**导航网格代理与其他组件结合使用**

您也可以将导航网格代理 (NavMesh Agent)、导航网格障碍物 (NavMesh Obstacle) 和网格外链接 (Off Mesh Link) 组件与其他 Unity 组件一起使用。此处列出了混用不同组件时的一些注意事项。

**导航网格代理和物理组件**

1. 无需向导航网格代理添加物理碰撞体来让它们彼此避开
   * 也就是说，导航系统会模拟代理及其对障碍物和静态世界的反应。此处所说的静态世界是指烘焙的导航网格。
2. 如果希望导航网格代理推开物理对象或使用物理触发器：
   * 添加碰撞体 (Collider) 组件（如果不存在）
   * 添加刚体 (Rigidbody) 组件
     + 启用运动学 (Is Kinematic) - 这很重要！
     + 运动学意味着刚体由物理模拟以外的其他事物控制
3. 如果导航网格代理和刚体（非运动学）同时处于激活状态，表示存在竞争条件
   * 两个组件都可能尝试移动相同位置的代理，从而导致不明行为
4. 可使用导航网格代理移动玩家角色之类的代理，无需使用物理组件
   * 将玩家代理的躲避优先级设置为较小数字（高优先级），从而允许玩家穿过群体
   * 使用 [NavMeshAgent.velocity](file:///E:\\UnityDocumentation\\ScriptReference\\AI.NavMeshAgent-velocity.html) 移动玩家代理，使其他代理能够预测玩家的移动以避开玩家。

**导航网格代理和动画器**

1. 导航网格代理和带有根运动的动画器可能会导致竞争条件
   * 两个组件都尝试在每帧移动变换
   * 两种可能的解决方案
2. 信息应始终朝一个方向流动
   * 要么由代理移动角色并使动画跟随
   * 要么根据模拟结果由动画移动角色
   * 否则，最终将发生难以调试的反馈循环
3. *动画跟随代理*
   * 使用 [NavMeshAgent.velocity](file:///E:\\UnityDocumentation\\ScriptReference\\AI.NavMeshAgent-velocity.html) 作为动画器的输入，从而将代理的移动大致匹配成动画
   * 强大且易于实现，将导致脚滑（此情况下动画无法与速度匹配）
4. *代理跟随动画*
   * 禁用 [NavMeshAgent.updatePosition](file:///E:\\UnityDocumentation\\ScriptReference\\AI.NavMeshAgent-updatePosition.html) 和 [NavMeshAgent.updateRotation](file:///E:\\UnityDocumentation\\ScriptReference\\AI.NavMeshAgent-updateRotation.html) 以从游戏对象位置解除模拟
   * 使用模拟代理的位置 ([NavMeshAgent.nextPosition](file:///E:\\UnityDocumentation\\ScriptReference\\AI.NavMeshAgent-nextPosition.html)) 和动画根 ([Animator.rootPosition](file:///E:\\UnityDocumentation\\ScriptReference\\Animator-rootPosition.html)) 之间的差异来计算动画的控制
   * 有关更多详细信息，请参阅[耦合动画和导航](file:///E:\UnityDocumentation\Manual\nav-CouplingAnimationAndNavigation.html)

**导航网格代理和导航网格障碍物**

* 不要混用！
  + 同时启用两者将使代理尝试避开自己
  + 如果此外还启用雕刻，则代理会尝试不断重新映射到雕刻孔的边缘，甚至会伴随出现更多错误行为
* 确保在任何给定时间只有其中一个为激活状态
  + 死亡状态，可关闭代理并开启障碍物以迫使其他代理避开它
  + 或者，可使用优先级来更多地避开某些代理

**导航网格障碍物和物理组件**

* 如果希望物理控制的对象影响导航网格代理的行为
  + 将导航网格障碍物组件添加到代理应该知道的对象，这允许避让系统推断障碍物
* 如果游戏对象附加了刚体和导航网格障碍物，则自动从刚体获得障碍物的速度
  + 这可让导航网格代理预测并避开移动的障碍物