# AI状态机

AIState：  
        Wandering, 巡逻  
        Collecting, 拾取道具  
                Attacking, 攻击  
              Fleeing, 逃跑  
                Diverting, 绕路  
               Retrieving, 收回回旋镖  
                FlickingSwitch, 试图攻击机关开关  
              Idle, 休闲  
             Dead 死亡

状态机初始化：

AI初始化。

各个目标初始设置为null：

this.threatTarget = null;  
     this.victimTarget = null;  
     this.desireTarget = null;  
     this.targetObject = null;

创建寻路路径，设定AI难度：

this.CreatePath();  
     this.SetDifficulty();

初始化AI相关的各个参数：

ambientInterestMod：环境兴趣值

greedyMod：对应期待值

nervousMod：对应威胁值

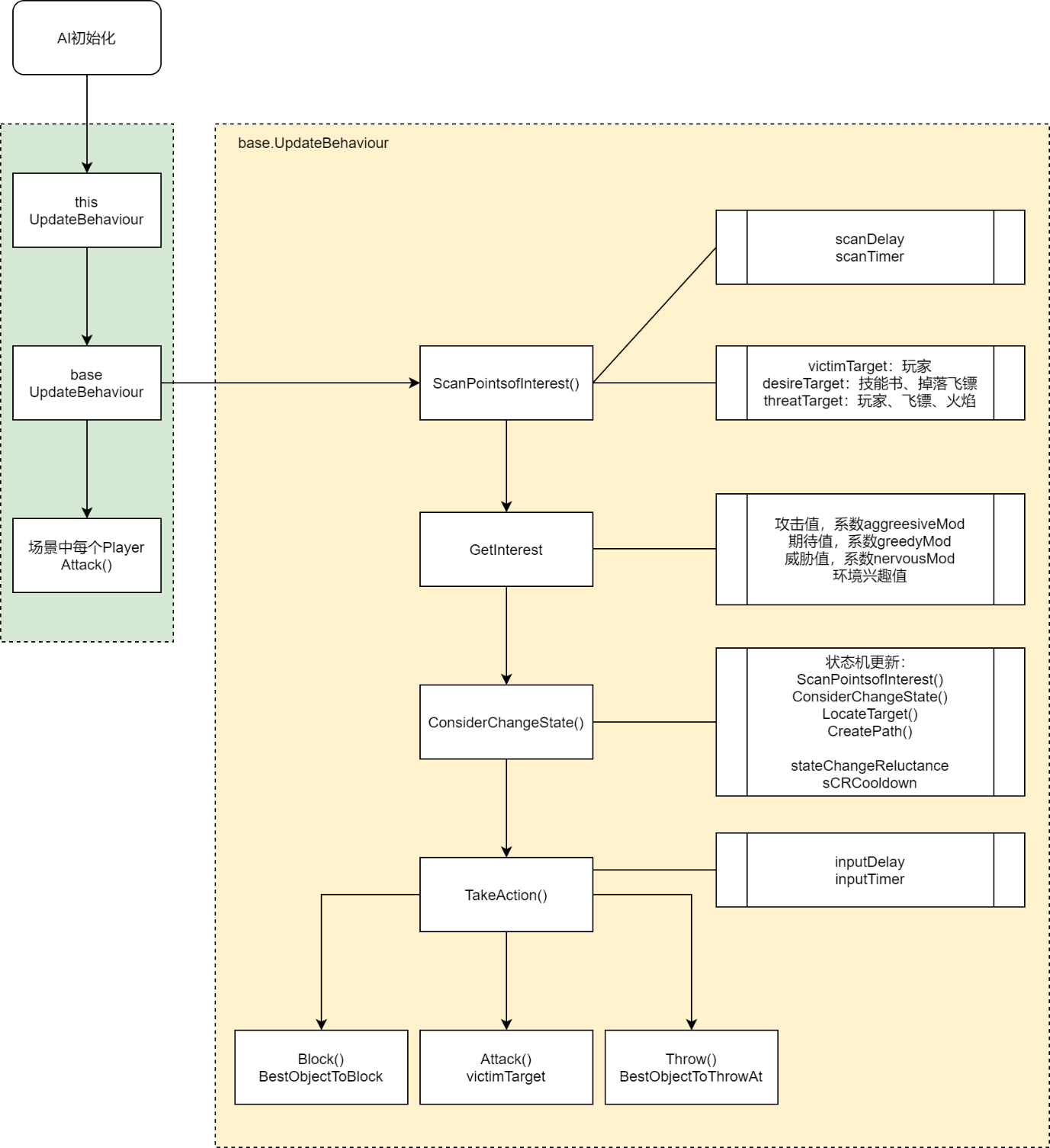
aggressionMod：对应攻击值

状态机初始化为Wandering状态。

unsafeDashAttempts = 0（如果这个值大于7，就执行FindSafePlaceToDash()）

# 决策

## 决策流程



## 扫描兴趣点ScanPointsOfInterest()

this.bestDesireSoFar = 0f;  
     this.bestVictimSoFar = 0f;  
     this.bestThreatSoFar = 0f;

对于场景中的每个玩家，如果处于存活状态并且是关系敌对：

则进行攻击目标扫描

且进行威胁目标扫描

对于场景中每个激活状态的回旋镖，

如果处于掉落状态：

则进行期望目标扫描。

如果处于有伤害状态或多回旋镖状态：

则进行威胁目标扫描

对于场景中每个技能道具，

进行期待目标扫描

## 三类目标扫描（期待目标、威胁目标、攻击目标）

期待目标扫描：

用GetInterestOfDesire方法获得interestLevel。

如果interestLevel大于当前最大期待值（bestDesireSoFar），则把扫描目标置为期待目标。

当前最大期待值 = interestLevel。

this.totalDesireLevel += this.interestLevel;

威胁目标扫描和攻击目标扫描类似

## 三类数值计算（期待值、威胁值、攻击值）

* **期待值：**

基础系数 = Min(1, Max(1.25 – 期待目标和角色距离 /场景大小， 0))

对于技能道具：

如果是Debuff类（目前只有操作反向），则期待值 = 0

其他技能道具，期待值 = 基础系数 \* 1.5 \* greedyMod

**以下都针对回旋镖**

如果期待目标是金色回旋镖：

期待值 = 100

如果期待目标是有爆炸技能的回旋镖 或 正在燃烧的回旋镖 或 处于危险位置没有拥有者的回旋镖：

期待值 = 0

对场景内每个回旋镖进行检测，如果玩家拥有回旋镖数量 + 场景中飞行的回旋镖数量大于1：

期待值 = 0

如果期待目标是回旋镖且回旋镖的拥有者是该角色：

则增益系数 = 4

期待值 = 基础系数 \* 增益系数 \* greedyMod

如果回旋镖没有拥有者 或 游戏模式允许抢夺回旋镖，并且玩家处于解除武装状态：

则增益系数 = 4

期待值 = 基础系数 \* 增益系数 \* greedyMod

如果回旋镖没有拥有者 或 游戏模式允许抢夺回旋镖，并且期待目标不在警戒线范围内：

则增益系数 = 0.5

期待值 = 基础系数 \* 增益系数 \* greedyMod

如果期待目标x坐标大于40 或 y坐标大于30：

期待值 = 0

如果回旋镖没有拥有者 或 游戏模式允许抢夺回旋镖，并且拥有金色回旋镖（持有或扔出）：

则增益系数 = 0.5

期待值 = 基础系数 \* 增益系数 \* greedyMod

对于回旋镖没有拥有者 或 游戏模式允许抢夺回旋镖的其他情况：

增益系数 = 3

期待值 = 基础系数 \* 增益系数 \* greedyMod

如果回旋镖有拥有者且拥有者死亡：

期待值 = 0

如果回旋镖有拥有者且回旋镖与角色是敌对状态：

期待值 = 0

* **威胁值：**

**以下是威胁目标为玩家的情况**

如果威胁目标处于解除武装状态或被冰冻状态：

系数1 = 0

如果角色处于解除武装状态：

系数1 = 5.25

如果角色拥有回旋镖数量 = 0：

系数1 = 2

如果威胁目标拥有的回旋镖数量 小于 角色的回旋镖数量：

系数1 = 0.8

如果威胁目标拥有的回旋镖数量 等于 角色的回旋镖数量：

系数1 = 1

如果威胁目标处于着火状态并且角色不处于着火状态：

系数2 = 5

其余情况：

系数2 = 1

系数3 = 0.45 – 0.05 \* cos(威胁目标和角色正前方夹角)

如果角色能够看见威胁目标：

系数4 = 1

如果角色不能看见威胁目标：

系数4 = 0.5

如果威胁目标在角色的直线视线内：

系数5 = 1

如果威胁目标不在角色的直线视线内：

系数5 = 0.6

如果威胁目标正在瞄准 并且 cos(威胁目标正前方和角色夹角) > 0.25：

系数6 = 2

否则：

系数6 = 1

系数7 = Max(1 – 威胁目标和角色距离 /场景大小 \* 2, 0)

威胁值 = 系数1 \* 系数2 \* 系数3 \* 系数4 \* 系数5 \* 系数6 \* 系数7 \* nervousMod

**以下是威胁目标为回旋镖(Explosive)的情况**

如果威胁目标在角色的直线视线内并且威胁目标与角色距离 < 17：

威胁值 = 6 \* nervousMod

其余情况：

威胁值 = 1 \* nervousMod

**以下是威胁目标为火焰的情况**

系数8 = Max(1 – 威胁目标和角色距离 /场景大小 \* 2, 0)

如果系数8 < 0.5：

威胁值 = 0

其余情况：

威胁值 = 系数8 \* (0.5 – 0.5 \* cos(威胁目标和角色正前方夹角)) \* nervousMod

* **攻击值：**

如果角色处于解除武装状态：

攻击值 = 0

对于金色回旋镖模式，如果攻击目标拥有金色回旋镖：

攻击值 = 10

对于金色回旋镖模式，如果角色拥有金色回旋镖并且胜利条件为击杀：

系数1 = 2

对于金色回旋镖模式，如果角色不拥有金色回旋镖：

系数1 = 0.5

对于其他游戏模式：

系数1 = 1

如果攻击目标处于解除武装状态：

系数2 = 1.2

如果攻击目标拥有的回旋镖数量小于或等于角色拥有的回旋镖数量：

系数2 = 1

其余情况：

系数2 = 0.8

系数3 = Min(cos(威胁目标和角色正前方夹角) \* 0.2 + 1, 1)

如果攻击目标在角色的直线视线内：

系数4 = 1.1

如果攻击目标不在角色的直线视线内：

系数4 = 1

如果能够看见攻击目标：

系数5 = 1

如果不能看见攻击目标：

系数5 = 0.25

系数6 = Min(Max(1.5 – 攻击目标与角色距离/场景大小, 0), 2)

如果攻击目标被冰冻：

系数7 = 1.3

如果攻击目标不被冰冻：

系数7 = 1

如果攻击目标是领先玩家：

系数8 = 2.5 - Min(剩余胜利分数 \* 0.2, 1.25)

如果攻击目标不是领先玩家：

系数8 = 1

攻击值 = 系数1 \* 系数2 \* 系数3 \* 系数4 \* 系数5 \* 系数6 \* 系数7 \* 系数8 \* aggressionMod

* **各个值初始化**

调整兴趣值相关的各个参数：

ambientInterestMod：环境兴趣值

greedyMod：对应期待值

nervousMod：对应威胁值

aggressionMod：对应攻击值

针对不同难度，设定不同的ambientInterestMod：

简单：random(1, 2)

中等：random(0.9, 1.5)

困难：random(0.8, 1.2)

变态难度：random(0.6, 1)

初始aggressionMod = random(0.6, 1)

如果难度大于中等难度：aggressionMod 增加0.3。

如果胜利条件时击杀并且角色拥有金色回旋镖：

aggressionMod = aggressionMod \* random(1, 2)

如果胜利条件是存活并且角色拥有金色回旋镖：

nervousMod = random(1, 2)

其余情况：

nervousMod = random(0.8, 1.2)

如果是简单难度：

nervousMod 增加0.3。

如果玩家拥有控制反向技能：

nervousMod = nervousMod \* 1.5

ambientInterestMod = ambientInterestMod \* 5

如果玩家处于Retrieving状态：

nervousMod = nervousMod \* 1.25

greedyMod初始值 = random(0.8, 1.2)

## 状态机切换

教学阶段，有一个特殊处理，会调用FlickingSwitch状态。

最开始的一段时间（gettingBearingsTimer > 0.1f），处于Wandering状态。

this.gettingBearingsTimer = this.gettingBearingsDuration \* Random.Range(0.75f, 1f);

gettingBearingsDuration在设定AI难度的时候初始化，与AI难度相关。

进入Collecting状态的条件集合（各条件之间为且关系）：

* 当前最佳期待值（bestDesireSoFar） > 当前最佳攻击值（bestVictimSoFar）
* 当前最佳期待值（bestDesireSoFar） > 当前最佳威胁值（bestThreatSoFar）
* 当前最佳期待值（bestDesireSoFar） > 环境兴趣值（ambientInterestLevel）
* 当前最佳期待值（bestDesireSoFar） > 状态改变阻力值（stateChangeReluctance）
* 剩余敌人数量 > 1
* 期待目标（desireTarget）不为空
* 期待目标（desireTarget）和角色的距离大于下个路点的距离（nextWaypointDistance）

以上条件满足后，把当前目标置为期待目标（desireTarget），并且角色 进入Collecting状态。

进入Retrieving状态的条件集合（各条件之间为且关系）：

* 所有Collecting状态的条件都需要满足
* 上一个状态（fsm.LastState）不是Retrieving
* 角色不拥有爆炸回旋镖技能
* 角色在安全区内
* 角色手中没有回旋镖
* 角色扔出去的回旋镖数量 > 0
* 角色当前目标为回旋镖
* 角色当前目标在角色的直线视线内（IsInLineOfSight）
* 以下两个条件满足一个 ：
  + 目标和角色之间的距离 > 角色跳跃距离的2倍 且 以下两个条件满足一个：
    - 简单难度，随机10%概率，否则随机33%概率
    - 目标不在安全区内
  + 目标所在位置跳跃不安全
* 当前最佳威胁值 < 当前最佳期待值 \* 0.5

进入Attacking状态的条件集合（各条件之间为且关系）：

* 当前最佳攻击值（bestVictimSoFar）> 当前最佳期待值（bestDesireSoFar）
* 当前最佳攻击值（bestVictimSoFar）> 当前最佳威胁值（bestThreatSoFar）
* 当前最佳攻击值（bestVictimSoFar）> 环境兴趣值（ambientInterestLevel）
* 当前最佳攻击值（bestVictimSoFar）> 状态改变阻力值（stateChangeReluctance）

如果满足上述条件集合，则把当前目标置为攻击目标（victimTarget），并且进入Attacking状态。

如果攻击目标处于草丛隐藏状态，则先执行靠近靠近草丛行为，再进入Attacking状态。

进入Fleeing状态的条件集合（各条件之间为且关系）：

* 当前最佳威胁值（bestThreatSoFar）> 当前最佳期待值（bestDesireSoFar）
* 当前最佳威胁值（bestThreatSoFar）> 当前最佳攻击值（bestVictimSoFar）
* 当前最佳威胁值（bestThreatSoFar）> 环境兴趣值（ambientInterestLevel）
* 当前最佳威胁值（bestThreatSoFar）> 状态改变阻力值（stateChangeReluctance）

如果满足上述条件集合，并且角色当前状态不处于Fleeing状态，或者随机33%的概率成功，则把当前目标置为威胁目标（threatTarget），并且进入Fleeing状态。

以上情况都不满足时，如果环境兴趣值（ambientInterestLevel）> 状态改变阻力值（stateChangeReluctance），则角色进入Wandering状态。

## 攻击最佳目标判定

最佳攻击目标

敌人，机关开关。

判断条件：距离是否在攻击范围内、是否在视线内。

最佳投掷目标

敌人。

判断条件：是否在视线内、是否在投掷攻击范围内、投掷后目标位置预判（是否在投掷直线内）。

最佳阻挡目标

可以阻挡敌人和阻挡回旋镖。

根据玩家和回旋镖与自己的距离来判定。

## 采取行动TakeAction()

根据扫描到的可用目标点及对应值，进行状态转换，进入攻击状态，执行追击、进程攻击、投掷的攻击行为。

# 行为树

## 条件枚举

是否拥有目标（包括之前提及的三个目标）

是否在近程攻击范围内

是否在投掷攻击范围内

是否在视线内

跳跃是否安全

目标是否可以寻路到达

飞镖是否在手中

## 行为枚举

近程攻击

投掷飞镖

跳跃

追击：

攻击开关

收回飞镖：走到可以收回的地点，长按投掷键收回飞镖。

拾取技能道具

阻挡：

巡逻：

逃跑：朝目标的反方向移动。

绕路

## 行为优先级

攻击机关开关

阻挡目标

近程攻击

投掷回旋镖

# 难度参数

## 反应时间

在每次采取行动之前，会随机取一次反应时间，随机范围和AI难度有关系。

## 按钮输入延迟

AI点击按钮的延迟时间。

## 投掷和攻击概率

不同难度AI在进入攻击和投掷状态前，会进行一次概率判定。

## Block()行为概率

执行Block行为前，会按难度进行概率判定。

## 出生时间长度

gettingBearingsDuration

在出生时间内会有特殊逻辑，难度越大，这个时间越小。

## 执行TakeAction()的概率

按难度有执行概率的判断，如果是简单难度，有25%的概率不执行任何操作。

## 处于冰冻状态下，执行攻击操作的概率

在冰冻状态下，执行攻击操作可以解除冰冻。

难度大于简单难度，则有66%概率执行攻击操作。

## 判定最佳投掷目标的角度范围

难度越大，角度判定越精确。

## 投掷后目标位置预测系数

根据目标的速度预测目标在投掷后的位置偏移量。

偏移量乘以这个系数，简单=0，中等=0.47，困难=0.9。

## 投掷时转向精准度

简单：66%概率往投掷目标方向（-30,30）内随机旋转一个角度，33%概率不旋转。

中等：往投掷目标方向与自己方向夹角旋转 一半角度。

困难：直接转向投掷目标。

## 收回飞镖概率

对不同难度的AI，会有不同的进入收回飞镖的概率。

## 不同的环境兴趣值

简单：random(1, 2)

中等：random(0.9, 1.5)

困难：random(0.8, 1.2)

变态难度：random(0.6, 1)

## 不同的攻击值系数（aggressionMod）

如果难度大于中等难度：aggressionMod 增加0.3。

## 不同的威胁值系数（nervousMod）

如果是简单难度：nervousMod 增加0.3。

## 不同的扫描兴趣点延迟（scanDelay）

困难难度的机器人x3

## 进入收回回旋镖状态的条件判断

简单难度进入收回回旋镖状态概率更低

## 能否发现伪装状态下的敌人

## 能否发现草丛中的敌人