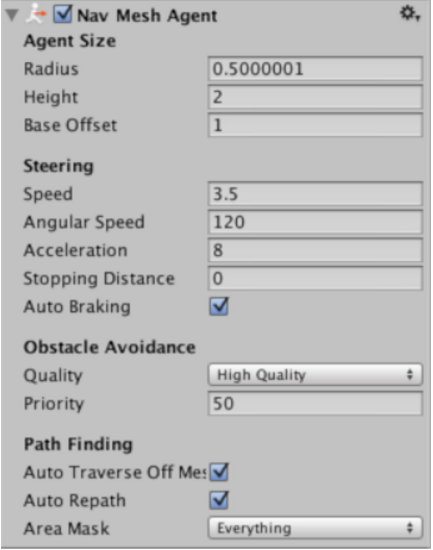
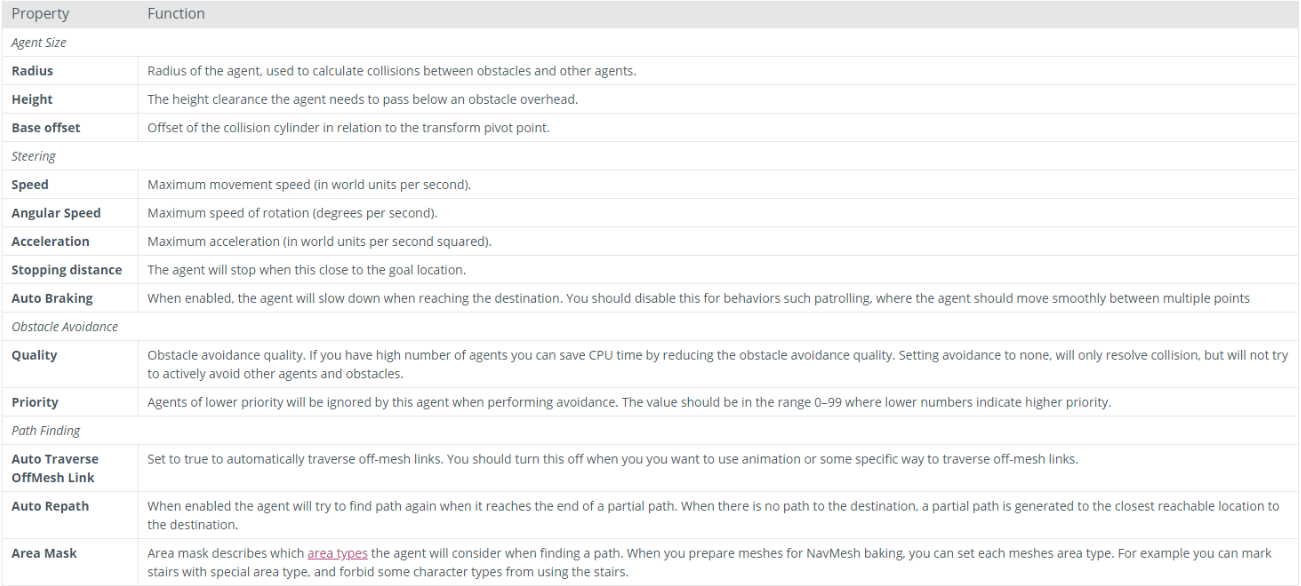
# Navigation2（补漏的章节）

# 1.NavMesh Agent(导航代理)

NavMeshAgent 组件帮你创建那些朝着目标法向移动的能够躲避多方的角色。在游戏世界里使用导航网格并且他们知道如何躲避对方以及其他的移动障碍物。寻路和空间推理通过NavMeshAgent的脚本API处理的。

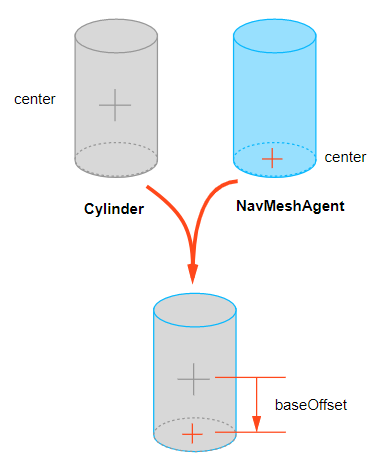
## 1.1属性



* *Agent Size(***代理尺寸***)*
  + Radius(半径):代理的半径，用来计算和其他代理以及障碍物之间的碰撞
  + Height(高度)：代理可通过的高度。（头顶上有障碍物）
  + Base offset(基础偏移)：碰撞相对于自身transform的偏移
* *Steering（***操控***）*
  + Speed(速度)：最大的移动速度（unity的世界单位/每秒）
  + Angular Speed（角速度）：最大旋转速度（度/每秒）
  + Acceleration（加速度）：最大的加速度（unity世界单位/秒的平方）
  + Stopping distance（停止距离）：代理将要靠近目标地点停下来距离
  + Auto BraKing（自动制动）：当选中时，代理会放慢速度到大目的地。应该不选中这个选项如果代理正在巡逻状态，因为代理此时需要在多个点之间平滑移动。
* *Obstacle Avoidance(***障碍规避***)*
  + Quality（质量）：规避质量。如果代理数量很多，可以通过减少规避质量来节约CPU时间。设置为none,只会解决碰撞，并不会解尝试主动规避其他代理和障碍物。
  + Priority（优先级）：执行规避时，低优先级代理将会被忽略。这个值应该在0-99之间，数值越小优先级越高。
* *Path Finding*(**寻路**)
  + Auto Traverse Off-Mesh Link(自动遍历 Off-Mesh Link):设置为true时代理将自动使用Off-Mesh Link 链接的线路。当你使用动画或者特殊的方式去遍历Off-Mesh Link 的路径时你因该关闭这个选项。
  + Auto Repath(自动从新寻路)：启用时当代理到达部分路劲的末尾时会去再次从新寻路。当没有路径到达目的地时，部分路径会被生成以到达接近目的地的位置。
  + Area Mask（区域遮罩）：其描述的是代理在巡路时会考虑的区域类型（area type）。当你准备进行网格烘焙时，你可以设定每个网格的区域类型。比如，你可以给台阶指定为特殊的区域类型，然后精致某些角色使用这些台阶。

# 1.2细节

* 代理圆柱体，其大小由“Radius”和“Height”属性决定。圆柱体随着物体一起移动，但即使物体本身更发生旋转圆柱体也会保持直立。圆柱体的形状用于检测和响应其他物体与障碍物之间的碰撞。如果GameObject的锚点不在圆柱体的底部，则可以使用“基础偏移”属性调节高度差。

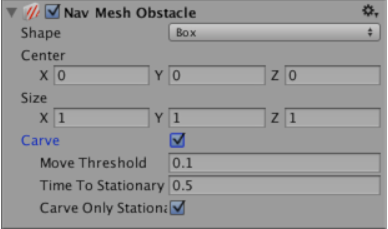


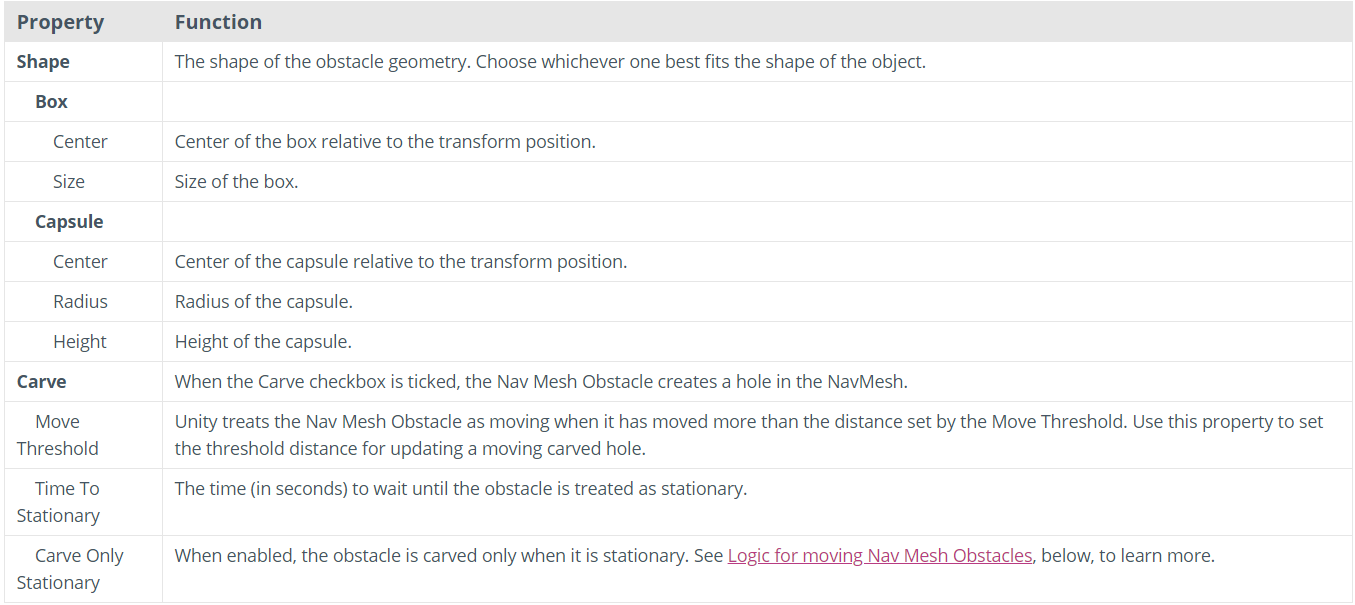
* 气缸的高度和半径实际上在两个不同的位置指定：NavMesh烘烤设置和各个代理的属性：
  + NavMesh烘焙设置描述了所有NavMesh代理如何碰撞或避免静态世界几何。为了保持内存预算和CPU负载的检查，在烘烤设置中只能指定一个尺寸。
  + NavMesh Agent的属性值描述的是代理如何与移动的障碍物和其他代理人发生冲突。
* 大多数情况下，您在两个地方都将代理人的大小设置为相同。 但是，例如，一个重兵可能有更大的半径，所以其他特工会在他周围留下更多的空间，否则他会避免环境相同。

# NavMesh Obstancle（导航障碍）

导航网格障碍组件允许您在导航世界时（例如，由物理系统控制的桶或板条箱）描述导航网格代理应避免的移动障碍物。 在障碍物移动的同时，导航网格代理尽力避免它。 当障碍物静止时，它在NavMesh上雕刻一个洞。 导航网格代理然后改变他们的路径来绕过它，或者如果障碍物导致路径被完全阻塞，则找到不同的路线。

## 2.1属性和功能





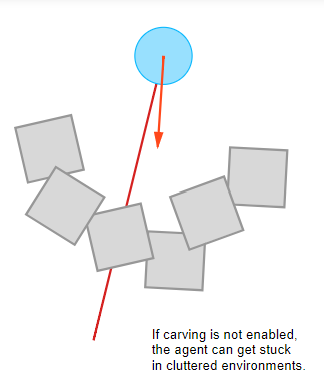
* + Shape :障碍几何形状。 选择最适合物体形状的一种。
    - Box:
      * Center:盒子的中心相对于Transform的位置。
      * Size:盒子的尺寸。
    - Capsule：
      * Center:胶囊的中心相对于变换位置。
      * Radius:胶囊的半径。
      * Height:胶囊的高度。
  + Carve:当勾选复选框时，导航网格障碍物在导航网格中创建一个洞。
    - Move Threshold:当移动超过移动阈值设置的距离时，Unity会将导航网格障碍物视为移动。 使用此属性设置更新移动雕刻孔的阈值距离。
    - Time To Stationary:等待障碍物视为静止的时间（以秒为单位）.
    - Carve Only Stationary:启用后，障碍物只有在静止时才被雕刻。 有关详细信息，请参阅下面的移动导航网格障碍的逻辑。

## 2.2 细节

## 导航网格障碍物可以通过两种方式影响导航期间导航网格代理的导航。

## Obstructing（阻碍）

* + 当Carve未启​​用时，Nav Mesh Obstacle的默认行为与Collider相似。 导航网格代理试图避免与导航网格障碍物碰撞，并且当它们靠近时，它们与导航网格障碍物碰撞。 避障行为是非常基本的，并且具有短的半径。 因此，导航网格代理可能无法在混乱导航网格障碍的环境中找到方法。 这种模式最适用于障碍物持续移动的情况下（例如车辆或玩家角色）。
* Carving(雕刻)
  + 启用Carve时，障碍物在静止时会在NavMesh中雕刻一个洞。 移动时，障碍物是障碍物。 当在NavMesh中刻出一个洞时，探路者能够在障碍物周围的位置周围导航Nav Mesh Agent，或者在当前路径被障碍物阻挡的情况下找到另一条路线。 开启雕刻导航网格障碍物是一种很好的做法，通常阻挡导航，但可以由玩家或其他游戏事件（如爆炸物（例如，箱子或桶））移动。



## 2.3用于移动导航网格障碍物的逻辑

* 当移动超过由Carve> Move Threshold设置的距离时，Unity会将Nav Mesh Obstacle视为移动。 导航网格障碍物移动时，雕刻孔也会移动。 但是，为了减少CPU开销，只在必要时重新计算孔。 此计算的结果可在下一帧更新中获得。 重新计算逻辑有两个选项：
  + 只有在导航网格障碍物静止时才进行雕刻
  + 当导航网格障碍物移动时雕刻

## 2.4只有在导航网格障碍物静止时才进行雕刻

* 这是默认行为。 要启用它，请勾选Nav Mesh Obstacle组件的Carve Only Stationary复选框。 在这种模式下，当导航网格障碍物移动时，雕刻的孔被移除。 当导航网格障碍物停止移动并且静止超过固定时间设置的时间时，它被视为静止并且雕刻的孔再次被更新。 在导航网格障碍物正在移动时，导航网格物体避免使用碰撞避免，但不要在其周围规划路径。
* 仅雕刻通常是性能方面的最佳选择，当与导航网格障碍物关联的游戏对象受物理控制时，它是一个很好的匹配。

## 2.5当导航网格障碍物移动时雕刻

* 要启用此模式，请取消导航网格障碍物组件的仅雕刻静止复选框。 如果未选中，当障碍物移动超过雕刻移动阈值设置的距离时，雕刻的孔会更新。 这种模式适用于缓慢移动的大型障碍物（例如，步兵正在躲避的坦克）。
* 注意：当使用NavMesh查询方法时，您应该考虑到在更改导航网格障碍物和更改导航网格物体之间存在一帧的延迟。

# Off\_Mesh Link(网格链接)

OffMeshLink组件允许您合并无法用步行表面表示的导航快捷方式。 例如，跳过沟渠或围栏，或者在穿过它之前打开门，都可以被描述为离网链接。

## 3.1功能和属性

* start:描述Off-Mesh Link开始位置的对象。
* end:描述离网连接终点位置的对象。
* cost override:如果值为正值，则在处理路径请求时计算路径开销时使用它。 否则，使用默认成本（该游戏对象所属区域的成本）。 如果“成本覆盖”设置为值3.0，则移动离网链接将比在默认NavMesh区域上移动相同距离贵3倍。 当你想让代理人一般都喜欢步行时，成本覆盖变得很有用，但当步行距离明显更长时，使用离网连接。
* bi-directional:如果启用，链接可以在任一方向移动。 否则，它只能从开始到结束。
* activated:指定这个链接是否会被探路者使用（如果这个设置为false，它将被忽略）。
* auto update position:启用后，当端点移动时，关闭网格链接将重新连接到NavMesh。 如果禁用，即使端点移动，链接仍将停留在起始位置。
* navigation area:描述链接的导航区域类型。 区域类型允许您对相似区域类型应用公共遍历成本，并防止某些角色根据代理的区域掩码访问离网连接。

## 3.2详细信息

* 如果代理未遍历离线网格链接，请确保两个端点连接正确。 正确连接的终点应在接入点周围显示一个圆圈。
* 另一个常见原因是Navmesh Agent的面罩没有包含关闭网格链接的区域。