**导航网格构建组件**

[导航网格](file:///E:\UnityDocumentation\Manual\nav-NavigationSystem.html)构建[组件](file:///E:\UnityDocumentation\Manual\UsingComponents.html)为您提供在运行时以及在 Unity Editor 中构建（也称为[烘焙](file:///E:\\UnityDocumentation\\Manual\\nav-BuildingNavMesh.html)）和使用导航网格的额外控制力。

从 [Unity 商店](https://store.unity.com/)下载的标准 Unity Editor 安装程序中未提供下列高级导航网格构建组件。请从 [Unity Technologies GitHub](https://github.com/Unity-Technologies/NavMeshComponents) 下载这些组件并单独安装。

导航网格有四个配套的高级组件：

* [导航网格表面 (NavMesh Surface)](file:///E:\UnityDocumentation\Manual\class-NavMeshSurface.html) - 用于为一种类型的代理构建和启用导航网格表面。
* [导航网格修改器 (NavMesh Modifier)](file:///E:\UnityDocumentation\Manual\class-NavMeshModifier.html) - 用于根据变换层级视图来影响导航网格区域类型的导航网格生成。
* [导航网格修改器体积 (NavMeshModifierVolume)](file:///E:\UnityDocumentation\Manual\class-NavMesh-ModifierVolume.html) - 用于根据体积来影响导航网格区域类型的导航网格生成。
* [导航网格链接 (NavMeshLink)](file:///E:\UnityDocumentation\Manual\class-NavMeshLink.html) - 用于为一种类型的代理连接相同或不同的导航网格表面。

另请参阅有关 [Mesh-BuildingComponents-API](file:///E:\UnityDocumentation\Manual\NavMesh-BuildingComponents-API.html) 的文档。

有关代理类型的更多信息，请参阅[创建导航网格代理](file:///E:\UnityDocumentation\Manual\nav-CreateNavMeshAgent.html)的相关文档。

有关导航网格区域类型的更多详细信息，请参阅[导航网格区域](file:///E:\\UnityDocumentation\\Manual\\nav-AreasAndCosts.html)的相关文档。

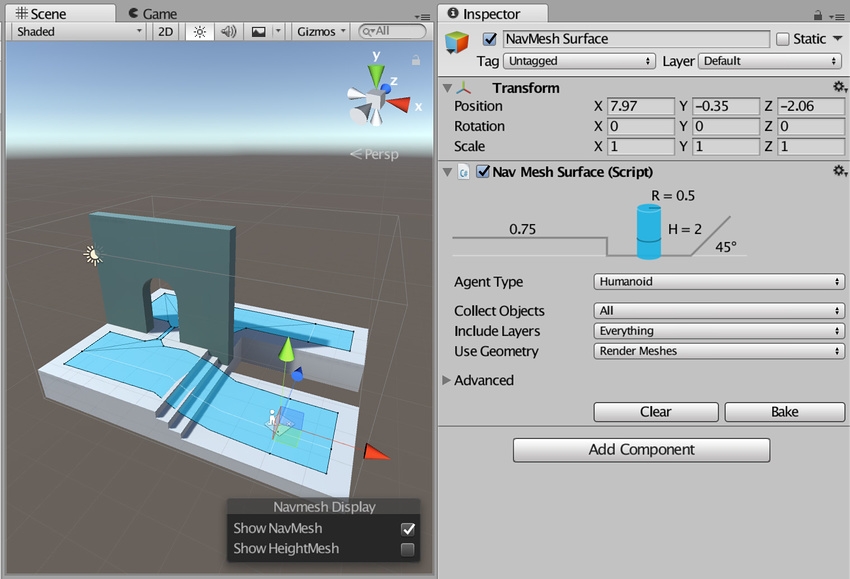
# 导航网格表面 (NavMesh Surface)

导航网格表面 (NavMesh Surface) 组件表示特定[导航网格代理 (NavMesh Agent)](file:///E:\UnityDocumentation\Manual\class-NavMeshAgent.html) 类型的可行走区域，并定义应构建导航网格的场景部分。

导航网格表面组件不在 Unity 标准安装中；有关如何访问该组件的信息，请参阅[高级导航网格构建组件](file:///E:\UnityDocumentation\Manual\NavMesh-BuildingComponents.html)的文档。

要使用导航网格表面组件，请导航至 **GameObject** > **AI** > **NavMesh Surface**。此操作会创建一个附加了导航网格表面组件的空游戏对象。一个场景可以包含多个导航网格表面。

您可以将导航网格表面组件添加到任何游戏对象。当您想要使用游戏对象管控[层级视图](file:///E:\UnityDocumentation\Manual\Hierarchy.html)来定义哪些游戏对象对导航网格有影响时，这非常有用。

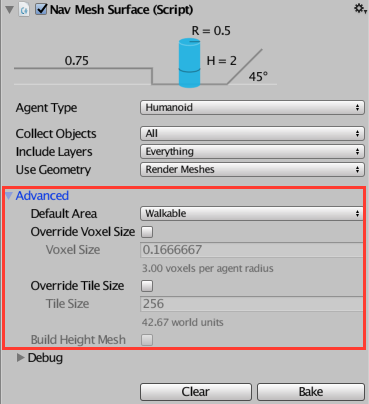
在 Inspector 窗口中打开的导航网格表面 (NavMesh Surface) 组件

|  |  |
| --- | --- |
| **属性** | **功能** |
| **Agent Type** | 使用导航网格表面的[导航网格代理](file:///E:\\UnityDocumentation\\Manual\\nav-CreateNavMeshAgent.html)类型。用于烘焙设置以及在寻路期间将导航网格代理与正确的表面相匹配。  - Humanoid - Ogre |
| **Collect Objects** | 定义要用于烘焙的游戏对象。 - **All** – 使用所有激活的游戏对象（这是默认选项）。 - **Volume** – 使用与包围体重叠的所有激活状态游戏对象。 - **Children** – 使用所有充当导航网格表面组件的子项的激活状态游戏对象。 |
| **Include Layers** | 定义要将哪些层上的游戏对象包含在[烘焙](file:///E:\\UnityDocumentation\\Manual\\nav-BuildingNavMesh.html)中。除了 **Collect Objects** 之外，此属性还允许从烘焙中进一步排除特定的游戏对象（例如，效果或动画角色）。 |
| **Use Geometry** | 选择用于烘焙的几何体。 - **Render Meshes** – 使用渲染网格和[地形](file:///E:\\UnityDocumentation\\Manual\\terrain-UsingTerrains.html)中的几何体。 – **Physics**[**Colliders**](file:///E:\UnityDocumentation\Manual\CollidersOverview.html) – 使用碰撞体和地形中的几何体。此选项与 **Render Meshes** 选项相比，代理可以移动到更接近环境物理边界的边缘。 |

使用导航网格表面组件的主要设置可以大范围过滤输入几何体。使用[导航网格修改器 (NavMesh Modifier) 组件](file:///E:\\UnityDocumentation\\Manual\\class-NavMeshModifier.html)可以微调 Unity 针对每个游戏对象处理输入几何体的方式。

烘焙过程会自动排除具有导航网格代理 (NavMesh Agent) 或导航网格障碍物 (NavMesh Obstacle) 的游戏对象。这些游戏对象是导航网格的动态用户，因此不参与导航网格构建。

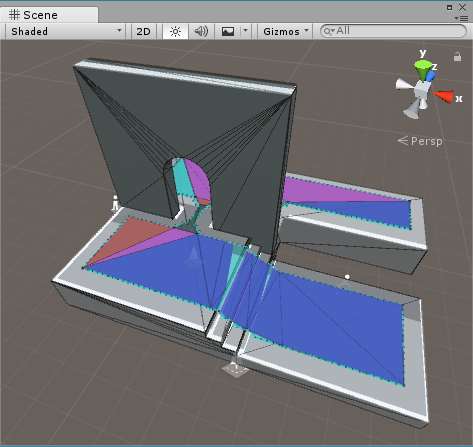
## Advanced 设置

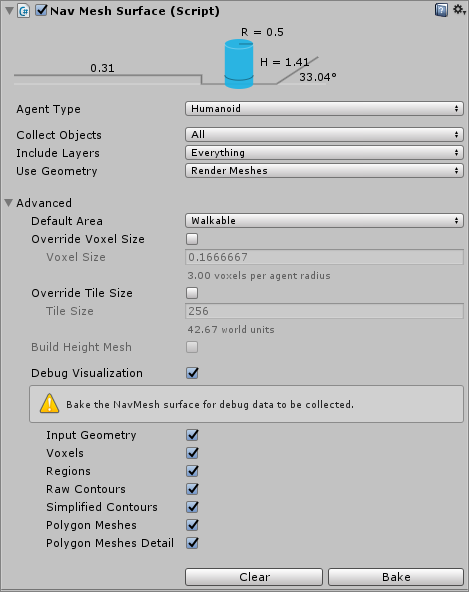
导航网格表面 (NavMesh Surface) 的 Advanced 设置面板

在 Advanced 设置部分可自定义以下附加参数：

|  |  |
| --- | --- |
| **属性** | **功能** |
| **Default Area** | 定义构建导航网格时生成的区域类型。  使用[导航网格修改器](file:///E:\\UnityDocumentation\\Manual\\class-NavMeshModifier.html)组件可以更详细地修改区域类型。 |
| **Override Voxel Size** | 控制 Unity 处理导航网格烘焙的输入几何体的准确程度（这是速度和准确度之间的权衡）。选中复选框可启用该属性。默认情况下会取消选中该复选框（禁用）。 每个代理半径 3 个体素（每个直径 6 个）允许捕获狭窄的通道（例如门），同时保持快速烘焙。对于大型开放区域，每个半径使用 1 或 2 个体素可加快烘焙速度。室内的密集点更适合较小的体素，例如每个半径 4 到 6 个体素。每个半径超过 8 个体素通常不会进一步带来好处。 |
| **Override Tile Size** | 为了使烘焙过程并行执行且提高内存效率，场景将分区进行烘焙。导航网格上可见的白线是区块边界。 默认区块大小为 256 个体素，这种大小在内存使用和导航网格碎片之间提供了良好的折衷。 要更改此默认区块大小，请选中此复选框，然后在 **Tile Size** 字段中输入您希望 Tile Size 包含的体素数。 区块越小，导航网格就越碎片化。  这种情况下有时会导致非最佳路径。导航网格雕刻也基于区块进行。如果有很多障碍物，通常可以通过减小区块大小（例如大约 64 到 128 个体素）来加速雕刻。如果您计划在运行时烘焙导航网格，请使用较小的区块大小以使最高内存使用率处于较低水平。 |
| **Build Height Mesh** | 不支持。 |

## 高级调试可视化



使用调试选项构建导航网格后显示的输入几何体、区域、多边形网格细节和原始轮廓

包含 Debug Visualization 选项的导航网格表面检视面板 (Nav Mesh Surface Inspector)

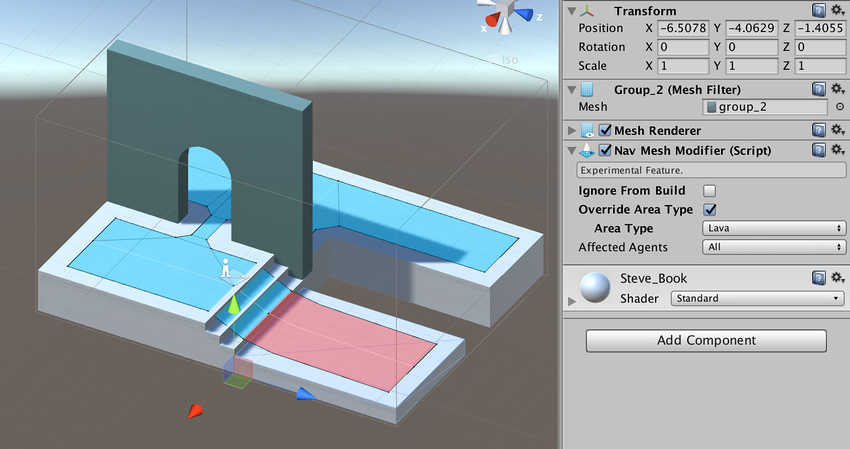
使用 **Debug Visualization** 部分中的设置可诊断导航网格构建过程中遇到的任何问题。不同的复选框显示了导航网格构建过程的每个步骤，包括输入场景体素化 (**Input Geometry**)、区域分割 (**Regions**)、轮廓生成 (**Contours**) 和导航网格多边形 (**Polygon Meshes**)。

# 导航网格修改器 (NavMesh Modifier)

导航网格修改器 (NavMesh Modifier) 可在运行时调整特定游戏对象在导航网格烘焙期间的行为方式。导航网格修改器不在 Unity 标准安装中；有关如何访问这些组件的信息，请参阅[高级导航网格构建组件](file:///E:\UnityDocumentation\Manual\NavMesh-BuildingComponents.html)的文档。

要使用导航网格修改器组件，请导航至 **GameObject** > **AI** > **NavMesh Modifier**。

在下图中，右下方的平台附加了一个修改器，用于将其 **Area Type** 设置为 **Lava**。

在 Inspector 窗口中打开的导航网格修改器 (NavMesh Modifier) 组件

导航网格修改器在整个层级视图上影响游戏对象，这意味着该组件附加到的游戏对象及其所有子项都受到影响。此外，如果在变换层级视图的下层找到另一个导航网格修改器，则新的导航网格修改器将覆盖层级视图上层的导航网格修改器。

导航网格修改器也会影响导航网格生成过程，这意味着必须更新导航网格以反映对导航网格修改器的任何更改。

| **属性：** | **功能：** |
| --- | --- |
| **Ignore From Build** | 选中此复选框可从构建过程中排除游戏对象及其所有子项。 |
| **Override Area Type** | 选中此复选框可更改包含修改器的游戏对象及其所有子项的区域类型。 |
| **Area Type** | 从下拉菜单中选择要应用的新区域类型。 |
| **Affected Agents** | 修改器影响的一系列代理 (Agent)。例如，可选择为特定代理排除某些障碍物。 |

# 导航网格修改器体积 (NavMesh Modifier Volume)

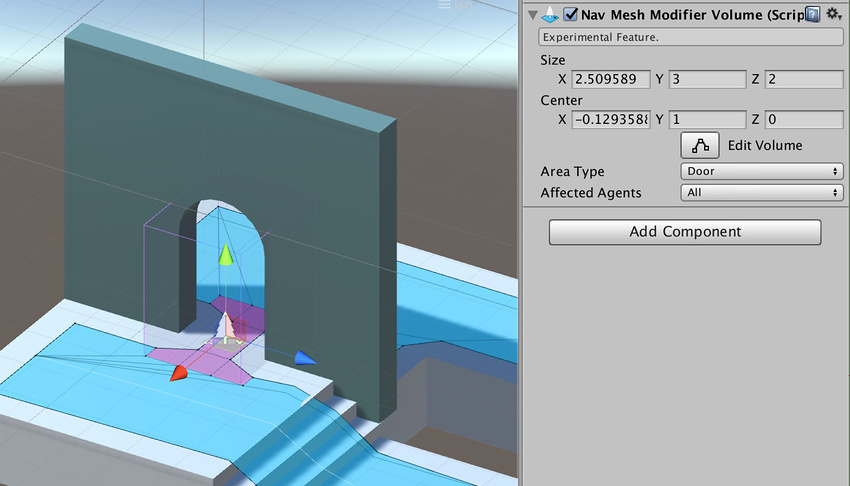
导航网格修改器体积 (NavMesh Modifier Volume) 组件不在 Unity 标准安装中；有关如何访问该组件的信息，请参阅[高级导航网格构建组件](file:///E:\UnityDocumentation\Manual\NavMesh-BuildingComponents.html)的文档。

导航网格修改器体积会将一个已定义的区域标记为特定类型（例如，\_\_Lava\_\_ 或 \_\_Door\_\_）。而[导航网格修改器](file:///E:\UnityDocumentation\Manual\class-NavMeshModifier.html)会将某些游戏对象标记为某种区域类型。导航网格修改器体积允许您在本地根据特定体积更改区域类型。

要使用导航网格修改器体积组件，请导航至 **GameObject** > **AI** > **NavMesh Modifier Volume**。

导航网格修改器体积用于标记不能表示为单独几何体的可行走表面的某些区域，例如危险区域。此组件也可用于使某些区域不可行走。

导航网格修改器体积也会影响导航网格生成过程，这意味着必须更新导航网格以反映对导航网格修改器体积的任何更改。

在 Inspector 窗口中打开的导航网格修改器体积 (NavMesh Modifier Volume) 组件

| **属性** | **功能** |
| --- | --- |
| **Size** | 导航网格修改器体积的尺寸，由 XYZ 测量值定义。 |
| **Center** | 相对于游戏对象中心的导航网格修改器体积的中心，由 XYZ 测量值定义。 |
| **Area Type** | 描述导航网格修改器体积适用的区域类型。 - **Walkable\_\_（这是默认选项） -**Not Walkable **-**Jump\_\_ |
| **Affected Agents** | 导航网格修改器体积影响的一系列代理 (Agent)。例如，可选择仅针对特定代理类型将选定的导航网格修改器体积标记为危险区域。 - **None** - **All\_\_（这是默认选项） -**Humanoid **-**Ogre\_\_ |

# 导航网格链接 (NavMesh Link)

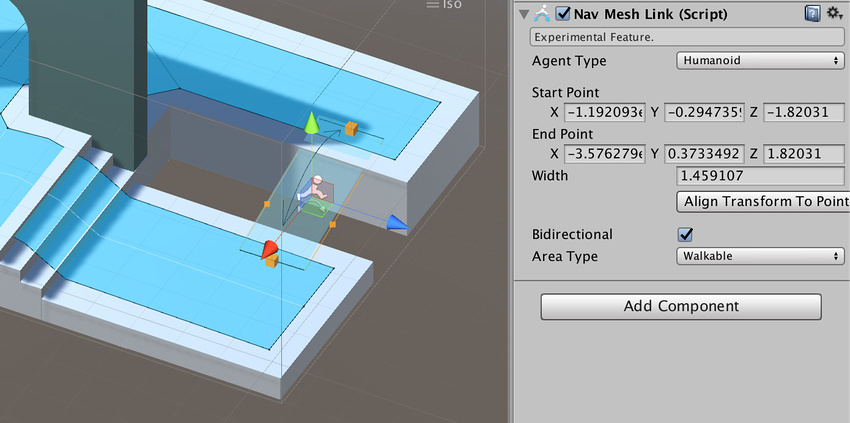
导航网格链接 (NavMesh Link) 组件不在 Unity 标准安装中；有关如何访问该组件的信息，请参阅[高级导航网格构建组件](file:///E:\UnityDocumentation\Manual\NavMesh-BuildingComponents.html)的文档。

导航网格链接在使用导航网格的两个位置之间创建可导航的链接。

此链接可以是点到点，也可以跨越间隙，在后一种情况下，代理 (Agent) 使用沿着入口边缘的最近位置来跨越链接。

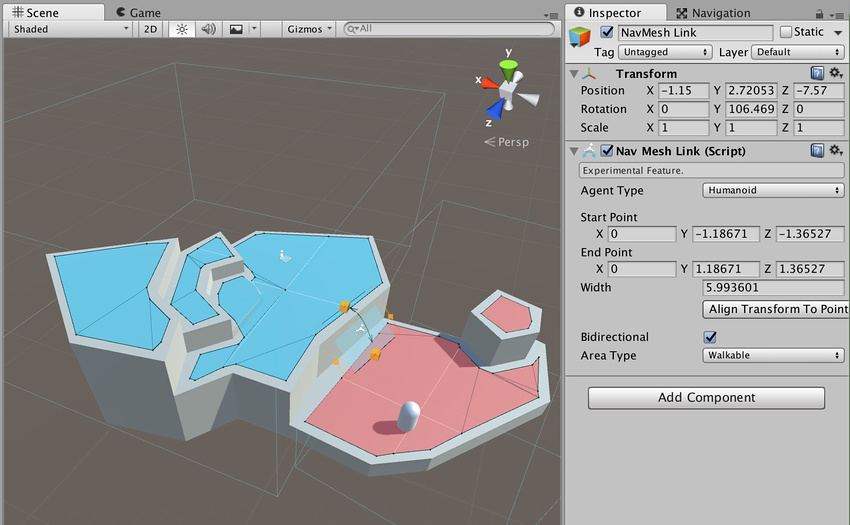
必须使用导航网格链接来连接不同的导航网格表面。

要使用导航网格链接组件，请导航至 **GameObject** > **AI** > **NavMesh Link**。

在 Inspector 窗口中打开的导航网格链接 (NavMesh Link) 组件

| **属性** | **功能** |
| --- | --- |
| **Agent Type** | 可使用该链接的代理类型。 - **Humanoid** - **Ogre** |
| **Start Point** | 链接起点（相对于游戏对象）。由 XYZ 测量值定义。 |
| **End Point** | 链接终点（相对于游戏对象）。由 XYZ 测量值定义。 |
| **Align Transform To Points** | 单击此按钮可将游戏对象移到链接的中心点，并将变换的前向轴与终点对齐。 |
| **Bidirectional** | 选中此复选框时，导航网格代理 (NavMesh Agent) 会双向遍历导航网格链接（从起点到终点，再从终点回到起点）。 取消选中此复选框时，导航网格链接仅单向运行（仅从起点到终点）。 |
| **Area Type** | 导航网格链接的区域类型（这会影响寻路成本）。 - **Walkable\_\_（这是默认选项） -**Not Walkable\_\_  - **Jump** |

## 将多个导航网格表面连接在一起

在此图像中，蓝色和红色导航网格在两个不同的导航网格表面中定义，并通过导航网格链接进行连接

如果希望代理在场景中的多个导航网格表面之间移动，必须使用导航网格链接连接它们。

在上面的示例场景中，蓝色和红色导航网格在不同的导航网格表面中定义，并使用导航网格链接连接它们。

注意在连接导航网格表面时：

* 可使用多个导航网格链接来连接导航网格表面。
* 导航网格表面和导航网格链接都必须具有相同的代理类型。
* 导航网格链接的起点和终点只能在一个导航网格表面上；如果在同一位置有多个导航网格，请务必小心。
* 如果要加载第二个导航网格表面，并在第一个场景中有未连接的导航网格链接，请确保它们未连接到任何不需要的导航网格表面。

# 导航网格构建组件 API

[导航网格](file:///E:\UnityDocumentation\Manual\nav-NavigationSystem.html)构建[组件](file:///E:\UnityDocumentation\Manual\UsingComponents.html)为您提供在运行时以及在 Unity Editor 中构建（也称为[烘焙](file:///E:\UnityDocumentation\Manual\nav-BuildingNavMesh.html)）和使用导航网格的额外控制力。

导航网格修改器不在 Unity 标准安装中；有关如何访问这些组件的信息，请参阅[高级导航网格构建组件](file:///E:\UnityDocumentation\Manual\NavMesh-BuildingComponents.html)的文档。

### 导航网格表面 (NavMeshSurface)

#### 属性

* agentTypeID – 需要构建导航网格的代理类型的 ID。
* collectObjects – 定义如何从场景收集输入几何体，为 UnityEngine.AI.CollectObjects 之一：
  + All – 使用场景中的所有对象。
  + Volume – 使用场景中与包围体接触的所有游戏对象（请参阅 size 和 center）
  + Children – 使用导航网格表面 (NavMesh Surface) 附加到的游戏对象的所有子对象。
* size – 构建体积的尺寸。该大小不受缩放影响。
* center – 构建体积的中心（相对于变换中心）。
* layerMask – 位掩码，用于定义必须将哪些层上的游戏对象包含在烘焙中。
* useGeometry – 定义用于烘焙的几何体，为 UnityEngine.NavMeshCollectGeometry 之一：
  + RenderMeshes – 使用渲染网格和地形中的几何体
  + PhysicsColliders – 使用碰撞体和地形中的几何体。
* defaultArea – 所有输入几何体的默认区域类型（除非另有说明）。
* ignoreNavMeshAgent – 如果具有导航网格代理 (Nav Mesh Agent) 组件的游戏对象应作为输入被忽略，则为 True。
* ignoreNavMeshObstacle – 如果具有导航网格障碍物 (Nav Mesh Obstacle) 组件的游戏对象应作为输入被忽略，则为 True。
* overrideTileSize – 如果设置了区块大小，则为 True。
* tileSize – 以体素为单位的区块大小（组件描述包含有关如何选择区块大小的信息）。
* overrideVoxelSize – 如果设置了体素大小，则为 True。
* voxelSize – 以世界单位表示的体素大小（组件描述包含有关如何选择区块大小的信息）。
* buildHeightMesh – 未实现。
* bakedNavMeshData – 对表面使用的 NavMeshData 的引用，如果未设置，则为 null。
* activeSurfaces – 所有激活状态的导航网格表面的列表。

\_\_注意：\_\_上述值会影响烘焙的结果，因此必须调用 Bake() 来包含它们。

#### 公共函数

* void Bake ()

根据导航网格表面上设置的参数烘焙新的 NavMeshData。可通过 bakedNavMeshData 访问该数据。

### 导航网格修改器 (NavMesh Modifier)

#### 属性

* overrideArea – 如果修改器覆盖区域类型，则为 True。
* area – 要应用的新区域类型。
* ignoreFromBuild – 如果包含修改器的游戏对象及其子项不应当用于导航网格烘焙，则为 True。
* activeModifiers – 所有激活状态的导航网格修改器的列表。

#### 公共函数

* bool AffectsAgentType(int agentTypeID)

如果修改器应用于指定的代理类型，则返回 true，否则返回 false。

### 导航网格修改器体积 (NavMesh Modifier Volume)

#### 属性

* size – 包围体的大小（采用局部空间单位）。变换会影响该大小。
* center – 包围体的中心（采用局部空间单位）。变换会影响该中心。
* area – 要应用于包围体内的导航网格区域的区域类型。

#### 公共函数

* bool AffectsAgentType(int agentTypeID)

如果修改器应用于指定的代理类型，则返回 true。

### 导航网格链接 (NavMesh Link)

#### 属性

* agentTypeID – 可使用该链接的代理类型。
* startPoint – 链接的起点（采用局部空间单位）。变换会影响该位置。
* endPoint – 链接的终点（采用局部空间单位）。变换会影响该位置。
* width – 链接的宽度（采用世界长度单位）。
* bidirectional – 如果为 true，则可以双向遍历链接。如果为 false，只能按照从起点到终点的方向遍历链接。
* autoUpdate – 如果为 true，则链接将更新端点以跟随每帧的游戏对象变换。
* area – 链接的区域类型（用于计算寻路成本）。

#### 公共函数

* void UpdateLink()

更新链接以便与关联的变换匹配。这对于更新链接很有用（例如在更改变换位置之后），但如果启用了 autoUpdate 属性，则不需要。但是，如果您很少更改链接变换，则调用 UpdateLink 可能对性能产生的影响要小得多。

# 官方案例使用的类说明

### 类的使用：

NavMeshSourceTag：将对象设置为NavMesh的原材料。

通过获取对象的MeshFilter和Terrain，将其转化为*NavMeshBuildSource*对象。

通过Collect函数将生成的*NavMeshBuildSource*对象列表传出去。

LocalNavMeshBuilder: 创建NavMesh。主要使用的是类NavMeshBuilder的两个函数：

同步函数：

# UpdateNavMeshData(*NavMeshData* data, *NavMeshBuildSettings* buildSettings, *List*<*NavMeshBuildSource*> sources, *Bounds* localBounds);

# 异步函数：

# UpdateNavMeshDataAsync(*NavMeshData* data, *NavMeshBuildSettings* buildSettings, *List*<*NavMeshBuildSource*> sources, *Bounds* localBounds);

# NavMeshData的使用：

# *NavMeshData* m\_NavMeshData = new *NavMeshData*(); //创建对象

# NavMesh类添加NavMeshData对象，并返回NavMeshDataInstance

# *NavMeshDataInstance* m\_Instance = *NavMesh*.*AddNavMeshData*(m\_NavMeshData);

NavMeshDataInstance可以清空NavMeshData对象

*NavMeshBuildSettings* 获取某一代理的设置类型：

var defaultBuildSettings = NavMesh.GetSettingsByID(0);

*List*<*NavMeshBuildSource*> sources 是*NavMeshBuildSource对象列表;*

*Bounds* localBounds 是NavMesh的范围：

new Bounds(*Vector3* center, *Vector3* size);

PS：画边界的函数:

void OnDrawGizmosSelected()

{

*Gizmos*.*color* = *Color*.*yellow*;

var bounds = new Bounds(*Vector3* center, *Vector3* size); *Gizmos*.*DrawWireCube*(bounds.center, bounds.size);

}