一个起始位置，然后像一个代理在路径上留下踪迹一样进行绘制。RandomWalkAvoider： 同样的事情，但Walker 将在您可以指定的单独瓷砖地图上避开障碍物。RandomWalkGround： 随机绘制一条小路，但尽量保持可步行的地面。路径：雕刻指定随机尺寸的路径。复制：复制目标瓦片地图。
- **Layers : Bounds**：用 1s 绘制左侧、右侧、顶部或底部边界（或这些边界的组合）。
- **Layers : Safe Spots**：让您定义许多安全点（在网格坐标中定义），其中该层只会绘制 0。

TopDown Engine 的 KoalaProceduralTilemap 演示场景中的图层

这个系统可能需要一些时间来适应，但它功能强大，可以作为您自己的更大解决方案的良好基础。不要犹豫，看看 KoalaProceduralTilemap 演示场景，在里面你会发现一个已经设置了 5 层的生成器来生成有趣的模式：

- 草：这一层只是在地面瓷砖地图上绘制一个完整的草瓷砖网格，作为我们关卡的基础
- 墙壁随机游走：使用随机游走生成，该层用墙壁绘制地图的 50%
- 墙壁路径：此图层使用组合融合模式为由其上方的图层生成的墙壁网格添加额外的垂直路径
- 地砖：该层随机在地面上掉落装饰砖，目标是20%的填充率
- 地面路径：该层使用 RandomWalkAvoider 算法在地面上绘制橙色路径，试图避开墙壁以提供逼真的结果

最后一件事：这个系统没有限制。该演示生成相对较小的关卡（测试速度更快），但对上述图层进行一些调整并将网格大小乘以 10，您可以生成整个世界：



只需点击几下即可生成更大的世界