**创建导航网格代理**

为关卡烘焙导航网格后，即可创建能够在场景中导航的角色了。我们将使用圆柱体构建原型代理，并将代理设置为运动状态。为实现此目的，需要使用导航网格代理 (NavMesh Agent) 组件和简单脚本。

首先，让我们创建角色：

1.创建一个**圆柱体**：\_\_GameObject > 3D Object > Cylinder**。 2.默认的圆柱体尺寸（高度 2 和半径 0.5）适用于人形代理，因此我们将这些尺寸保持原样。 3.添加一个导航网格代理组件：\_\_Component > Navigation > NavMesh Agent**。

现在已设置简单的导航网格代理来准备接收命令！

开始尝试使用导航网格代理时，很可能需要根据角色大小和速度调整代理的尺寸。

\_\_导航网格代理\_\_组件将负责角色的寻路和移动控制。在脚本中，导航的设置十分简单，只需设置所需的目标点：导航网格代理可从此处进行所有内容的处理。

// MoveTo.cs

using UnityEngine;

using System.Collections;

public class MoveTo : MonoBehaviour {

public Transform goal;

void Start () {

NavMeshAgent agent = GetComponent<NavMeshAgent>();

agent.destination = goal.position;

}

}

接下来，我们需要构建一个简单的脚本，通过此脚本可将角色发送到另一个游戏对象指定的目标，并构建一个用作移动目标的球体：

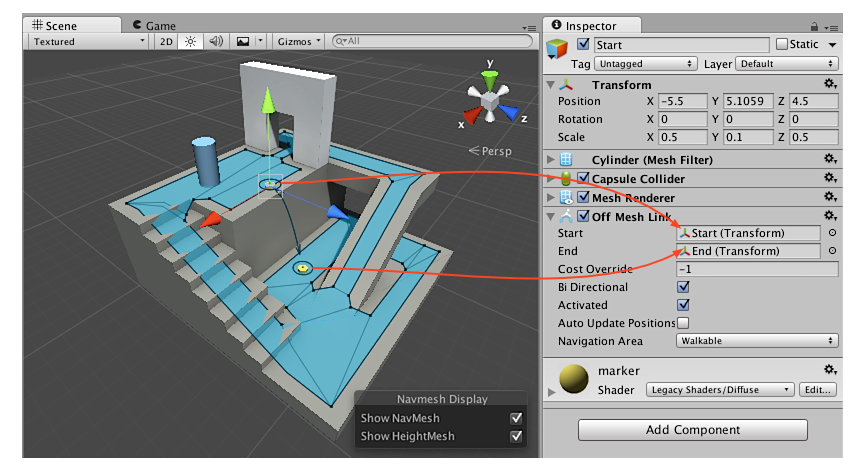
1.创建一个新的 **C# 脚本** (MoveTo.cs) 并将其内容替换为以上脚本。 2.将 MoveTo 脚本分配给刚刚创建的角色。 3.创建一个**球体**，此球体将用作代理移动到的目标。 4.将球体从角色移动到靠近导航网格表面的位置。 5.选择角色，找到 MoveTo 脚本，并将球体分配给 **Goal** 属性。 6.**按 Play**；您应该会看到代理导航到球体的位置。

总而言之，在脚本中，您需要获取对导航网格代理组件的引用，然后为了将代理设置为运动状态，只需将一个位置分配给其 [destination](file:///E:\UnityDocumentation\ScriptReference\AI.NavMeshAgent-destination.html) 属性。[导航操作方法](file:///E:\\UnityDocumentation\\Manual\\nav-HowTos.html)部分提供了进一步的一些示例来说明如何使用导航网格代理解决常见的游戏问题。

# 创建网格外链接

网格外链接 (Off-Mesh Link) 用于创建穿过可步行导航网格表面外部的路径。例如，跳过沟渠或围栏，或在通过门之前打开门，全都可以描述为网格外链接。

我们将添加一个网格外链接组件来描述从上层平台到地面的跳跃。



1.首先创建**两个圆柱体**：\_\_Game Object > 3D Object > Cylinder\_。

2.可将圆柱体的比例设置为 (0.1, 0.5, 0.1)\_ 以便于使用它们。

3.将**第一个圆柱体**移动到上层平台的边缘，靠近导航网格表面。

4.将**第二个圆柱体**放在地面上，靠近导航网格，在链接应着陆的位置。

5.选择左侧的圆柱体，并为其添加网格外链接组件。从 Inspector 中选择 **Add Component**，然后选择 **Navigation > Off Mesh Link**。

6.在 **Start** 字段中分配最左侧的圆柱体，并在 **End** 字段中分配最右侧的圆柱体。

现在设置好了能正常运行的网格外链接！如果通过网格外链接的路径短于沿导航网格行走的路径，则将使用网格外链接。

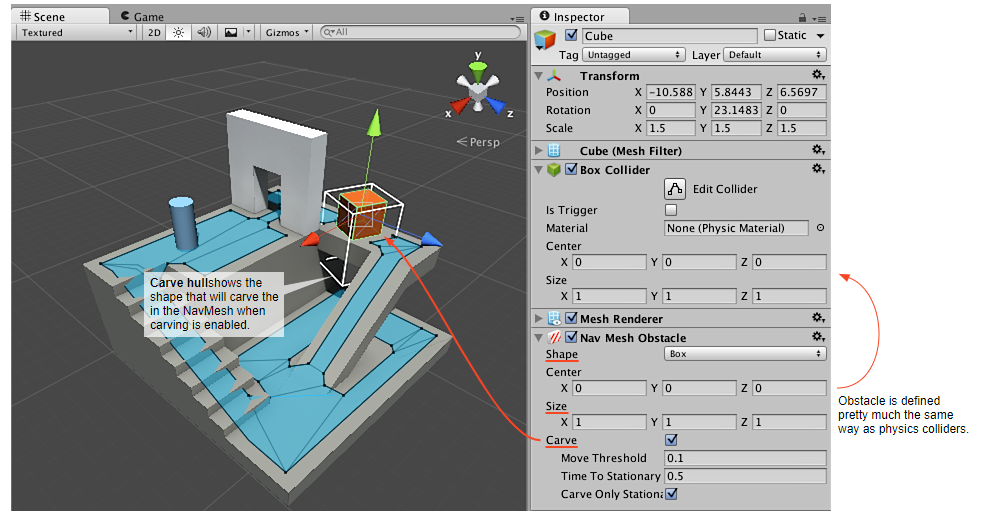
可使用场景中的任何游戏对象来容纳网格外链接组件，例如，围栏预制件可包含网格外链接组件。同样，可使用任何以变换作为开始和结束标记的游戏对象。

导航网格烘焙过程可以自动检测并创建常见的跳过和掉下链接。请参阅[自动构建网格外链接](file:///E:\UnityDocumentation\Manual\nav-BuildingOffMeshLinksAutomatically.html)以了解更多详细信息。

# 创建导航网格障碍物

导航网格障碍物 (NavMesh Obstacle) 组件可用于描述代理在导航时应避开的障碍物。例如，代理应该在移动时避开物理控制的物体，例如板条箱和木桶。

我们将添加一个板条箱来阻挡该关卡顶部的通道。



1.首先创建一个**立方体**来表现板条箱：\_\_Game Object > 3D Object > Cube\_\_。

2.将立方体移动到平台顶部，立方体的默认大小很适合板条箱，因此请保持原样。

3.将一个**导航网格障碍物组件**添加到立方体。从 Inspector 中选择 **Add Component**，然后选择 **Navigation > NavMesh Obstacle**。

4.将障碍物的形状设置为**盒体 (Box)**；更改形状将使中心和大小自动适应渲染网格。

5.将一个**刚体**添加到障碍物。从 Inspector 中选择 **Add Component**，然后选择 **Physics > Rigid Body**。

6.最后从 NavMesh Obstacle Inspector 中开启 **Carve** 设置，以便代理了解如何在障碍物周围寻路。

现在我们有一个物理控制的有效板条箱，并且 AI 了解如何在导航时避开。

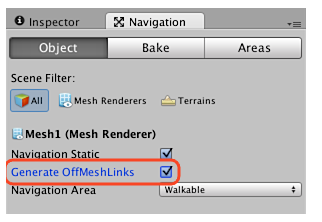
# 自动构建网格外链接

可自动检测网格外链接 (Off-Mesh Link) 的一些用例。两个最常见的用例为：\_掉下 (Drop-Down)\_ 和\_跳过 (Jump-Across)\_。

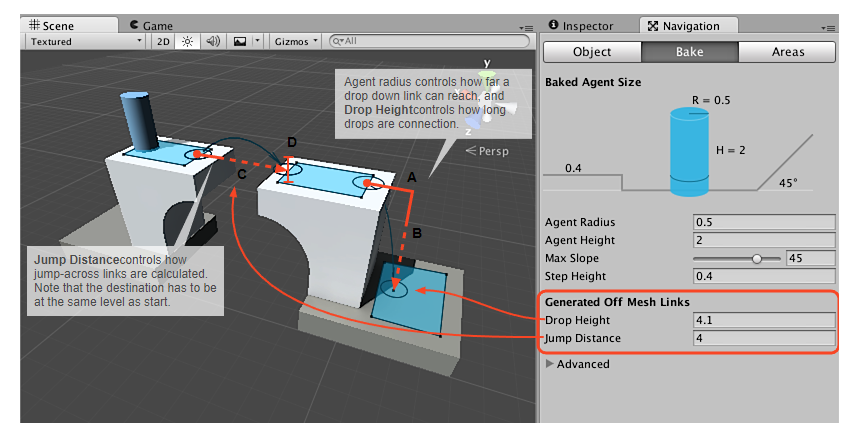
* **掉下**链接的作用是从平台上掉下。
* **跳过**链接的作用是跳过缝隙。

为了自动找到跳跃位置，构建过程沿着导航网格的边缘行走，并检查跳跃的着陆位置是否在导航网格上。如果跳跃轨迹畅通无阻，则会创建网格外链接。

让我们来设置网格外链接的自动生成。如果您不熟悉导航网格烘焙，请查看[构建导航网格](file:///E:\UnityDocumentation\Manual\nav-BuildingNavMesh.html)。



首先，需要标记场景中的跳跃\_起始位置\_对象。为此，需要在 Navigation 窗口\_中的 Objects\_ 选项卡下选中 **Generate Off-Mesh Links** 选项。



第二步是设置掉下和跳过轨迹：

* **掉下**链接的生成由 Drop Height 参数控制。该参数可控制将要连接的最高掉落高度，将值设置为 0 将禁用生成。
  + 定义掉下链接的轨迹时应使水平行程 **(A)** 为：\_2\*agentRadius + 4\*voxelSize。即，掉落将刚好落在平台的边缘之外。此外，垂直行程 ***(B)*** 需要大于烘焙设置的 Step Height\_（否则只会走下平台）并小于 Drop Height。应按体素大小进行调整，确保在体素化期间的任何舍入误差都不会阻止生成链接。应将 Drop Height 的值设置得比在关卡中测量的值略大，使链接正确连接。
* **跳过**链接的生成由 Jump Distance 参数控制。该参数可控制将要连接的最远距离。将值设置为 0 将禁用生成。
  + 定义跳过链接的轨迹时应使水平行程 **(C)** 大于 2\*agentRadius 且小于 Jump Distance。此外，着陆位置 **(D)** 不得超过距离起始位置关卡的 voxelSize。

现在已经标记对象并调整了设置，因此该按 Bake 按钮了，随即自动生成网格外链接！当您更改场景并烘焙时，旧链接将被丢弃，并将根据新场景创建新链接。

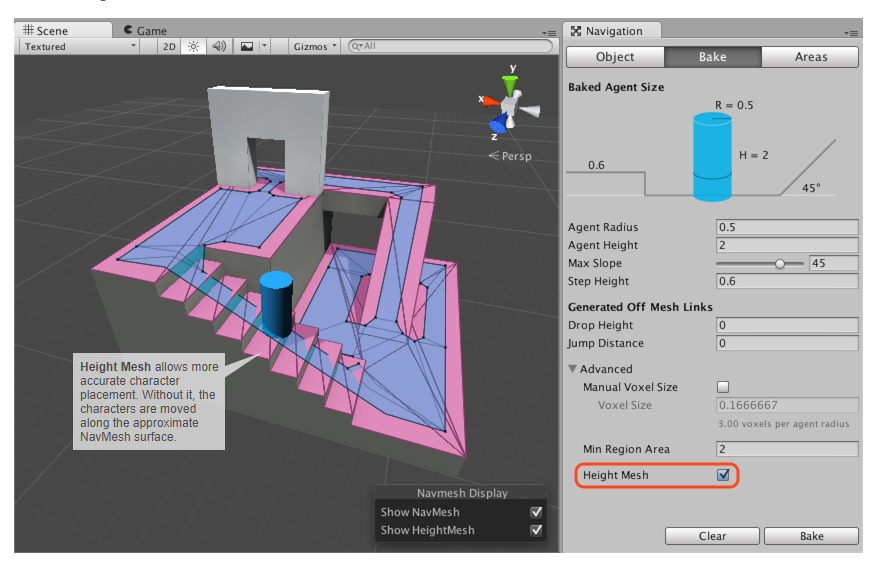
## 故障排除

如果未在所需位置生成网格外链接，请注意以下几点：

* Drop Height 应当比在关卡中测量的实际距离略大一点。这样可确保在导航网格烘焙过程中发生的小偏差不会阻止链接的连接。
* Jump Distance 应当比在关卡中测量的实际距离略远一点。Jump Distance 是从导航网格上的一个位置到导航网格上的另一个位置之间的测量值，这意味着应加上 \_2\*agentRadius\_（加一点）才能确保跨过缝隙。

# 构建高度网格以准确放置角色

高度网格 (Height Mesh) 可用于将角色更准确地放置在可行走表面上。



在导航时，导航网格代理 (NavMesh Agent) 被约束在导航网格的表面上。由于导航网格是可行走空间的近似形状，因此在构建导航网格时会使某些特征扁平化。例如，楼梯可能在导航网格中显示为斜坡。如果游戏需要准确放置代理，则应在烘焙导航网格时启用 Height Mesh 构建。可在 Navigation 窗口的 Advanced 设置下找到该设置。请注意，构建高度网格将在运行时占用内存和处理资源，并需要稍微更长一点的时间来烘焙导航网格。