射击战场

主要内容

- 游戏内容, 玩法简介
- 用到的技术
- 遇到的困难和解决方法
- 可以做得更好的地方

玩法简介

- 第三人称射击游戏
  - 玩家可进行移动, 瞄准, 射击
  - 可以拾取和切换不同的枪械
- 敌人为 AI 控制的角色
  - 不同的敌人会有不同的行为和特性
  - 敌人和玩家均有血量设定, 受到攻击后血量减少
  - 玩家血量小于 0 后游戏失败, 敌人血量小于 0 后被击败
- 击败关卡中所有敌人为通关
  - 关卡中有不同种类的敌人
  - 关卡中会随机刷新 buff, 玩家拾取获得血量/子弹增益

人物动画

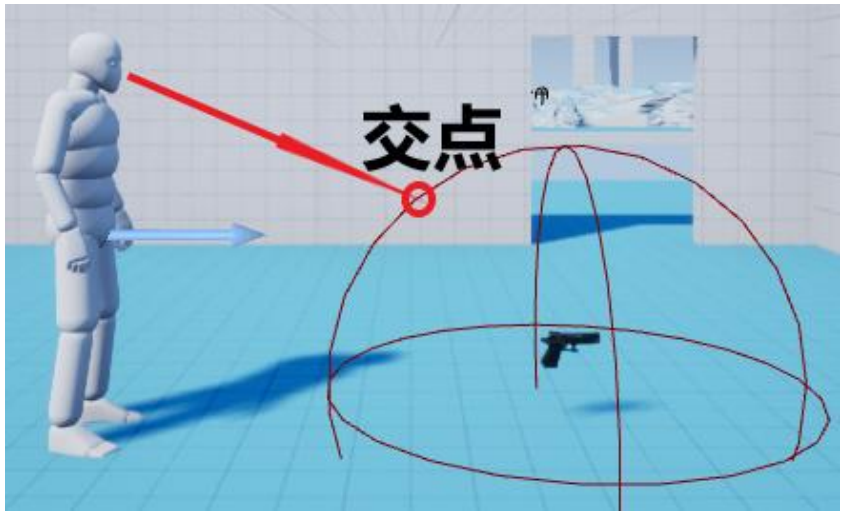
- 角色的基础移动
  - 使用动画蓝图中动画状态机实现
  - 定义好站立, 走动, 跑动, 跳等状态, 由角色的数据驱动状态转移

- 角色的俯仰与瞄准
  - 使用瞄准偏移实现
  - 不同瞄准角度的姿势构成的混合而成
  - 通过玩家的偏移角度确定使用哪个混合姿势

- 角色的特殊动作 (攻击, 换弹)
  - 播放动画蒙太奇
  - 使用分层动画

物体的拾取

- 物体自带包围球
- 从玩家视线方向做射线检测
  - 得到与射线相交的物体
  - 如果物体可拾取, 则拾取
  - 武器和 buff 的拾取都是这样实现




```
// 从视线方向做射线，与场景物体求交
if (GetWorld()->LineTraceSingleByChannel(HitResult, StartPostion, EndPosition, ECC_Visibility))
{
    // 相交的物体为可拾取的物体
    if (HitResult.GetActor()->IsA(ABaseObject::StaticClass()))
    {
        ABaseObject* PickUpObject = Cast<ABaseObject>(HitResult.GetActor());
        PickUp(PickUpObject);
    }
}
```

武器实现:射线枪械

- 不产生真实的子弹和弹道
- 从枪口方向出发做射线检测, 与场景内物体求交
  - 返回的交点就是子弹击中的点, 在该点上模拟子弹击中/爆炸的特效
  - 如果击中物体为敌人/玩家的角色, 则扣除角色血量

```
if (GetWorld()->LineTraceSingleByChannel(HitResult, StartLocation, EndLocation, ECC_Camera))
{
    // 释放子弹击中爆炸的特效
    SimulateHitAtLocation(HitResult.Location);
    if (HitResult.GetActor()->IsA(ASBaseCharacter::StaticClass()))
    {
        // 子弹击中了角色，造成伤害
        FTakeHitInfo HitInfo;
        // 初始化受击参数
        // ...
        ASBaseCharacter* HitCharacter = Cast<ASBaseCharacter>(HitResult.GetActor());
        HitCharacter->TakeHit(HitInfo);
    }
}
```

Pros: 实现简单, 即时命中

Cons: 某些情况下命中效果不正确

武器实现: 抛体枪械

- 模拟真实的子弹和弹道
- 枪械开火时, 在枪口方向发射出一个带初速度的小球
  - 当小球和场景中的物体发生碰撞时, 子弹击中了物体
  - 在碰撞位置模拟击中/爆炸的效果

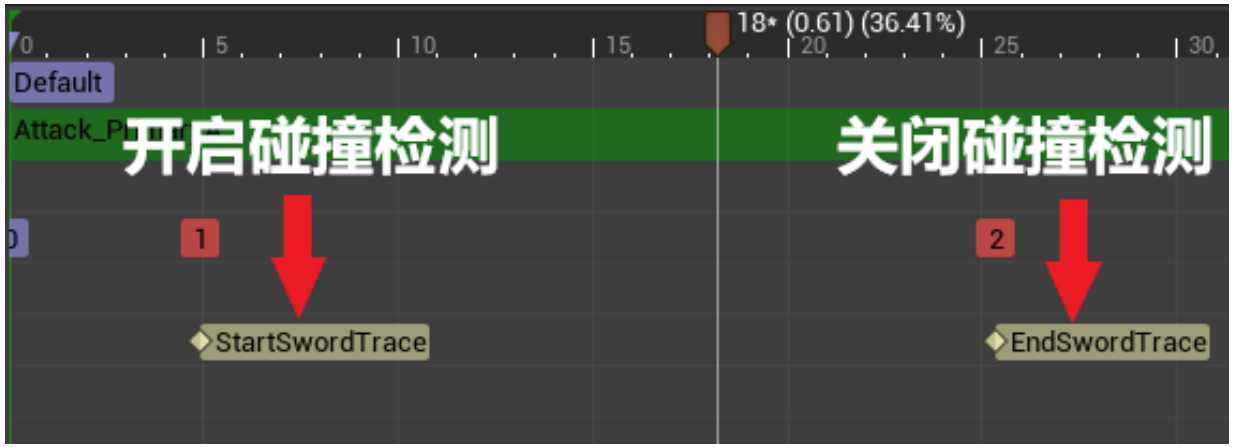
```
void ASRifleBullet::OnBulletImpact(UPrimitiveComponent* HitComponent, AActor* OtherActor
{
    if (OtherActor->IsA(ASBaseCharacter::StaticClass()))
    {
        ASBaseCharacter* HitCharacter = Cast<ASBaseCharacter>(OtherActor);
        FTakeHitInfo HitInfo;
        // 初始化受击参数
        // ...
        HitCharacter->TakeHit(HitInfo);
    }
    // 释放子弹爆炸的特效
    UGameplayStatics::SpawnEmitterAtLocation(this, HitEffect, GetActorLocation(), FRotat
    // 销毁子弹自身
    Destroy();
}
```

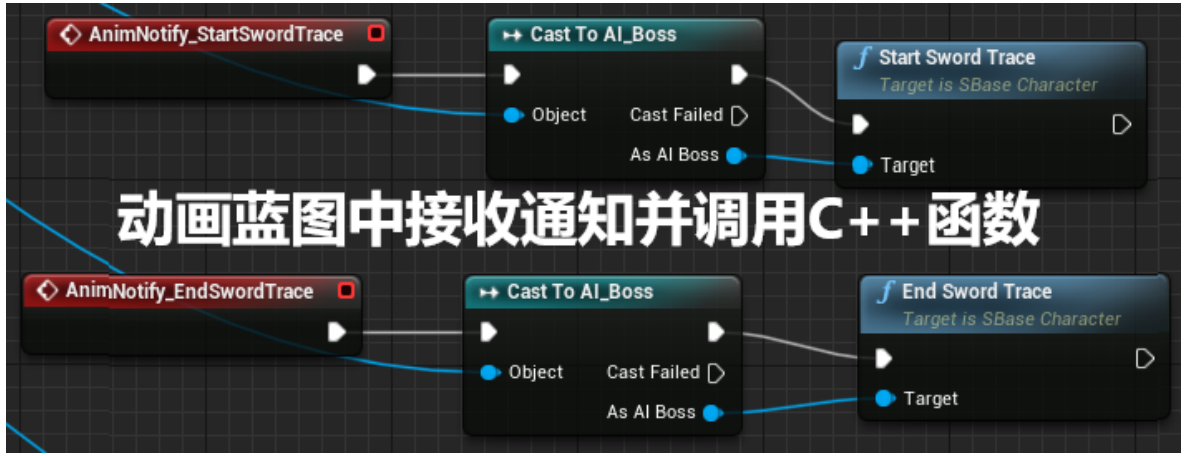
Pros: 真实的效果

Cons: 实现较复杂

武器实现: 近战武器

- 攻击动作由动画蒙太奇释放
- 在攻击动作的有效帧数内, 开启武器的碰撞检测
  - 通过 animation notify 进行碰撞检测的开/关
  - 如果武器和角色发生碰撞, 那么判定为击中
  - 为了避免一次攻击动作被多次判定, 维护单次动作中被击中角色的列表





敌人 AI 的设计

- 黑板 Blackboards
  - 记录必要的变量
- 行为树 Behavior Trees
  - 决定 AI 在当前时刻该做什么
- 行为树任务 Behavior Tree Tasks
  - 执行具体的任务

哨兵

-站岗

-攻击/掩护

-逃离

巡逻兵

-巡逻

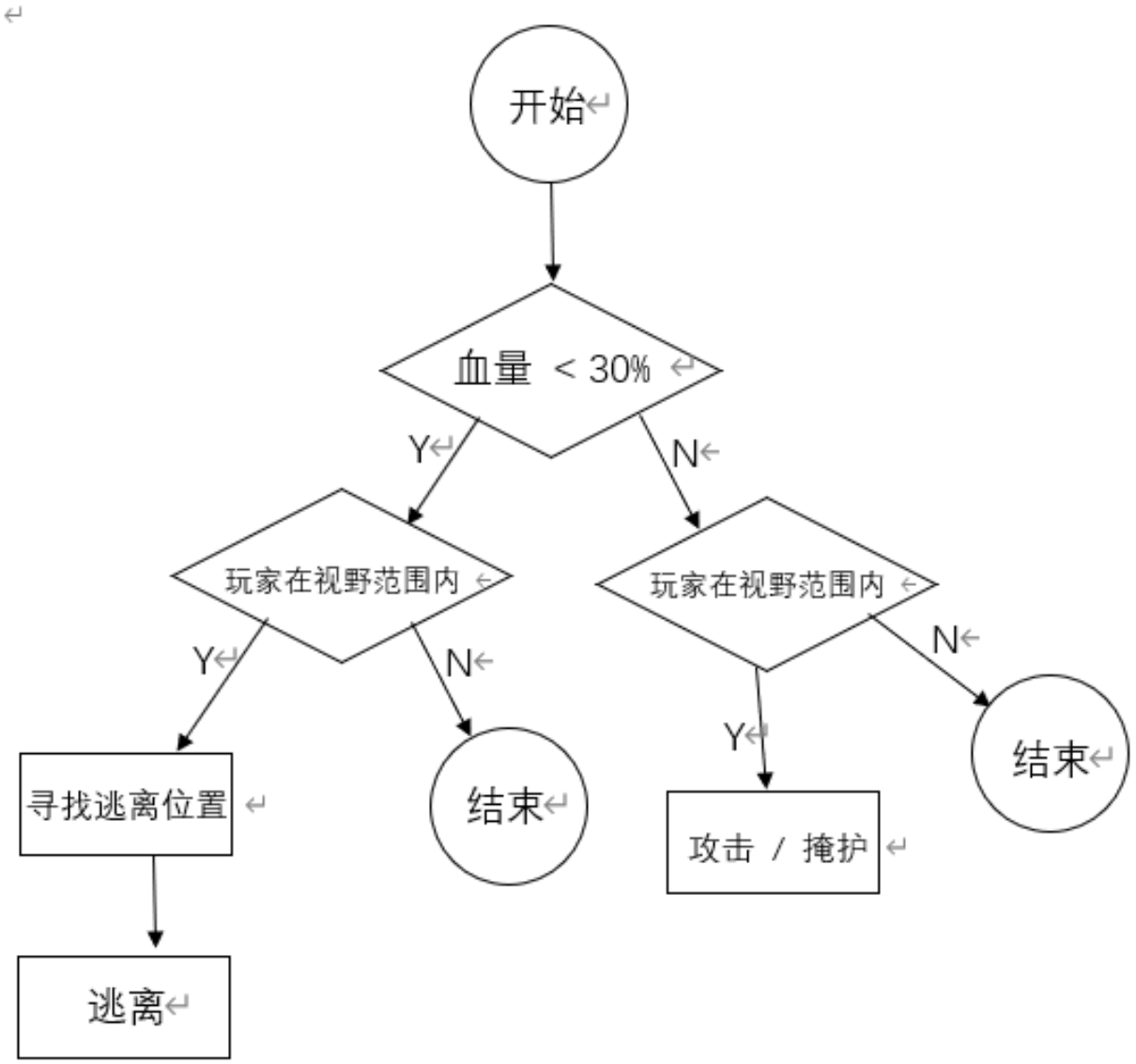
-攻击

Boss

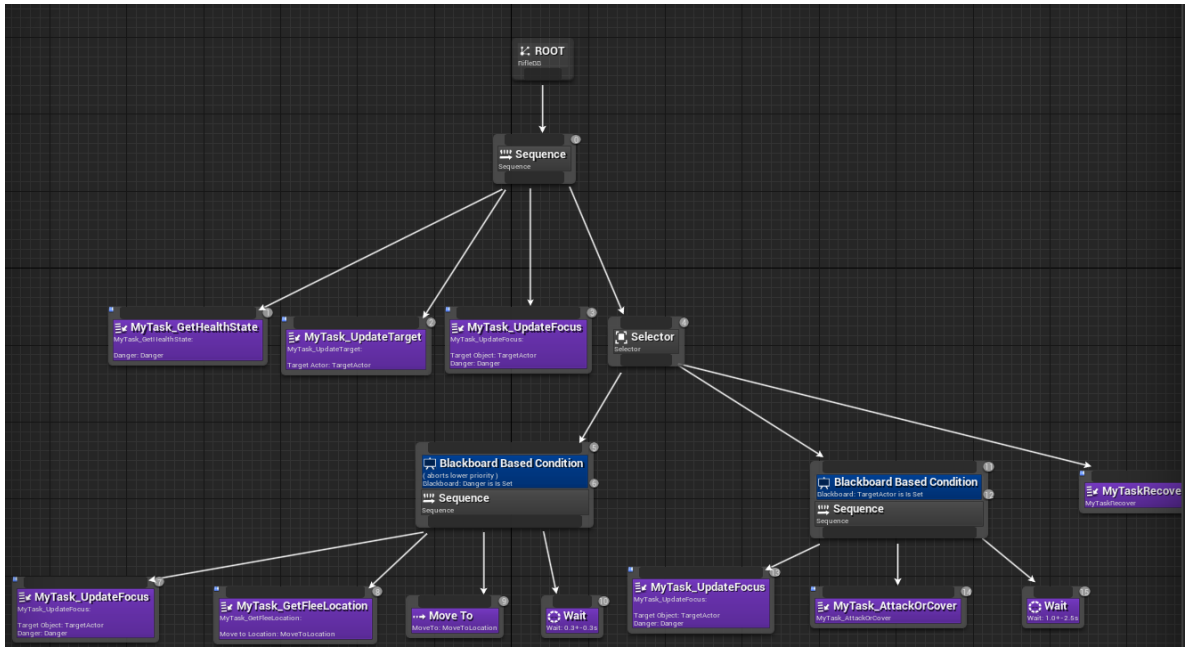
-寻敌

-攻击

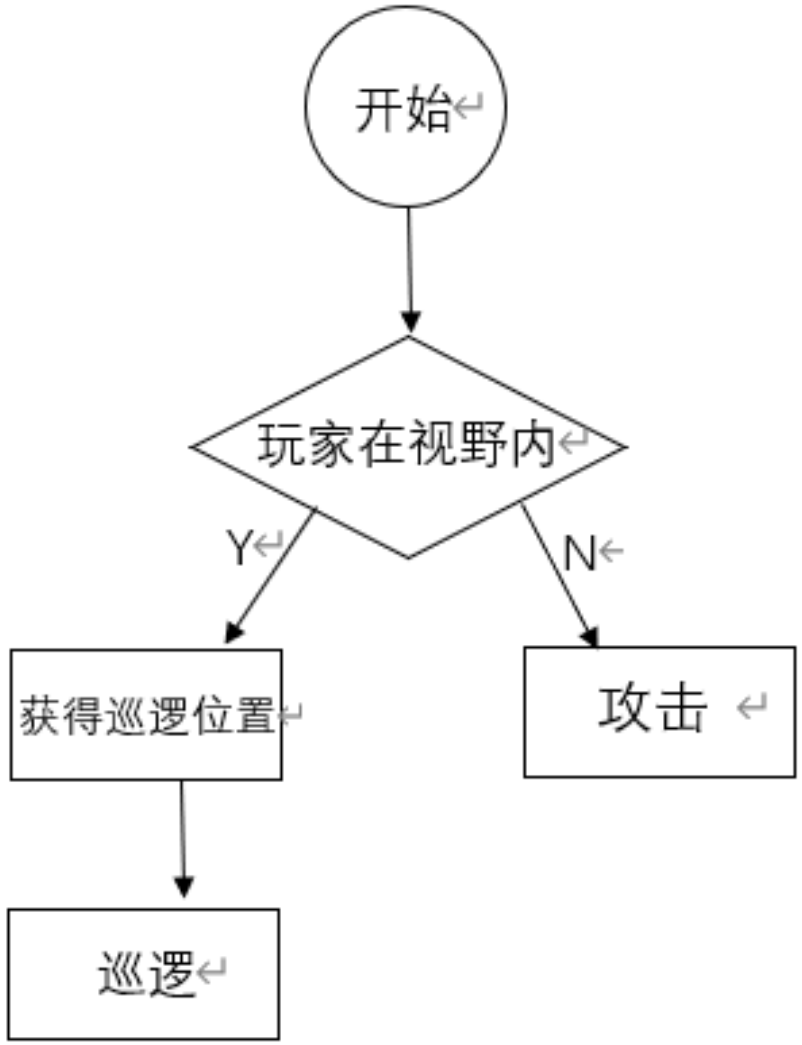
敌人 AI 的设计: 哨兵



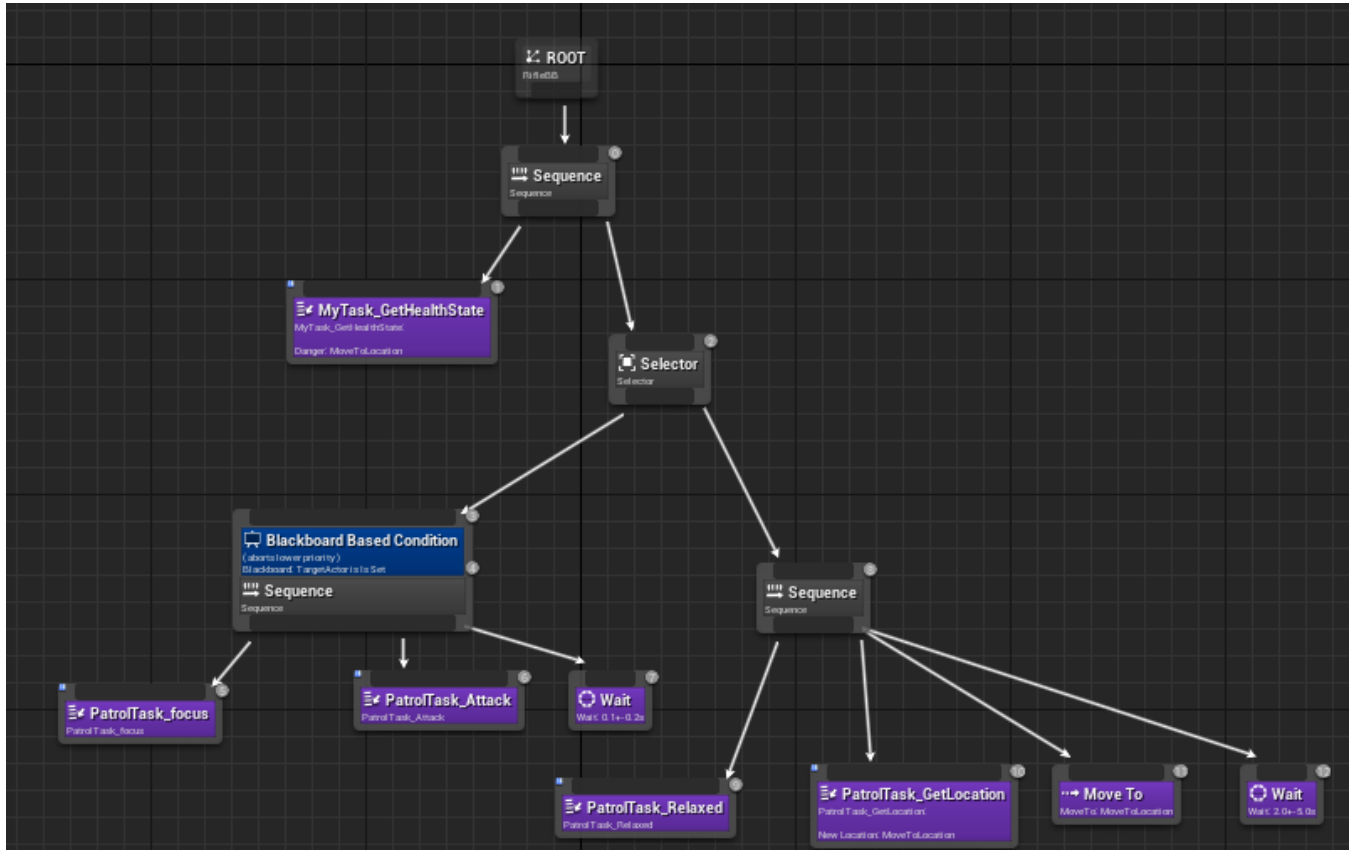
- 血量小于 %30 且玩家在视野内时, 逃离
- 玩家在视野内, 进行攻击 / 掩护
- 若都不满足, 进入站岗状态



敌人 AI 的设计: 巡逻兵

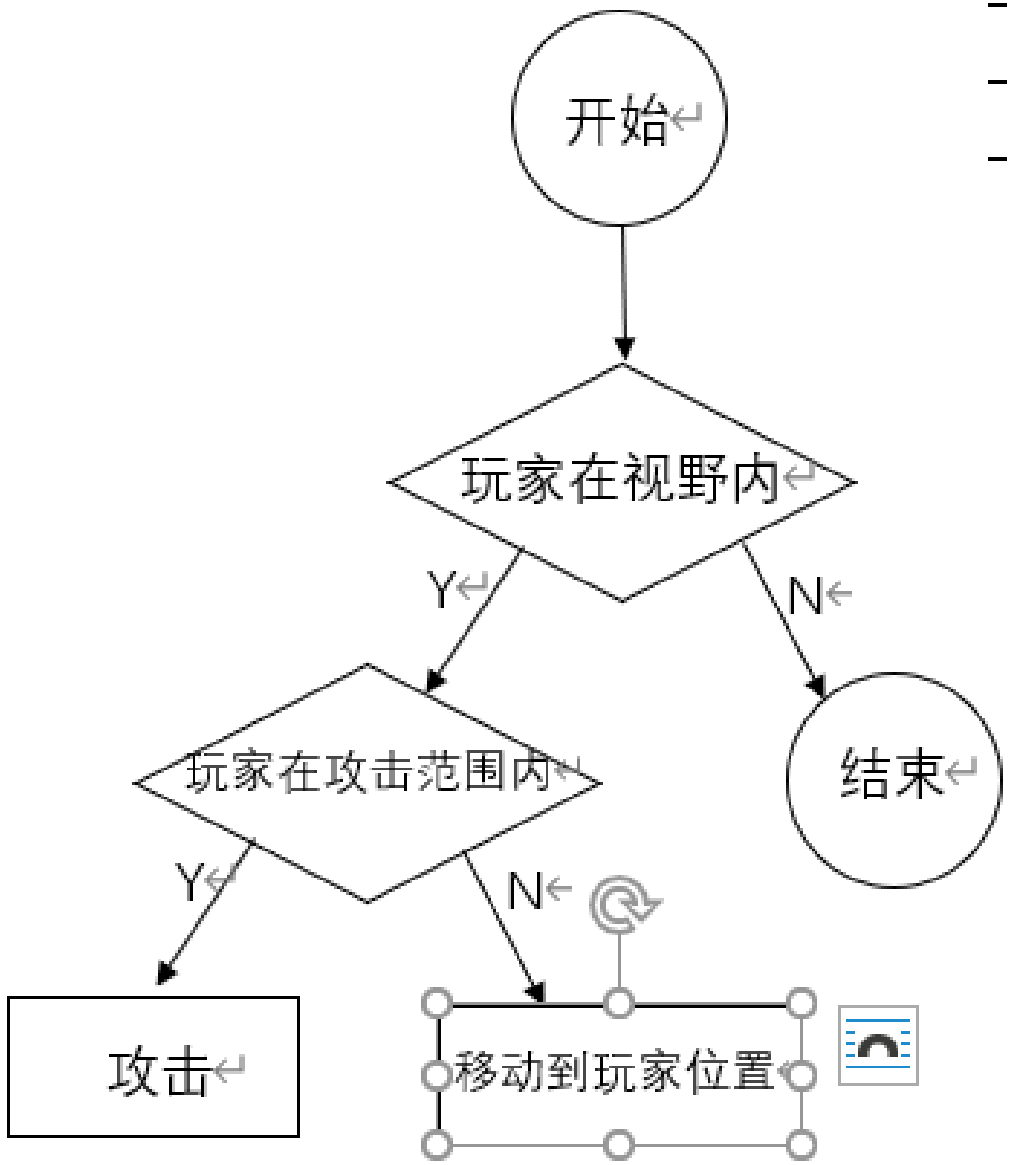


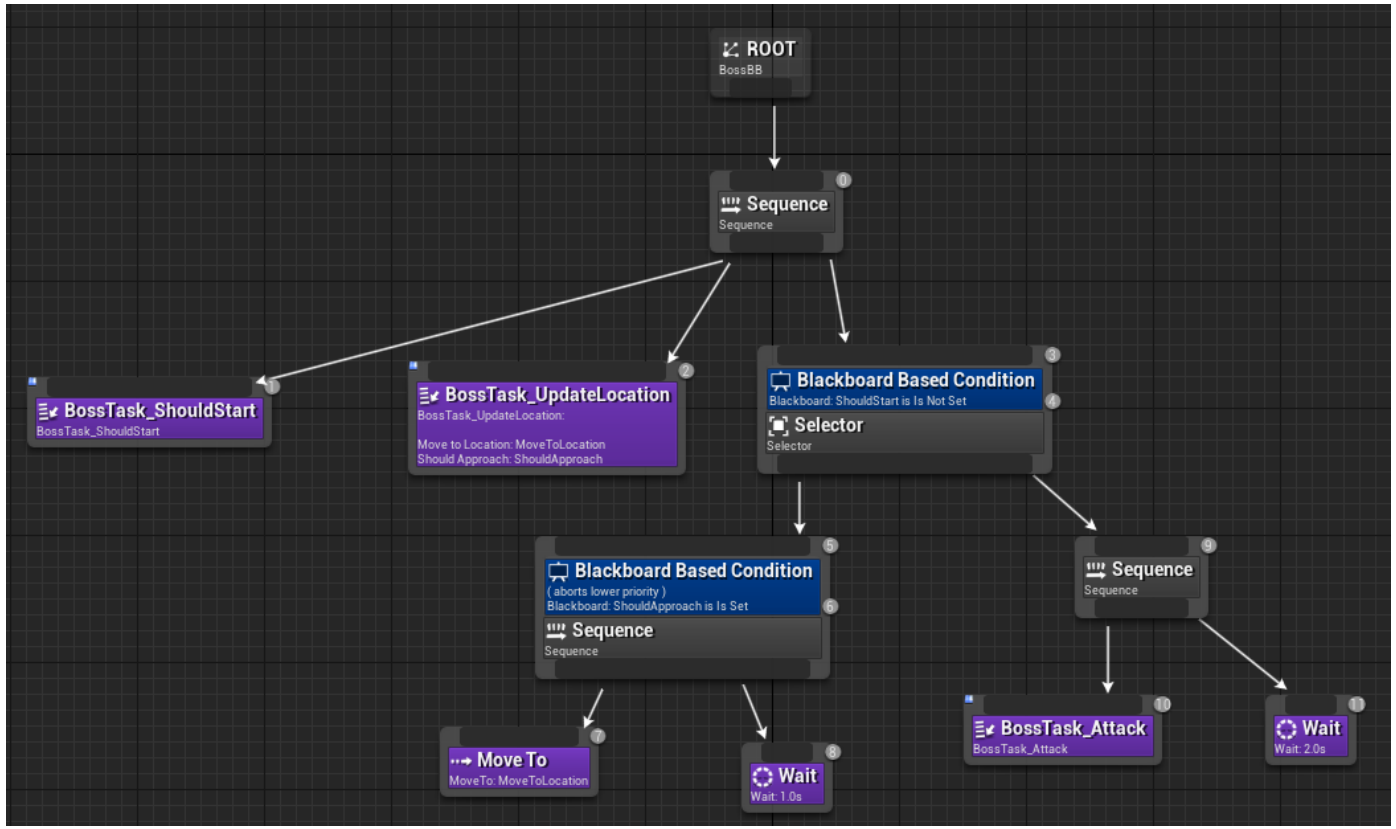
- 玩家在视野内, 攻击
- 否则进行巡逻



敌人 AI 的设计: Boss

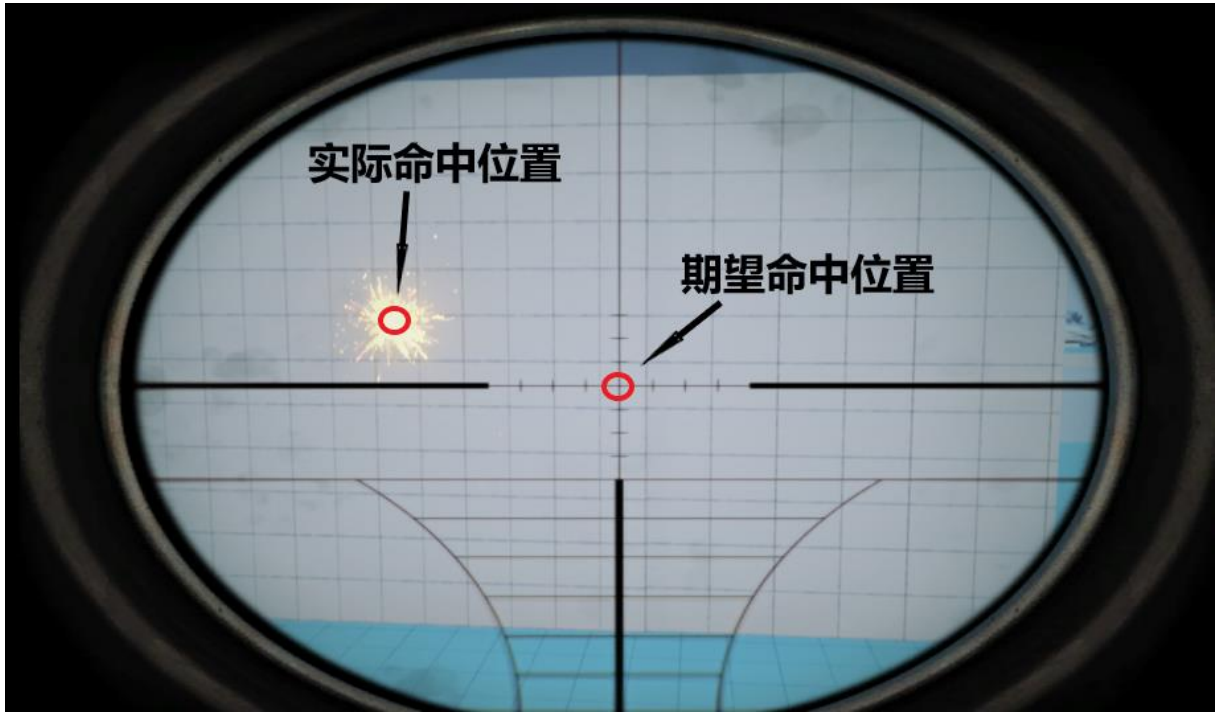
- 如果不在视野范围内, 则不行动
- 如果在视野范围内且在攻击范围内, 攻击
- 否则移动到玩家的位置





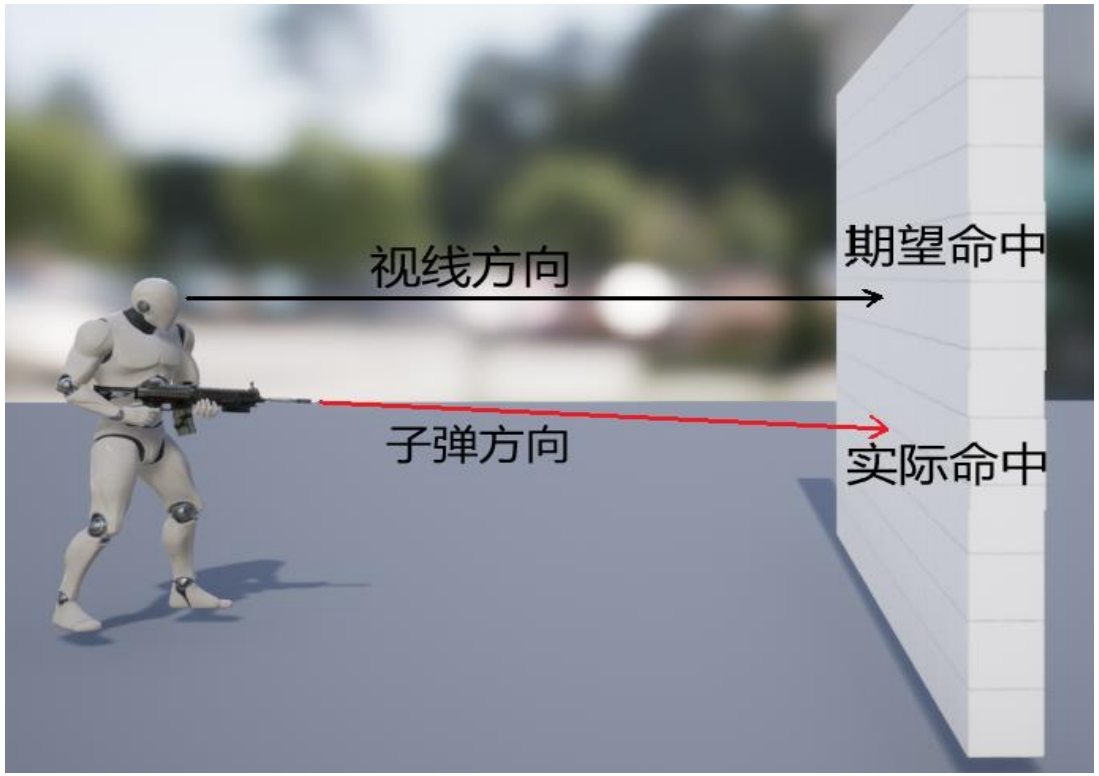
射不准问题

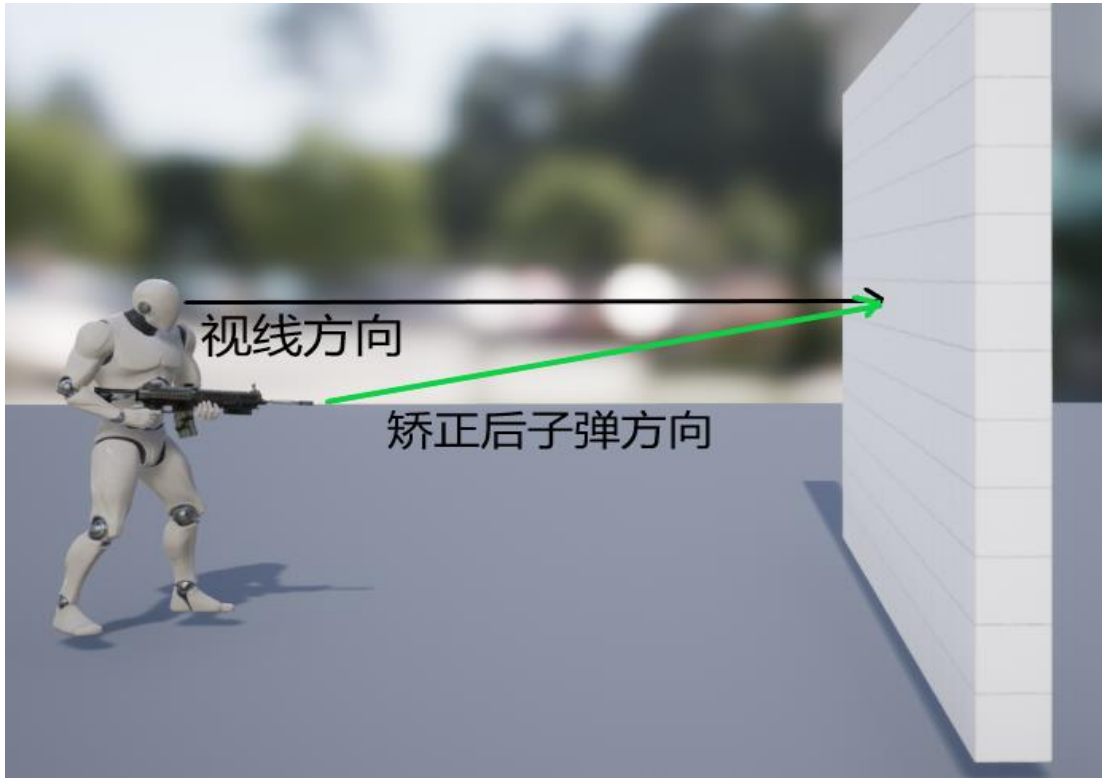
- 子弹击中的位置不在屏幕正中间
  - 玩家无法进行瞄准
- 击中位置的偏移量是变化的
  - 偏移量和场景中的物体有关



射不准问题: Plan A

- 对子弹发射方向进行矫正
  - 从玩家视线方向 (相机方向) 发出射线检测与场景求交
  - 得出的交点就是子弹应该击中的点
  - 知道起点 (枪口), 终点(交点)之后, 可以得出矫正后武器的开火方向

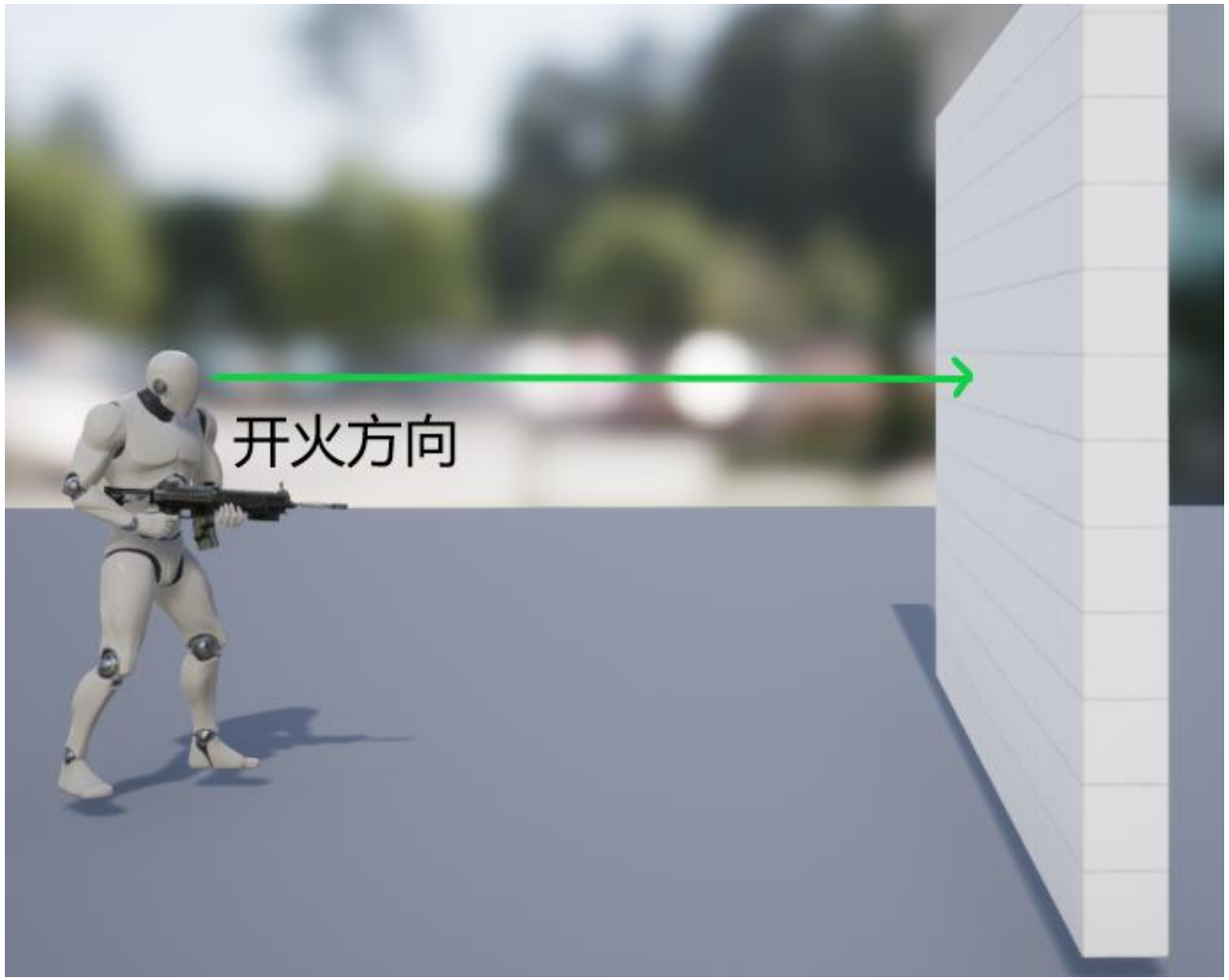




Cons: 视线与场景无交点时需要特殊处理; 某些情况下还是射不准

射不准问题: Plan B

- 用相机的位置和朝向作为开火的方向
  - 击中的位置总在屏幕中心



Pros:

- 实现原理简单

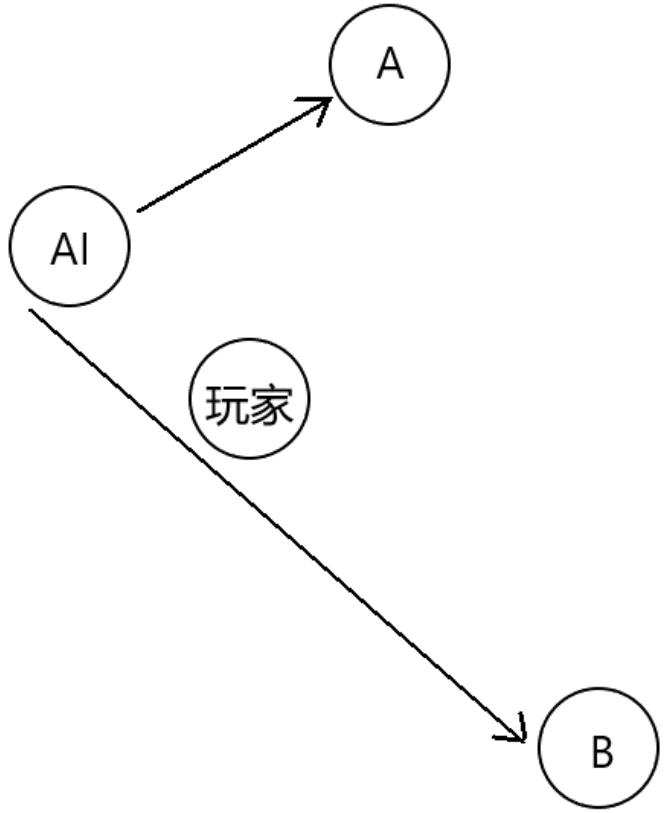
Cons:

- 只适用于射线枪械,
- 对模拟真实子弹和弹道的枪械不适用

逃离问题

- 需求: 敌人 AI 的逃离行为
  - 在敌人血量足够低之后, 主动逃离玩家

- 随机抽点法并不能满足需求
  - 随机取点, 然后走向距离玩家最远的点
  - 虽然距离上看 B 点比 A 点离玩家更远
  - 但效果上看 A 点比 B 点更好
  - 因为 AI 走到 B 的过程中, 会先靠近玩家
  - 仅从距离上无法分辨这种情况



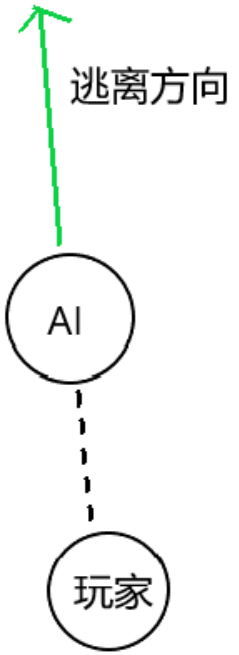
逃离问题: Plan A

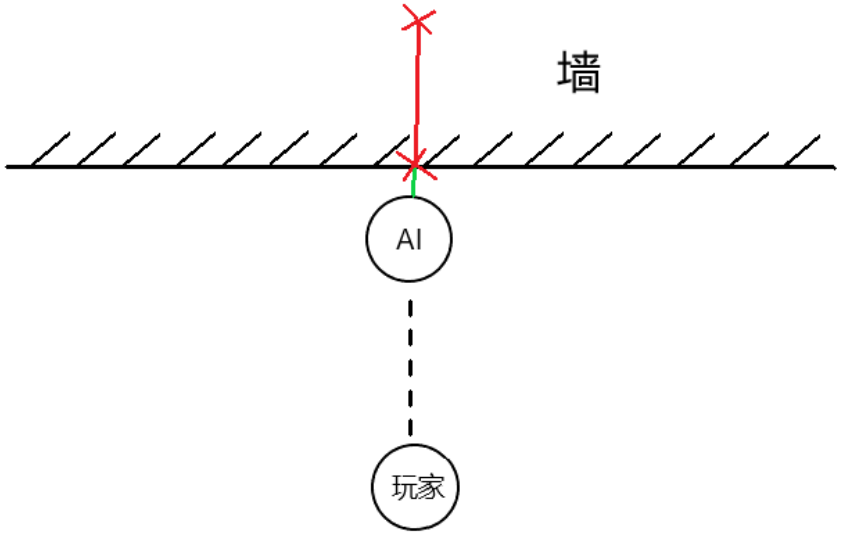
- 往相反方向逃跑

Pros: 不会出现靠近玩家的问题

Cons:

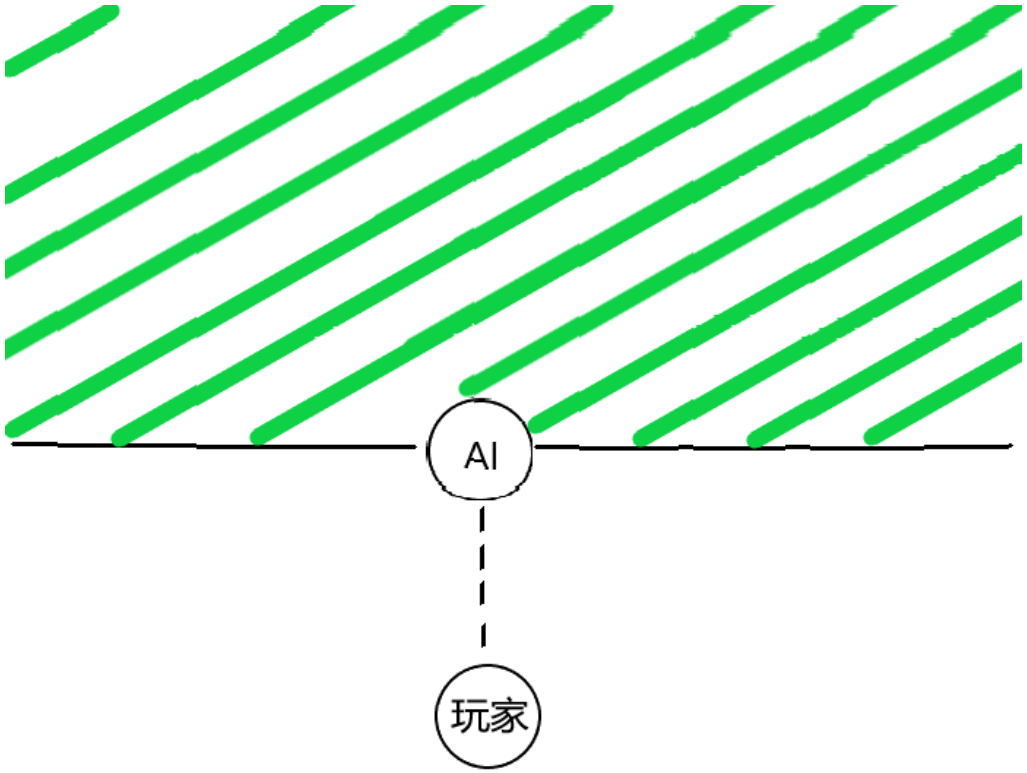
- 某些情况下无法移动
- 移动的方向过于单调





逃离问题: Plan B

- 一定区域内随机采样 + 可达性检测



Pros: 不会有靠近玩家的行为, 基本不会有无法移动的情况

Cons: 需要一些额外的开销

可以做得更好的地方

- 游戏内容扩展
  - 玩家增加特殊技能
- 项目代码组织
  - C++, 蓝图的使用
- 性能优化方面
  - 减少 Tick 的使用