始转弯来证明它是合理的。

我们还可以扩展最终请求，使其与初始请求的大小相同。这可以确保角色不会因为它的请求被过滤而移动得太慢。

但是在图 3.69 中，过滤的问题则变得很明显。现在没有可以由汽车执行的请求的组件。单独过滤将使汽车保持不动，直至目标移动或直到计算中的数字错误解决了死锁。

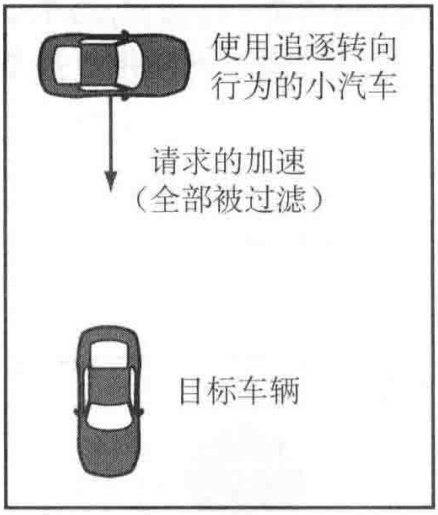

图 3.69 任何加速请求都将被执行器过滤

为了解决这种情况，我们可以检测最终结果是否为零并采用不同的执行方法。这可能是一个完整的解决方案，如下面的与能力匹配的转向技术，也可能是一个简单的启发式方法，如前进和强行转弯。

根据我们的经验，大多数情况可以基于过滤的执行方式简单地解决问题。之所以有不起作用的地方是因为转向请求中存在少量误差范围。为了高速行驶，在狭窄的空间内操纵，在动画中匹配动作或跳跃，都需要尽可能地尊重转向请求。在这种情况下，过滤可能会导致问题，但是，公允地说，本节中的其他方法也是如此（尽管程度可能较小）。

3.8.2 与能力匹配的转向

所谓执行的不同方法就是将执行带入转向行为本身。AI 不仅根据角色想要去的地方生成移动请求，还要考虑角色的物理能力。

如果角色正在追击敌人，那么它将考虑自己可以实现的每个动作，并选择能以最佳方式捕捉到目标的方法。如果可以执行的一组策略相对较小（例如，可以向前移动或左转或右转），那么可以简单地依次查看每个策略并确定策略完成后的情况，以确保最终选择执行的动作是能引发最佳状态的动作。例如，对于追击敌人而言，最佳状态无疑就是角色距离其目标最近。