游戏中智能体行为的选择与学习

知识的表述和游戏智能体目标的确定

在详细叙述每一种算法类型之前，需要概述两个主要概念：第一个是算法的表示；第二个则是它的效用。一方面，任何人工智能算法都要在某种成都上存储和维护与需要处理的特定任务有所关联的知识。另一方面，大多数人工智能算法都在寻求更好的只是表示。这种寻求的过程由某种形式的效用函数所驱动。我们需要注意的是，效用在那些仅仅依赖于静态只是表示的方法中是无法使用的，例如有限状态机或行为树。

一、表示

合适地表示只是一般人工智能所面对的一个关键挑战，它是由人类大脑存储以及取回已经获得的有关世界的只是的能力所启发的。推动设计为 AI 使用的表示的关键问题包括人们如何表达知识而 AI 如何模仿这种能力？知识的本质是什么？一种表示方式可以有多通用？然而，上述问题的普遍回答在当前还是十分粗略的。

在应对关于知识及其表示的开放性的一般问题上，AI 已经确定了多种非常明确的方法来存储与取回已经创作过、获得过或是学习过的信息。对于一个任务或一个问题相关的知识 的表示可以被视为被调查任务的计算映射。在这个基础上，表示通过一种机器能够处理的形式来存储任务相关的知识，例如某种数据结构。

让任意形式的 AI 知识需求能够以可计算的方式得到表示的实现方法是多种多样的。表示的类型包括文法（如文法演变）、图（如有限

状态机与概率模型）、树（如决策树、行为树以及遗传算法）、连接机制（如人工神经网络）、进化（如遗传算法和进化策略）以及表格（如时序差分学习）。在本书的剩余部分看到，我们会看到所有上述的表示类型在游戏中有着不同的用处，并且可以与各种游戏 AI 任务进行结合。

对于在某种特定任务上进行尝试的任何 AI 算法而言，有一点是毫无疑问的：所选择的表示方法对算法的性能有重大的影响。不幸的是，为了某个任务而被选中的表示方法也要遵循“没有免费午餐”原理，这说明，对所有需要处理的任务来说，并不会存在某种理想的单一表示类型。然而，就一种一般的指导方案来说，被选中的表示方法应当尽可能简单。简单性通常可以在计算量与算法性能之间达成一个微妙的平衡，因为无论是过于详细或者是过于简化都将会影响到算法的表现。除此之外，考虑到当前任务的复杂度，被选中的表示也应当尽可能小。表示方法的简单性或大小都是不容易抉择的。良好的表示方法来源于足够的对于 AI 正在试图解决的问题的复杂度及其定性特征的实践智慧与经验知识。

二、效用

博弈论（和一般经济学）中的效用是指在博弈过程中对理性选择的一种衡量。一般来说可以被视为一个能够协助搜索算法选择路径的函数。出于这个目的，效用函数需要对搜索空间的各个方面进行采样，并且收集搜索空间中各个区域的“优秀程度”的信息。在某种意义上，一个效用函数就是我们正在尝试寻咋哄的解的某种近似值。换句话

说，它也就是一种对于我们已经搜索到的表示在优秀程度上的衡量。

与效用类似的概念包括了计算机科学与 AI 中使用的启发式，在精确算法太慢而且难以负担的情况下，它可以作为一种更快地、近似地解决问题的方式，特别是与树搜索相结合的相结合的情况下。适应度的概念也在类似的情况下被用作一种能够衡量解决方案优秀程度的效用函数，其主要被用于进化计算领域中。而在数学优化上，目标函数、损失函数、代价函数、误差函数都是需要被最小化的效用函数（不过在作为目标函数的时候需要被最大化）。特别需要说的是，在监督学习中，误差函数代表了某种方法在将样本映射到（期望的）目标输出这个过程上的优秀成都。在强化学习以及马尔可夫决策过程中，效用也被称为奖赏，是智能试图通过学习在特定状态下选择正确动作而最大化函数。最后，在无监督学习领域中，效用通常由自身所提供，并且存在于表示之中，例如竞争学习或自组织中。

与选择适当的表示方法类似，对效用函数的选择也遵循“没有免费的午餐”原理。效用函数通常都难以设计，并且在某些时候这个设计任务基本无从下手。设计的简单性和完整性同样有效。效用函数的质量极大地取决于从被研究领域当中所获得到的全面的实证研究与实践经历。

三、学习=最大化效用（表示）

效用函数是搜索的驱动，而在本质上也是学习的驱动。在这个基础上来说，效用函数可以说是每一类机器学习算法的训练信号，因为它对我们已有的表示提供了一种在优秀程度上的衡量。因此，对于如

何进一步地提高现有表示的优秀程度，它隐含地提供了指示。而不需要学习的系统也并不需要效用。在监督学习当中，效用是从数据中采样而来的——例如某种优秀的输入-输出模式。在强化学习与进化计算中，训练信号是由环境所提供的——也就是在表现优秀时得到的奖赏与在表现糟糕时得到的惩罚。最后，在监督学习中，训练信号来源于表示的内部结构。

特定行为的编辑

有限状态机、行为树与基于效用的 AI 都属于特定行为编辑方法，在传统上主导了对游戏中的非玩家角色的控制。它们的地位显而易见，因为当前的游戏开发角度中，游戏 AI 这个术语依然是这些方法的代名词。

一、有限状态机

有限状态机（FSM）——以及有限状态机的变体，例如分层有限状态机——直到 20 世纪 90 年代中期，都是对游戏中的非玩家角色的控制与决策过程的主导游戏 AI 方法。

有限状态机属于专家知识系统领域，可以用图来表示。一个有限状态机对应的图是一组相互关联的对象、符号、事件、动作或现象属性的集合的抽象表示，需要进行专门的设计（表示）。特别的是，这个图包含了一些潜入了数学抽象的节点（状态），还有一些代表了节点见的条件关系的边（转换）。有限状态机在单个时间内只能处于一个状态；而在对应的转换条件满足的情况下，当前状态可以转变为另

一种状态。简而言之，一个有限状态机是由三个主要组成部分所定义的：

- 一定数量的状态，其存储了关于某种任务的信息——例如，你当前正处于探索状态。
- 一定数量的在状态之间的转换，其指出了某种状态的改变，并通过一个需要得到满足的条件来描述——例如，你听到了一声强项，转换为警戒状态。
- 一组需要在每种状态中遵循的动作——例如，当处于探索状态时，随机地移动并且寻找敌人。

有限状态机的设计、实现、可视化和调试十分简单。此外，在与游戏共同存在的多年时间中，它们也证明了自身在表现上的优秀。然而，在大规模的游戏环境中，它们的设计会变得极端复杂，因此，在游戏ＡＩ中的某些任务上存在计算性上的限制。另一个有限状态机的关键限制（也是对所有特定编辑方法来说）是它们缺乏灵活性与动态性（除非得到了特殊的设计）。而在有限状态机的设计、测试和调试完成后，适应性和演化性的发展空间有限。结果就是，有限状态机最终在游戏中产生了非常容易预测的行为。我们可以通过转换表示为某种模糊规则或概率来在一定程度上客服这个缺点。

1.1. 一个用于《吃豆人》的有限状态机

一个假设的并简化过的吃豆人有限状态机控制器如图所示。在这个例子中，我们的有限状态机有三个状态（寻找豆子、追逐鬼魂以及逃避鬼魂）与四个转换（鬼魂闪烁、没有看见鬼魂、鬼魂出现在视野

中，以及吃下能量药丸）。在处于寻找豆子状态中时，吃豆人会随机地移动，直到探测到一个豆子，然后采用某种寻路算法来尽可能吃掉尽可能多的豆子。如果吃到了能量药丸，那么吃豆人就会转变为追逐鬼魂状态，此时它可以使用任何的树搜索来逃避鬼魂，使得在一定距离看不见任何鬼魂；而当达到这一情况后，吃豆人又会转回寻找豆子状态。

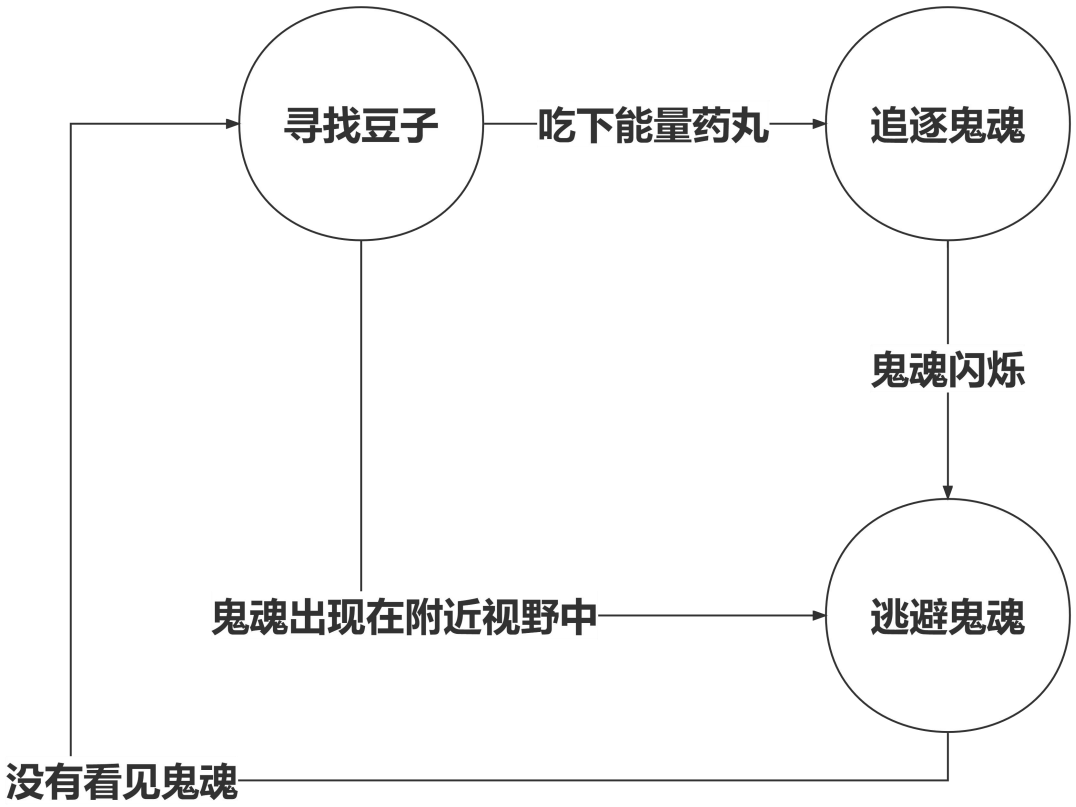

图 1：一种用于控制吃豆人的有限状态机

1.2. 一个用于《我的世界》的有限状态机

《我的世界》（Minecraft）是一款沙盒类电子游戏，开创者为马库斯·阿列克谢·泊松（Notch）。游戏由 Mojang Studios 维护，隶属于微软 Xbox 游戏工作室。中国版现由网易游戏代理，于 2017 年 8 月 8 日在中国大陆运营。

图 2 ：我的世界

自开创伊始到延斯·伯根斯坦加入并负责开发之前，我的世界几乎全部的开发工作由 Notch 完成。游戏音乐由丹尼尔·罗森菲尔德（C418）和莉娜·雷恩（Lena Raine）创作；克里斯托弗·泽特斯特兰绘制了游戏中的画。游戏最初于 2009 年 5 月 17 日作为 Classic 版本发布，并于 2011 年 11 月 18 日发布 Java 正式版。我的世界的游戏平台囊括桌面设备、移动设备和游戏主机。

该游戏以玩家在三维空间中自由地创造和破坏不同种类的方块为主题。玩家在游戏中可以在单人或多人模式中通过摧毁或创造精妙绝伦的建筑物和艺术，或者收集物品探索地图以完成游戏的成就（进度）。玩家也可以尝试在创造模式下(打开作弊)红石电路和指令等玩法。

村民（Villager）是 Minecraft 中的一种出生并生活在村庄周围的被动型智能 NPC，他们出生在村庄附近或是对应各自职业的建筑物内。

图 3：我的世界中的村民

村民会依自己的职业进行工作，能够繁殖和与玩家互动。它们的衣服因其职业和村庄所在生物群系而异。玩家可以以绿宝石作为货币和村民交易。

村民在白天会离开它们的家，并开始在它们所在的村庄闲逛。通常，它们会在白天在村内进行漫无目的地闲逛。它们会到户外或者户内活动，还会不时地喃喃自语。偶尔，2 个村民可能会相遇，并互相看着对方，以独特的语言进行交谈，这就是村民的社交行为，一个村民会盯另一个村民 4-5 秒。只要玩家靠得够近，它们就会一直注视玩家，除非村民试图收割农作物、工作、躲避僵尸与灾厄村民，或是在夜间回到屋子里。

村民不会一直停留在玩家面前，虽然它们走开的时候也会盯着玩

家看。如果村民受到玩家攻击，它们会逃跑。

即使村庄足够大，村民也往往不会远离它们的床，除非它们的工作站点或最近的会合点（钟）离它们的床很远。村民像其他生物一样会寻找躲避障碍的路径，避开伤害性的方块与悬崖。然而，在拥挤的情况下，村民可能会被其他村民推下悬崖或碰到伤害性的方块。

在晚上或下雨期间，村民会躲在屋内躲避，并关上屋内的门。它们会试图睡觉，如果它们无法睡觉，就会待在屋内直至早晨。到早上时，村民会恢复正常的行为。村民可以在下界和末路之地睡觉，床不会因此爆炸。

村民会尝试与僵尸、卫道士、掠夺者、劫掠兽和恼鬼保持至少 8 格的距离，而唤魔者和幻术师则是 12 格。

如果村民发现它走到村庄边界以外，或村民检测到 32 个方块范围内有村庄，它会快速回到村庄界内。若村民在离开村庄 32 格外，它会在 6 秒后忘记该村庄。无论村民是否在村庄中，它们都不会被清除。

村民可以开关任何的木制门并寻找门后使他们感兴趣的路径或方块。然而，它们不能打开活板门、栅栏门、铁门。村民可以攀爬梯子，但不会将梯子认作路径，并且不会刻意地使用梯子。村民攀爬梯子的行为似乎是一种被其他生物尝试推入梯子附着的方块内的意外后果（常常是因为其他村民）。不幸的是，这个行为将会让它们滞留在二楼或是一些结构的屋顶上，因为他们缺乏从梯子下降的必要 AI。[需要验证]对于这种情况，一个简单的解决方法是玩家可以手动将村民推下梯子，并在村民到达地板之后在梯子顶端安装一个活板门。一个

阻止村民攀爬梯子的方法是破坏与地板相邻的那一个梯子，若如此做则需要玩家跳上梯子来攀爬。

- 首选路径（此特性为基岩版独有）

村民会优先步行在通往目的地的下列方块成本花费较低的方块上，并且它们尽量会避免跳跃。

- 工作站点方块

成年村民（傻子除外）会检测球面半径 48 个方块内的工作站点方块来寻找工作。失业的村民可以通过认领它找到的第一个无人认领的工作站点方块来获得职业和工作。无论这个工作站点方块是否为人为放置、可到达性甚至可见性，都可以被检测到，只要它在范围内并且未被认领。当这个工作站点被认领时，其所有者会发出绿色粒子。除非所有者主动放弃，否则其他村民不可以认领。

当村民准备认领空闲的工作站点时，其他村民均无法认领该工作站点。若此时该村民由于受到阻挡等原因无法将其成功认领，其他村民也将一直不能认领该工作站点。

当村民距离自己的工作站点过远时，若附近有未被认领的工作站点，则它也会改变职业。

如果一个工作站点方块被破坏或毁坏，其所有者（如果有）会发出愤怒粒子[仅基岩版]并变成失业者，但如果已与其交易，则会保留其职业。如果在检测范围内找不到其他工作站点方块，该村民就会失业。村民传送到其他维度后会失去工作站点。已经有职业但没有工作站点的村民会试图寻找一份新工作：

没有交易过的村民可以认领任何工作站点方块，改变其职业并获得新工作。

已经交易过的村民只能认领与其职业相对应的工作站点方块。

在 Java 版中，村民只能在非睡眠状态下改变或失去职业。

在 Java 版中，工作站点方块的高度必须和村民脚部高度相同或在其下方一格内才能被认领，换言之悬空或是村民脚部无法碰触到的工作站点方块无法被认领。此外村民和工作站点方块的寻路路径上若存在村民无法行走的方块（如岩浆块），可能会导致村民无法认领该工作站点方块。

- 交谈

村庄会以言论（gossip）的形式储存对玩家的特定记忆。村民会在交谈时接收各种各样的言论，随后将其传播给其他村民。每条言论都有五种类型，并存储一个目标（target）和值（value）。言论的类型一共有 5 种： major_negative、 major_positive、 minor_negative、 minor_positive 和 trading。其中 major_positive 不会被村民传播。言论的目标就是使得该言论产生的玩家。如果某村民将要产生或接收一条新的言论，但其已经有了一条类型和目标都与之相同的言论，则已有的这条言论的强度就会提高，而那条新的言论并不会被产生或接收。一种言论的强度越高，村民就越有可能将其传播出去。

当村民被玩家攻击时，会产生 minor_negative 言论；当玩家杀死村民时，附近的村民就会立即收到 major_negative 言论；如果玩家治疗好一个僵尸村民，被治疗的村民会产生 minor_positive 和

major_positive 言论；如果玩家和村民交易，村民就会产生 trading 言论。负面言论会使村民的交易加价，正面言论会使交易打折。总体的信誉也会影响铁傀儡是否攻击玩家。如果 major_negative 的强度达到 20 或以上，则该村庄的铁傀儡会与玩家敌对。

村民受言论影响后的交易价格的具体计算公式为：y = x - floor((5a + b + c - d - 5e) × p)，其中 y 是最终价格，x 是基准价格，a、b、c、d、e 分别对应 major_negative 、 major_positive 、 minor_negative 、 minor_positive 和 trading 的强度，而 p 对应 PriceMultiplier 的值。

言论被传播并被其他村民接收时，其强度会有所降低，降幅因言论的类型而异：trading 会降低 20，minor_negative 和 minor_positive 会降低 5，major_negative 会降低 10。每过 20 分钟，不同类型的言论也都会以不同的降幅减弱一次：trading 会降低 2，minor_negative 会降低 20，major_negative 会降低 10，minor_positive 会降低 1，而 major_positive 不会随时间自然减弱。村民关于单个玩家自然产生的 major_positive 言论的强度上限为 100。

- 拾取物品

村民拥有 8 个隐藏物品格，村民生成时，其物品格是空的。村民不会专门去找物品，但是它们会去收集在它们范围内面包、胡萝卜、马铃薯、小麦、小麦种子、甜菜根和甜菜种子，即使他们一开始在睡觉（此时他们会起床捡东西，然后接着睡觉）。农民村民也会捡起骨粉。村民仅可以捡起这些物品，尽管玩家可以通过/item[仅 Java 版]或/replaceitem[仅基岩版]强制将其他物品放入村民的物品栏。如果玩

家和村民同时在一个物品的拾取范围内，玩家总是会先捡起物品。

村民死亡后也不会掉落它们捡到的东西。

如果村民变成僵尸村民，则它们的物品将一律丢失，因为僵尸村民没有物品格。

如果/gamerule mobGriefing 被设为 false 的话，那么村民将不能捡起任何物品。

- 分享食物

如果村民的 1 个物品格有比较充足的食物，即至少 6 块面包或 24 个胡萝卜、马铃薯、甜菜根，或 18 个小麦（仅农民），并且看见有村民的物品格没有足够的食品（非农民：3 块面包或 12 个胡萝卜、马铃薯或甜菜根；农民：15 块面包、60 个胡萝卜、马铃薯或甜菜根，或 45 个小麦）的话，它可能会决定把食物分享给那个村民。

如果要分享，村民会找到它第一个至少有 4 个面包，胡萝卜、马铃薯或甜菜根，或者至少 6 个小麦的物品格，然后向着目标村民的方向扔去一半数量的物品（向下舍入）。

在 Java 版中，村民会将想要分享的食物一次性地投给另一个村民；在基岩版中，村民会将食物一组一组地投出。

- 耕作

农民村民会照料村庄界限内里的农作物。距离任意村庄界限外足够远的村民也会照料它附近的农作物。

需要照料的耕地是通过寻找距离村民 X 和 Z 坐标方向最多 15 格，Y 方向最多 1 格（总体积 31×31×3）的特定方块。

如果一个农民村民的物品栏里没有足够的食物（15 块面包、60 个胡萝卜、马铃薯或甜菜根，或者 45 个小麦），并且找到已成熟的小麦、胡萝卜、马铃薯或甜菜，村民就会移动到这些作物并采集它们。

如果一个农民村民的物品栏里拥有任何种子、胡萝卜、马铃薯或甜菜种子，并且在耕地上方 1 格找到空气方块，它会移动到那片农田并种下作物。它们总会种下在物品栏中第一个可以种植的作物。

农民村民会使用并捡起骨粉，还会用种子填满堆肥桶。

如果/gamerule mobGriefing 被设为 false，那么村民将不能耕作。

农民村民不能将泥土、草方块、土径转化为耕地，也不能捡起任何锄来锄地。

如果通过命令将锄放置在村民的主手或副手上，村民仍然不会锄地。

如果一个农民村民手中有骨粉，村民会尝试用这些骨粉催熟附近的作物。

农民村民如果有额外的作物和食物，会分享给其他的村民。

- 繁殖

成年村民会根据当天的时间进行繁殖，且需要有意愿来生成幼年村民，需要床且上方至少有两个空的方块。村民繁殖不需要工作站点。在 20 分钟后，幼年村民将长大。

如果一个村民死于非生物、非玩家来源，而玩家在 16 格（球面半径）内，或者如果一个怪物杀死一个村民，那么村庄中的村民将停止繁殖大约 3 分钟。

繁殖取决于有效床的数量。如果村民有“意愿”（见下文），且人口少于有效床的数量（即在村庄范围内有无人认领的床），村民才会繁殖。所有的幼年村民最初都是没有职业的。只要在村庄范围内有无人认领的床，村民就会重新繁殖。

村庄会定期进行“人口普查”以确定该村庄的当前人口。村庄水平边界内且以村庄中心为中心的 5 格高[仅 Java 版]范围内的所有村民都在村庄人口计算范围内，以确定是否允许继续进行村民繁殖。但是，只要边界内至少有一个村民，村庄水平边界内和村庄球形边界内的任何村民都将尝试进入繁殖模式。如果两个村民彼此靠近且都进入繁殖模式，它们就会繁殖并生成一个幼年村民。

如果有足够的食物让它自己和另一个村民“有意愿”，村民可能会进入繁殖模式（村民头上会出现红色的心型粒子）。它们会根据它们的食物数量进入繁殖模式，而不是人口上限（根据床的数量），但只有有多余的床让它们繁殖出的幼年村民认领才能繁殖出幼年村民。如果到达了人口上限或所有的有效床上方都没有两格高的空气方块时（幼年村民需要足够的空间来玩蹦床），它们的繁殖就会被阻止，且它们头像会出现愤怒粒子（与红色心型粒子一起）。就像农场动物一样，当两个村民处于繁殖模式并且能够看到对方时，它们会互相靠近并对视几秒钟，接着就会在它们的旁边生成一个幼年村民。繁殖村民不会掉落经验。在 Java 版中，产下的村民的穿着在其双亲任一所属的生物群系与其出生的生物群系之间随机决定；在基岩版中，其穿着由其所在村庄所属的生物群系决定。此村民成年后，如果它靠近有一

个有效的、无人认领的工作站点方块，它就会获得与该工作站点方块对应职业。

关于意愿

村民必须有“意愿（willing）”才能繁殖。村民会在繁殖后失去意愿，而且需再次进入有意愿的状态才能进行下次繁殖。

只有村庄范围内有多余的床可供幼年村民使用时，村民才会繁殖。

村民可以通过在它们物品栏的一个槽位中加入 3 个面包，12 个胡萝卜、12 个马铃薯或 12 个甜菜根来使它们变得有意愿。任何有多余食物的村民（通常是农民）都会向其他村民扔食物，让它们捡起食物并获得足够的食物来变得有意愿。玩家还可以向村民扔面包、胡萝卜、甜菜根或马铃薯，以促进它们繁殖。村民在没有意愿的情况下会消耗所需的食物来变得有意愿。如果/gamerule mobGriefing 被设为 false，则村民将不会捡起食物。而食物不足的村民将不会产生繁殖的意愿。

- 僵尸及其变种

村民会逃离僵尸及其变种，有时它们会为此躲在房屋里。村民也会逃离僵尸猪灵[仅基岩版][失效：BE 1.18.20]，尽管它们不会攻击村民。

村民唯一的“自然”守护者便是铁傀儡，铁傀儡可以攻击附近的敌对生物。

僵尸及其大多数变种会在半径 42 个方块内寻找并攻击村民（即使看不见村民），还会试图破门。但只有一小部分僵尸才能破门，并且只有在困难难度下破门才会成功。无法破门的僵尸往往会挤在将它们

与村民隔开的门上。

僵尸及其变种会杀死村民，这有可能将其转换成僵尸村民。在简单、普通和困难难度下，转换概率分别为 0%、50%和 100%。幼年村民和被溺尸用三叉戟杀死的村民也会被感染。

- 闪电

如果闪电劈中的地方距离村民所在的地方 3-4 个方块的话，村民会变成女巫。难度为和平时，村民则会消失。

- 生成铁傀儡

村民可以生成铁傀儡。如果村民在前 20 分钟内睡过觉且在前 30 秒内观察到铁傀儡的数量不足村庄的铁傀儡上限（上限为村民数量的十分之一，向下取整），则该村民会产生生成铁傀儡的意愿。村民每过 5 秒就会观察一次周围是否存在铁傀儡，铁傀儡到该村民的各个坐标轴上的距离不超过 16 格（无论其间是否存在障碍物）时才会被观察到。村民最多每 60 秒传播一次言论，而产生意愿的村民会在成功传播言论的同时尝试生成一次铁傀儡。若该村民周围 10 格内（测距机制和观察铁傀儡相同）还存在另外 4 个产生意愿的村民（合计 5 名村民为一组单位），铁傀儡就会成功生成。

铁傀儡生成后，该村民及其周围 16 格内的所有其他村民会进入 30 秒的冷却时间，时间过后它们会继续观察周围是否存在铁傀儡。

在 Java 版中，村民生成铁傀儡的过程与其职业和上次工作的时间无关。

在 Java 版中，如果村民每受伤 2 次（包含鸡蛋和雪球这种无伤害

的受伤状态）或处在惊慌状态下，也会每 5 秒触发生成铁傀儡的传播言论，这种情况下只需要 3 名村民而非 6 名为一组单位，这个传播言论一样会受到 16 格范围内的铁傀儡侦测而受到冷却 30 秒的限制。

- 惊慌状态

村民会在遇到僵尸、僵尸村民、尸壳、溺尸、僵尸疣猪兽、灾厄村民、凋灵、恼鬼和劫掠兽时进入惊慌（panicking）状态并迅速逃跑，有时村民也会躲进房屋。村民观察到这些生物的前提是其视线和该生物之间不能有障碍物遮挡，且其直线距离（两足足底中点的距离）必须在一定范围内。

综上所示，我的世界中村民的有限状态机如下图所示：

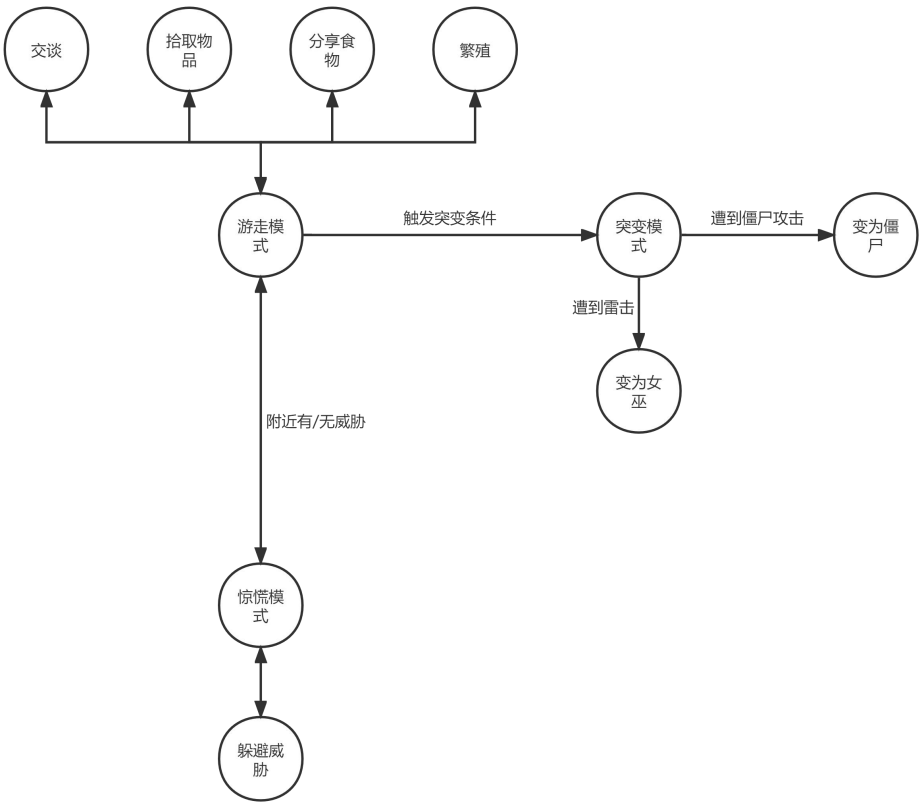

图 4：我的世界村民有限状态机

附录

附录 1

有限状态机伪代码

```
/**

*FSMState：

*/

public abstract class FSMState{

	public FSMAIControl m_parent;

	public int m_type;



	public abstract void Enter();		//状态进入时执行动作

	public abstract void Exit();		//状态退出时执行动作

	public abstract void Update();	//游戏主循环中状态的内部执行机制

	public abstract void Init();		//状态的初始化

	public FSMState CheckTransition();	//状态转移判断

}



/**

*FSMMachine

*/

public class FSMMachine{

	private ArrayList<FSMState> m_states;

	private FSMState m_currentState;
```

```
        private FSMState m_defaultState;

        private FSMState m_goalState;

        private int m_goalID;



        public void UpdateMachine();                    //更新状态机状态

        public void AddState(FSMState state);           //给状态机添加状态

        public void SetDefaultState(FSMState state);//设置默认状态

        public void SetGoalID(int goal);            //设置目标状态

        public void TransitionState(int goal);          //状态转移

        public void Reset();                    //状态重置

}



/**

*FSMAIControl

*/

public class FSMAIControl{

        public …//游戏感知数据

        private FSMMachine m_machine;



        public void Update();

        public void UpdatePerceptions();

        public void Init();
```

```
}
```