第 II 部分
在游戏中使用 AI 的方式

第3章

玩游戏

当大多数人想到游戏中的 AI 时候，他们会想到一个玩游戏的 AI，或者控制你在游戏中遇见的非玩家角色。这可能是因为人工智能与自动行动的概念之间的关联，或者是游戏角色与机器人之间的联系。虽然玩游戏对游戏中的 AI 来说远远不是并不是唯一引人注目的应用，但它是非常重要的一个应用，也是历史最悠久的一个。许多用于内容生成（第 4 章）与玩家建模（第 5 章）也依赖于玩游戏的方法，因此在内容生成与玩家建模之前讨论玩游戏是很有意义的。

本章主要介绍用于玩游戏的 AI 方法，包括用来在游戏中创建有趣的非玩家角色的方法。在赢得游戏的同时，表现的类似人类以及提供娱乐是非常不同的目标，它们面临着许多共同的挑战。事实上，关于一个人为什么愿意去使用 AI 方法来玩游戏存在许多不同的理由。我们首先讨论了这些多种多样的动机（章节 3.1）。不管为什么你想要使用 AI 来玩一个游戏，你可以使用什么方法来高效地玩游戏是由游戏多种多样的特性所决定的，这反过来也会影响 AI 方法的选择与设计。所以在本章的下一个章节（章节 3.2）将根据几种标准来描述游戏与 AI 算法的特征。一旦你已经充分地理解了你的游戏，你就可以做出一个明智的，关于使用什么算法的选择。接下来的章节（章节 3.3）主要探讨了可以用来玩游戏的多种方式，以及方法的正确选择如何依赖于游戏的特性。这里探讨的大部分方法在第 2 章中会被简要地并且略微抽象地探讨，但是本章将会深入地探讨这些方法在玩游戏上的应用。

接下来，一个长章节（章节 3.4）根据游戏种类对游戏的空间进行了划分，并且探讨了 AI 方法如何在多种游戏类型中得到应用。这个章节将会包含大量来自于文献的案例，还有一些来自于已经发布的游戏。这个章节也介绍了几种常被使用的基于游戏的框架以及用于测试 AI 游戏算法的竞赛。在本章中，我们将主要探讨 AI 方法在取胜方面上的使用，但是也大量涉及了玩游戏时的体验创造方面。

3.1 为什么使用 AI 来玩游戏？

关于为什么你想要部署某种形式的人工智能来玩一个游戏的问题可以被精简为两个更具体的问题：

AI 玩游戏是为了取胜吗？

这里的的问题是在游戏中取得尽可能高的性能是否是 AI 方法的首要目标。这里的高性能意味着得到一个高的分数，战胜对手，长时间地存活或者类似的目标。并非每一次都能定义“高性能”以及“在游戏中获胜”代表了什么——例如，《*模拟人生*》(Electronic Arts, 2000)并没有明确的胜利状态以及《*我的世界*》(Mojang, 2011)中的胜利条件与玩家玩的如何并没有强

烈的关系——但非常多的游戏，从《俄罗斯方块》Tetris (Alexey Pajitnov and Vladimir Pokhilko, 1984) 到《光环》(Microsoft Studios, 2001– 2015)系列，都直接地怎样意味着玩的优秀。然而，并不是所有的玩家都是为了取胜，而且很少有玩家可以在每个游戏中都获得胜利。玩家可以通过玩游戏消磨时间，放松，测试新的战略，探索游戏，角色扮演，维护他们的朋友团队等（在第 5 章有一个在这个主题上更为详细的探讨）。一个 AI 算法在玩游戏玩得好之外还可能被用于许多角色。例如，智能体可能以一种类人的方式玩游戏，以一种有趣的方式玩游戏，或者表现的可以预测。很重要的是要注意，为在游戏中取胜而优化智能体可能与玩游戏的方式不一致 :许多高性能的 AI 智能体以完全非人类，枯燥以及/或者无法预测的方式进行游戏，正如我们将在一些案例研究中看到的那样。

AI 是否扮演了一个人类玩家的角色？

某些游戏是单人游戏，而某些游戏是所有玩家都为人类的多人游戏。对于传统的棋盘游戏来说尤为如此。但许多，也许大多数视频游戏包含了多种多样的非玩家角色。这被计算机软件以某种形式控制着——事实上，对于许多游戏开发者来说“游戏 AI”就是指控制 NPC 的程序代码，无论这个代码多么简单或者复杂。很明显，在不同的游戏间，以及同个游戏内，NPC 的角色差异非常大。在本章的探讨中我们将非玩家角色指为那些人类不能胜任，或者不希望担当的角色。因此，在如《*反恐精英*》(Valve Corporation, 2000)这样专门的多人第一人称设计游戏（FPS）中，所有角色都是玩家角色，而在典型单人角色扮演游戏（RPG），如《*上古卷轴 5:天际*》 (Bethesda Softworks, 2011)中，仅有一个玩家角色，其他的都是非玩家角色。而一般来说，非玩家角色比起玩家角色有着更为限制的可能性。总之，AI 可以是为赢得一个游戏，或是游戏体验而玩游戏，无论是通过担当玩家角色还是担当非玩家角色。如图 3.1 所示，这产生了让 AI 玩游戏的四个核心用途。考虑到这些差异，我们现在来更为细致地看看构建玩玩游戏的 AI 的四个关键动机。

3.1.1 作为玩家角色取胜

可能在学术环境下最常见的与游戏结合的 AI 用途就是在游戏中取胜，同时扮演一个人类玩家的角色。当使用游戏作为一个 AI 测试台时尤为普遍。游戏已经在一个很长的时间内被用于测试 AI 算法的能力与性能，就像我们在章节 1.2 中探讨的那样。许多在 AI 以及游戏研究上的里程碑都采用了某种类型的 AI 程序在一些游戏中击败世界上最好的人类玩家的形式。例如可以参见，IBM 的深蓝在国际象棋上战胜 Garry Kasparov，谷歌 DeepMind 的 AlphaGo 在围棋上战胜李世石[628]以及柯杰，以及 IBM 的 Watson 在 Jeopardy!中获胜[199]。所有的这些都是高度公开的事件，被视为对 AI 方法日益增强的能力的确认。正如第一章讨论的，AI 研究者正在逐渐地转向视频游戏来为它们的算法寻找合适的挑战。与 IEEE CIG 与 AIIDE 会议有关的活跃竞赛的数量就是对这个的证明，就像 DeepMind 与 Facebook AI Research 选择将《*星际争霸 II*》(Blizzard Entertainment, 2015)作为他们的研究的测试平台那样。

出于很多原因，游戏是人工智能极好的测试平台，就像章节 1.3 阐述的那样。一个重要的理由是，游戏是创造来测试人类智能的。精心设计的游戏锻炼了我们的多种认知能力。在玩游戏时我们获得的许多乐趣来自于在玩游戏时学习游戏[350]，意味着精心设计的游戏也是优秀的教导者。这反过来又意味着它们提供了一种渐进的技能进步，允许在不能的能力水平上测试 AI。

除了 AI 基准之外，还有一些原因让人们愿意在应当由一个人类在游戏中取胜的位置使用一个 AI。例如，在一些游戏之中你需要强大的 AI 来为玩家提供一个挑战。这包括了许多

完美信息的策略游戏，例如传统的棋盘游戏，包括国际象棋，国际跳棋与围棋。然而，对于有隐藏信息的游戏，通过简单地“作弊”，提供挑战通常会变得更容易，例如，通过允许 AI 玩家访问游戏的隐藏状态甚至是修改隐藏状态来使得它对于人类来说更难应付。例如，在策略游戏《*文明*》(MicroProse, 1991)中，所有的文明可以被人类玩家操控。然而，在与人类相同条件下玩任何文明游戏都很具有挑战性，并且据我们所知，没有 AI 能够将这些游戏玩的与人类一样好。因此，当和几个电脑控制的文明对战时，游戏一般会通过各种方式为它们提供优势条件进行作弊。

以胜利为目的的 AI 的另一种使用情况是作为一名玩家角色来测试游戏。当设计一个新的游戏，或者一个新的游戏关卡时，你可以使用一个玩游戏的智能体来测试这个游戏或者关卡是否是可玩的，也称为**基于模拟的测试**(simulation-based testing)。然而，在许多情况下你希望智能体也能以一种类人的方式来玩游戏，以确保测试更具有相关性；参见下面的“为体验而玩”的部分。

在历史上，使用 AI 以一个玩家角色来取胜在学术工作中占据着主导地位，一些研究者甚至不曾考虑过 AI 在玩游戏中的其他角色。但在另一方面，在游戏开发中，玩游戏的 AI 的这种特别动机甚至更为稀有；在现有的游戏中的大部分玩游戏的 AI 专注于非玩家角色与/或为体验而玩。这种不匹配长久以来造成了游戏产业中工业界与学术界之间对彼此理解的缺乏。然而在近几年，对于玩游戏的 AI 的多种动机的理解已经得到了一种提升。

3.1.2 作为非玩家角色取胜

非玩家角色经常被设计为不提供最大的挑战，或者是尽可能地高效，而不是变得有趣或者类人；参见下面的“非玩家角色为体验而玩“部分。然而，也有某些你希望一个非玩家角色尽可能强地去玩游戏的情况。就像前面提到的，如《*文明*》(MicroProse, 1991)这样的策略游戏对于高性能的无作弊对手有一个（未被回答的）需求，尽管在这里我们在探讨其他人类玩家原则上可以担当的游戏角色。其他策略游戏，例如《*幽浮:未知敌人*》(2K Games, 2012)，有着人类不能扮演的游戏角色，为以取胜为目的的 NPC AI 创造了一种需求。

其他时候，创造一个用于赢得胜利的 NPC 是为游戏体验而创造 NPC 的一个必要前提。例如，在一个赛车游戏中，你可能想要实现“橡皮圈 AI”，在这里 NPC 车辆将会根据人类玩家调节它们的速度，所以它们将永远不会过于落后或者超前。这做起来很容易，但只是在你已经有了一个可以很好地进行游戏的 AI 控制者时才可以，无论是通过真正地玩游戏或是不容易被发现的方式进行作弊。控制者的表现可以在需要时降低，以此来匹配玩家的表现。

3.1.3 作为玩家角色为体验而玩

为什么你会想要一个智能体来扮演人类玩家的角色，但却不关注胜利呢？举个例子来说，当你想要一个**类人智能体**(human-like agent)时。这样子的智能体最重要的原因可能就像上面提到的那样：基于模拟的测试。这在手动设计游戏与游戏内容，以及在程序化地生成内容时都很重要；在后一种情况中，游戏内容的质量经常在一个玩游戏的智能体的帮助下自动地评估，就像在第四章讨论的那样。当试图了解某个游戏将会如何被一个人类游玩时候，以类人的方式游玩游戏的智能体就显得很重要，意味着它有着可以与人类相媲美的表现，有着相似的反应速度，与人类一样会犯同样种类的错误，像一个人类那样会好奇并探索同样的区域，等等。如果 AI 智能体的玩法与人类的玩法有着非常显著的区别，它可能会给出错误的信息，例如关于一个游戏是否是可以获胜的的（它可能是可以获胜的，但只在你拥有超越人类的反射时），或者一个游戏机制是否会被用到（可能一个人类会使用它，但一个尝试采用最佳玩

法的 AI 不会）。

另一种必须要类人的玩法的情况是当你希望向一个人类玩家展示如何进行一个关卡时。各类游戏的一个普遍功能是某种类型的**演示模式**，其手把手展示了这个游戏。一些游戏甚至拥有一个建立在核心游戏模式中的演示功能。例如，任天堂 Wii 上的《*新超级马里奥兄弟*》(Nintendo, 2006)将会给你展示如何玩一个关卡的一个特定部分，如果你重复地在这里失败。游戏简单地接管控制并为你游玩大约十到二十秒，之后再让你继续。如果所有的关卡内容都是事前知晓的，并且不存在其他玩家，这样的演示可以被硬编码。如果游戏的某些部分是用户设计的，或者程序化地生成的，这个游戏需要由它自己生成这些演示。

以一个"类人"的方式进行游戏似乎是一个有些模糊与主观的目的，并且的确是这样的。存在非常多的一个典型的 AI 智能体与一个典型的人类玩家在玩法上的不同。人类与 AI 如何不同取决于用于玩游戏的算法，游戏本身的性质，以及许多其他因素。为了进一步调查这些差异，促进能够以一种类人方式进行游戏的智能体的发展，已经有两种不同的类似图灵测试的竞赛被举办了。2K BotPrize 举办于 2008 年到 2013 年，并且要求竞赛者来开发可以游玩 FPS《*虚幻竞技场 2004*》（Epic Games, 2004）的智能体，并且要让人类选手认为这些机器人是人类[261, 260, 647]。同样，马里奥 AI 竞赛的图灵测试赛也让人们提交《*超级马里奥兄弟*》(Nintendo, 1985)的游戏智能体，其会被人类观众判断它们是人类还是非人类[618, 717]。在这里遍历这些竞赛的所有结果将会占据太多时间，但对于许多类型的 AI 智能体来说在游戏过程中仍然会出现一些非常明显的非人性化的信号。这包括欧了拥有极端迅速的反应，比起一个人类可以更快地在动作间转换，不会尝试失败的动作（因为拥有一个太好的关于动作导致的结果的模型。），不会做不必要的动作（例如在一个人可以只跑步时跳跃）并且不会犹豫或者停下来思考。

当然，并不是所有的玩家都以相同的方式玩游戏。事实上，正如第五章中进一步探讨的那样，如果分析任何游戏中的一组玩家轨迹，可以经常找到许多种玩家"原型"或者"人格"，例如，以显著不同的方式进行游戏的玩家的聚类可以用攻击性，速度，好奇性与技巧来划分。在以类人风格玩游戏的 AI 上的工作中，有同时学习与模仿个体玩家的游戏风格的工作[421, 422, 510, 326, 602]，也有学习以一个或者多个人格的风格进行游戏的[265, 267]。

3.1.4 作为非玩家角色为体验而玩

几乎可以肯定，在游戏业界中对于玩游戏的 AI 最普遍的目标是让非玩家角色行动起来，并且几乎都是不将击败玩家或是"获胜"游戏放在首位的方式（对于许多 NPC 来说，甚至没有定义获得胜利意味着什么）。NPC 在游戏中可能以多种，有时重叠的意图存在 :扮演敌人，提供协助与引导，作为一个难题的组成，来叙述一个故事，为游戏的动作提供一个背景，来情绪化地表达等等[724]。NPC 的复杂性与行为复杂度也会有巨大的变化，可以从《*太空侵略者*》(Midway, 1978)中常规的左右移动以及《*超级马里奥兄弟*》 (Nintendo, 1885–2016)系列中的库巴，到《*生化奇兵：无限*》(2K Games, 2013)中非玩家角色细微与多样的行为以及《*异形:隔离*》(Sega, 2014)中的外星人。

根据 NPC 的角色，可以对控制它的 AI 算法提出非常不同的任务（可以肯定的是，许多控制 NPC 的脚本在任何常规方式中都并不能被真正地描述为人工智能，但我们将继续在这里使用这个缩写，因为它在游戏工业界被普遍用于所有控制非玩家角色的代码）。在许多情况中，游戏设计者所寻求的是**智能的幻象**：让玩家相信 NPC 在某种程度上是智能的，即使控制它的代码是非常简单的。之前的章节探讨的这种情形中的类人性可能是也可能不是这里的目标，这要取决于它是什么类型的 NPC 的（一个机器人或是一只龙可能不应当以过于类人的方式表现）。

在其它情况下，一个 NPC 最重要的特征是它的可预测性。在一个典型的潜入游戏中，玩家挑战的一个重要部分就是记住与预测守卫与其它应当避免的角色的规律。在这种情况中，完全规律的巡逻也是有意义的，这样它们的时间表才能被玩家所收集。同样的，在许多游戏中的 BOSS 怪物也被设计为以一个序列重复确定的移动，并且只在动画循环中的固定阶段可以被玩家的攻击伤害。在这样的案例中，过于"智能"并且适应性的行为将会与游戏设计不相容。

要注意的是，即使在你期望需要 NPC 有灵活并且复杂的动作时的情况时，一个为取胜而玩的智能体可能是非常有问题的。许多高性能的可能被玩家视为非常**无聊**，也是"不光明正大的"的行为的首要例子。举例来说，在为一个回合制策略游戏构建高性能 AI 的一个实验中，发现解决方案之一（基于神经进化）在对抗上非常无聊，因为它简单地采取了一个防御位置并且使用长距离袭击来攻击任何进入的单位[489]。类似的，*扎营行为*(camping)（固定在一个受保护的位置并且等待敌人将自己暴露在火力中）是一个在 FPS 游戏中通常让人皱眉并经常被禁止的行为，但它通常对于 AI 来说非常有效并且容易学习（顺带地说，显示生活中的军事训练经常强调类似扎营行为的战术——有效的往往都不有趣）。另一个有趣的案例是 Denzinger 等人的工作[164]，在其中进化算法发现在《*FIFA 99*》(Electronic Arts, 1999)中进球的最佳方式就是强迫一次罚球。这个进化过程找到了适应度上的一个局部最佳值，对应了一个最佳击球点或是利用了可以产出高效率的游戏机制，也是可以预测但无聊的游戏体验。利用游戏的漏洞来胜利是一个不仅被 AI 利用也被人类玩家利用的创造性战术[380]。

3.1.5 关于 AI 在游戏过程中的目标与角色的总结

我们在上面讨论到，以玩家身份来在游戏中获得胜利已经被过分强调了，在许多的学术研究中，到了忽略其他视角的地步。与此同时，工业界在 AI 上的工作也逐渐地过分强调作为一个非玩家角色为了游戏体验而进行游戏，也到了忽略其他视角的地步。这导致了在工业界中在行为编辑方法上的强调，如有限状态机与行为树，因为基于搜索，优化与学习的 AI 方法已经被视为不利于游戏体验；一种共同的抱怨是这样的方法在可预测性与行为控制上有着明显的缺乏。然而，给予一种更好的关于 AI 可以充当的角色可以如何在游戏中被利用的理解，这种对于方法与视角的忽略有希望在学术界与工业界都迎来一种结束。

3.2 游戏设计与 AI 设计的考量

当选择一个 AI 方法用来玩一个特定的游戏时（以任何在章节 3.1 中探讨的角色），了解你要玩的游戏的特性以及你用于设计的算法的特性是非常关键的。这共同决定了什么样的算法会是高效的。在这个章节，我们首先探讨由于游戏本身的特点而面临的挑战（章节 3.2.1），然后我们探讨了 AI 算法设计的各个方面，其需要独立于我们思考到的游戏之外来考虑。

3.2.1 游戏的各类特性

在这个章节中我们探讨了多种游戏的特性，以及它们在 AI 方法的可能使用方式上的冲击。所有包含的特性都与游戏的设计有关，但少部分（例如输入表示与前向模型(forward model)）也依赖于游戏的技术实现并且容易改变。我们的探讨的许多部分受到了 Elias 等人的书籍《Characteristics of Games》的启发[190]，其从游戏设计的角度探讨了这些因素中的许多部分。为了更好的展示，图 3.2 将许多核心游戏案例放入了**可观察性**，**随机性**与**时间粒度**(time granularity)的三维空间内。

3.2.1.1 玩家数目

一个很好的开始角度是一个游戏拥有的玩家数目。Elias 等[190]区分为：

- **单人**游戏，例如解密与计时竞速。
- **一个半玩家**的游戏，例如带有比较复杂的 NPC 的某个 FPS 的战役模式。
- **双人**游戏，例如国际象棋，国际跳棋以及《*太空大战*》(Russell, 1962)等。
- **多人**游戏，例如《*英雄联盟*》(Riot Games, 2009)，《*马里奥赛车*》(Nintendo, 1992–2014)系列以及大多数 FPS 游戏的在线模式。

单人游戏与一个半玩家的游戏之间的区分并不是非常明显——并没有明显的边界用来区分如何将高级的 NPC 计算为“半个玩家”。在多人游戏的案例中，有许多可以只需要两个玩家就可以进行的，在这一点上它们可以被有效地定为是双人游戏。其它玩家并不一定总是敌对并且会试图阻止玩家的——也有许多协作的游戏，或者玩家间的关系是复杂的，并同时存在竞争与合作因素的游戏。但将玩家的数目记在脑海中在思考用于进行游戏的算法时候仍然是非常有用的。，

当使用树搜索算法来玩游戏时候，某些算法特别地适合某些数目的玩家。标准的单智能体搜索算法，例如宽度优先，深度优先以及 A*特别地适合单人的情况（包括有着 NPC，但是那些 NPC 非常简单并且好预测，可以作为环境的一部分来对待）。在这样的游戏中，游戏中将会发生什么完全取决于玩家采取的动作以及任何可能的随机效果；不存在其他“有意图”的玩家。这与单智能体树搜索算法非常适合，其基于**马尔科夫性质**，也就是下一个状态完全取决于之前的状态以及当时采取的动作。

一种特别的情况是**双人零和对抗游戏**(two-player zero-sum adversarial games)，也就是正好存在两个玩家；一个玩家将会胜利，而另一方将会失败（或者可能会成为平局）。我们不知道其他玩家会做什么，但是我们可以肯定地认为她将会尽其所能来获胜，以及阻止你取得胜利。极大极小算法（伴随或者不伴随 α-β 剪枝）非常完美地适合这样的情况，并且给定充足的计算时间将会得出最佳的玩法。

但当我们拥有**许多玩家**时候如何解决这个挑战呢，或者是一个单独的玩家被非常复杂的非玩家智能体包围时候呢？尽管将极大极小算法拓展到多人玩家在理论上可行，这只能在玩家之间不存在任何合谋（或者任何类型的联盟）时以及游戏的零和性质仍然存在时起作用（其通常并不会）。除此之外，极大极小算法的计算复杂度在超过两个玩家时会很快地变得无法处理，因为对每个你采取的移动来说，不仅是需要考虑一个玩家的反制，而是全部玩家的反制。所以这个方法很少是可行的。

在多人情况中更普遍的是将游戏视为一个单人游戏，但是使用某种类型的关于其他玩家会怎么做的模型。这可以是一种关于其他玩家将会如何对抗玩家的猜想，一种基于观察过的行为的学习模型甚至是一种随机模型。有着关于其他玩家将会做什么的模型，许多标准的单人玩游戏的方法可以在多人的情况下使用。

3.2.1.2 随机性

一种常见的，许多游戏违反了马尔科夫性质的方式是通过称为**随机的**（或者非确定性的）。在许多游戏中，发生的一些事是随机的。就像标准的电子计算机架构不允许“真正的”随机性那样，有效的随机性也由伪随机数生成器所提供。“随机(stochastic)”这个词被用于表示无法

被实际预测的过程，无论它们来自于真正的随机性或是复杂的计算。游戏可以具有不同的随机性数量，从完全确定性的游戏，像国际象棋，到完全由随机性所支配的游戏，像轮盘赌，鲁多游戏(Ludo)，快艇骰子(Yahtzee)甚至大富翁。很普遍地，有着部分或者完全确定性的游戏与一些随机因素相结合，通过抽牌，扔骰子或者一些类似的机制来减少规划的可能性。然而，随机性在本质上可以发生在游戏的任何部分。

在一个有着随机性的游戏中，游戏的结果并不完全由玩家采取的动作所决定。换句话说，如果你玩了同样的游戏的几次游戏过程，在同样的时间点采用同样的动作，你不能保证同样的结果。这对于 AI 算法也有影响。对于**树搜索**算法，这意味着我们不能保证一个动作序列将会导致的状态，因此也不能保证算法的结果。这导致了以它们的规范形式使用许多树搜索算法时出现的问题，并且需要我们添加一些修改来应对在前向模型中的不固定的不确定性。例如，在蒙特卡罗树搜索的修改中，如**确定化**(determinization)，其中每个动作不同的结果会被单独地探索[76]。尽管这些算法变种可以是有效的，它们通常会增加了基本算法的计算复杂度。

对于强化学习方法，包括进化强化学习(evolutionary reinforcement learning)，它意味着我们已经降低了一个给定的战略/策略可以准确地说有多优秀的确定性——由于游戏中的随机事件，一个好的策略可能会达到坏的结果，或者一个坏的策略有好的结果。这样的结果不确定性可以通过多次评估每一个策略来减轻，尽管这有着明显的计算开销。在另一方面，随机性有时候在学习策略时实际上可以成为一种优势：一个从随机游戏学习到的策略可能比从一个确定性的游戏学来的更为健壮，因为在后者中很可能会学习一个非常脆弱的策略，其只适用于游戏的某个特定配置。例如，学习了一个在特定时间在特定位置攻击敌人的策略，而不是变成可以解决在任意时间来自任何角度的敌人。

尽管对电子游戏来说包含某种形式的随机性是非常普遍的，但一个有趣的情况是，早期的游戏硬件，例如 1977 年的雅利达 2600，其并没有用于实现伪随机数生成器的功能（主要是因为它缺乏一个系统时钟）。如果一个玩家任务在完全相同的时间采取完全相同的动作（包括开始游戏时按下的按键），将会出现完全相同的结果。Arcade Learning Environment 是一个被广泛使用的基于游戏的 AI 的基准，围绕雅利达 2600 的模拟器而构建[39]。在训练玩不带有随机性的游戏的 AI 智能体时，完全有可能学习到会有效地绕开整个游戏的复杂性的脆弱策略（无论这是否真的发生，或者大多数智能体是否学习了更通用的策略，是一个开放性的问题）。正如我们已经看到的第二章的多种案例，《*吃豆人小姐*》(Namco, 1982)可以说是那个时代最流行的非确定性的街机游戏。

3.2.1.3 可观察性

可观察性是一个与随机性密切相关的特征。它是指有多少关于游戏状态的信息可供玩家使用。在一个极端，我们有着例如国际象棋，围棋与国际跳棋这样的经典棋盘游戏，其中完整的棋盘信息对玩家来说一直都是可用的，还有例如数独游戏与单词解谜这样的猜谜游戏。这些游戏有着**完美信息**(perfect information)。在其他极端我们可以思考经典的文字冒险，例如《*魔域帝国*》(Personal Software, 1980)或者《*巨穴历险*》，其中世界的极少部分以及它的状态在刚开始时被显露给玩家，而游戏的许多部分是关于探索这个世界是怎样的。这些游戏有着**隐藏信息**(hidden information)并因此只有**部分可观察性**。如果不是绝大部分，但也是有很多的计算机游戏有着明显隐藏信息：思考一个典型的平台游戏，例如《*超级马里奥兄弟*》(Nintendo, 1985)，或者像《*光环*》系列(Microsoft Studios, 2001–2015)这样的 FPS，在其中的任何时刻你只能感知到游戏世界的一小部分。在计算机策略游戏，例如《*星际争霸 II*》(Blizzard Entertainment, 2010)或者《文明》(MicroProse, 1991)中，用于隐藏信息的普遍术语是*战争迷*

雾。甚至许多经典的非电子游戏也拥有隐藏信息，包括了大部分玩家可以保证他们的手牌是私密的卡牌游戏(card games)（例如扑克），以及像海战棋这样的棋盘游戏。

在开发一个用于有着隐藏信息的游戏的 AI 智能体时，可以遵循的最简单的方法就是全然忽略隐藏信息，在每一个时间点，只为智能体提供可用的信息并且使用其来决定接下来的行动。这样做实际上在某些游戏中运作的非常好，特别是有着线性关卡的以动作为中心的游戏；例如，简单的《*超级马里奥兄弟*》(Nintendo, 1985)关卡只基于瞬间的可用信息就可以被玩的很好[706]。然而，如果你只基于可用信息来玩一个如《*星际争霸 II*》(Blizzard Entertainment, 2010)的策略游戏，直到事情已经变得太迟之前，你甚至不会见到敌人——优秀的玩法涉及到了积极的信息收集。即使在《*超级马里奥兄弟*》(Nintendo, 1985)中，具有回溯的复杂关卡也需要记住关卡在屏幕之外的部分[320]。在扑克这样可以进行欺骗的游戏中，可用的信息（你自己的手牌）在实际上相对来说并不重要；这个游戏的核心是为隐藏信息（你对手的手牌与思路）建模。

因此，有效的用于有着部分可观察性的游戏的 AI 经常需要某种形式的**对隐藏信息的建模**。对于一些游戏，特别是如双人有限注德州扑克(Heads-up limit hold'em)这样扑克的变体，在为包括对手玩法的隐藏信息建模的特定游戏方法上已经有着可观的研究了[62]。也有更为通用的，将某种形式的隐藏状态建模添加到现存算法中去的方法，例如信息集蒙特卡罗树搜索(Information Set Monte Carlo tree search)[145]。就像出现在用于解决随机性的方法中那样，这些方法比起基础算法一般会增加可观的计算复杂性。

3.2.1.4 动作空间与分支因子

当你玩极简主义以及自虐主义的手游《*Flappy Bird*》(dotGEARS, 2013)时，你在任意的时刻点都只有单种选择：飞或者不飞。（飞通过触摸屏幕来完成，并且使得主人公鸟在空中升起。）《*Flappy Bird*》以及类似的单键游戏，例如《*屋顶狂奔*》(Beatshapers, 2009)，可能有着最低的分支因子。分支因子是在任意决策点时你可以采取的不同的动作的数目，Flappy Bird 的分支因子是 2：飞或者不飞。

作为比较，《*吃豆人*》(Namco, 1980)有着一个为 4 的分支要素：上，下，左以及右。《*超级马里奥兄弟*》(Nintendo, 1985)有着大约 32 的分支要素：八个方向键上的方向乘以两个按钮（尽管你可能认为这些组合的部分是没有意义的并且实际上不应该被考虑）。国际象棋平均有着 35 的分支因子，而国际跳棋有着一个稍低的分支要素。围棋在第一步有高达 400 的分支要素 ；随着棋盘被填充，分支要素减少，但在落每个子时通常都有数百个可能的位置。

尽管比起 35 或者 2，400 是一个非常高的分支要素，它比起许多每个回合可以有多个单位被移动的计算机策略游戏来说仍然显得矮小。将个体单位的移动的每种组合考虑为一种动作，这意味着这个游戏的分支要素是个体单位的分支要素的乘积。如果你有着 6 个不同的单位，每个单位在一个给定时间可以采取 10 种不同的行动——一个相对比起典型游戏，如《*星际争霸*》(Blizzard Entertainment, 1998) 或《*文明*》(Micro- Prose, 1991)来说相对保守的估计——那么你的分支要素是一百万！

但是稍等，它变得更糟了。对于许多游戏来说，它甚至不可能枚举所有的动作，因为输入空间是**连续的**。想想任何现代的在计算机或是主机上玩的第一人称游戏。尽管计算机的确无法真正很好地捕捉到无限的信息，并且这种“”连续的“输入，例如计算机鼠标，触摸屏和摇杆（例如在 Xbox 与 Playstation 控制器上），实际上返回了一个电子数字，这个数字有着很好的精度，其事实上就是连续的。创造一个事实上可以枚举的动作集合的唯一方法是在某种程度上离散这个连续的输入空间，并且在压倒性的分支要素或者大量减少输入空间导致不能有效地进行游戏之间达成一种妥协。

分支要素是**树搜索**算法的有效性的一个关键性的决定因素。宽度优先算法（用于单人游戏）以及极大极小算法（用于对抗性双人游戏）在搜索到深度d的复杂度是b^d是，其中b是分支因子。换句话说，一个很高的分支要素使得搜索几乎不可能超过几步。这个事实对于可以使用树搜索玩的游戏来说有着非常明显的后果；例如，围棋有一个比国际象棋高一个数量级的分支因子，并且这也可以说是过去数十年间围棋上的所有类型的 AI 算法的不佳表现的主要原因（与此同时相同的方法在国际象棋上表现良好）。蒙特卡罗树搜索很好的解决了高分支因子，因为它构建了不平衡的树，但是这不意味着它免疫了这个问题。一旦分支因子变得足够高（例如说，一百万，也可能是十亿，取决于模拟器的速度），甚至列举深度 1 处的动作也变得不切实际，因此完全不能使用树搜索。

高分支因子对于强化学习算法来说也是一个问题，包括了进化强化学习。这大部分与控制器/策略表示有关。如果你正在使用一个神经网络（或者某些其他的函数逼近器）来表示你的策略，你可能需要为每个动作提供输出；或者，如果你使用网络来为所有动作赋值，你需要完全遍历它们。在这两种情况下，一个巨大的可行动作的数目也带来了一个成本。另一个问题是探索-开发困境(exploration-exploitation dilemma)：可行动作的数目越大，在学习时全部探索它们要花费的时间就越久。

关于分支因子的最后一个评论是，对于许多游戏来说，它们并不是一成不变的。在国际象棋中，你在游戏的开始阶段有着更少的可用招式，此时你的棋子大部分都被封锁，接近游戏中局时会更多，而在终局时再次减少，此时大部分棋子可能被封锁。在一个典型的 RPG，例如在《*最终幻想*》系列(Square Enix, 1987–2016)中的那些，可用的动作的数目随着玩家角色累积物品，法术以及其他可能性而增加。就像上面所提到的，围棋中的可用动作的数目随着你的游戏过程而减少。

3.2.1.5 时间粒度

当讨论上述的分支因子时，我们谈到了在任意"时间点"要采取的可行动作的数目。但是这有多经常？玩家多经常可以采取一个动作？一个基础的差别在于**回合制**与**即时游戏**之间。大部分经典的棋盘游戏是回合制游戏，在这一的游戏中，玩家相互轮流，并且在每个回合一个玩家可以采取一个动作，或者是某个特定数目的动作。回合间过去的时间的总量通常在游戏内不重要（尽管竞赛与职业玩家通常会包括某种形式的时间限制）。即时游戏包括了计算机游戏的流行类别，例如 FPS，竞速游戏以及平台游戏。即便在即时游戏中，一个游戏内的动作在实践上应当有多经常被执行也有着可观的差异。在一个极端，就是屏幕刷新频率；当前世代的视频游戏通常都会争取拥有一个每秒 60 帧的刷新频率来确保可以被感知上的平滑移动，但是由于渲染复杂场景的复杂度，许多游戏是通常一半的刷新频率甚至更少。在实践中，一个玩家角色每秒可以执行的动作数目（或者任意其他游戏中的角色）通常比这个更为限制。

为了举两个在时间粒度规模上相距甚远的案例，让我们考虑两个对抗性游戏：国际象棋与《*星际争霸*》(Blizzard Entertainment, 1998)。一局熟练玩家之间的国际象棋对弈平均持续 40 回合。在《*星际争霸*》(Blizzard Entertainment, 1998)，一个高度竞技性的即时策略类（RTS）游戏中，职业玩家通常每秒会执行三到五个动作（每个动作通常通过一次鼠标点击或者一个快捷键执行）。一场典型的游戏会持续 10 到 20 分钟，这意味着数以千计的动作在一个游戏中被执行。但在一局《*星际争霸*》(Blizzard Entertainment, 1998)的游戏中，并没有比一局国际象棋对弈中有着更多的明显事件——主导并不会经常改变，并且宏观策略决策也不会更频繁做出去。这意味着《*星际争霸*》(Blizzard Entertainment, 1998)里明显事件之间的动作数目

比国际象棋中的更高。

时间粒度通过限制你可以向前看多远来影响 AI 玩游戏的方法。一个给定的搜索**深度**根据游戏的时间粒度意味着非常不同的事情。国际象棋中的十回合已经足够执行一个完整的策略；《*星际争霸*》(Blizzard Entertainment, 1998)中的十个动作可能只是几秒，而在其中游戏可能没有任何显著的改变。为了使用树搜索来很好地玩《*星际争霸*》(Blizzard Entertainment, 1998)，一个人可能需要在成百或者上千的动作中有着一个优秀的搜索深度，这在计算性上显然是不可行的。解决这个挑战的一种方式是考虑**宏观行动**（例如[524, 523]中那样），其是更小的，有着良好细粒度的动作的序列的集合。

3.2.2 AI 算法设计的各种特性

在接下来，我们将探讨在将 AI 算法应用到游戏中时的一些重要问题。这些是与游戏设计（涵盖在前面的章节中）关联并不大的设计选择，而与 AI 算法设计以及算法的使用约束有关。这个章节在重点放在执行游戏的 AI 的情况下拓展了第 2 章中关于表示与效用的讨论。

3.2.2.1 游戏状态是如何表示的？

游戏在它们向玩家展示的信息，以及如何展示上是不同的。文字冒险输出文本，经典棋盘游戏的状态可以通过所有棋子的位置来描述，而图形视频游戏提供了图像，并伴随着声音与某些偶然的输出，例如手柄的振动。对于数字游戏，游戏在实现上使用的硬件的技术限制影响了它如何被表示；随着处理速度与内存容量增加，视频游戏的像素分辨率与场景复杂度也成比例地增加。

很重要的是，同一个游戏可以用不同的方式来表示，而它用哪一个方式来表达对于玩这个游戏的算法来说非常重要。以竞速游戏作为一个例子，算法可以从 3D 渲染的赛车的挡风玻璃中收到一个第一人称的视角，或者一个以 2D 渲染的赛道与多个车辆的俯视图。它也可以简单地接受在赛道参考系中所有车辆的位置以及速度的列表（连同轨道的模型），或者在赛道参考系中对其他车辆的角度与距离的集合（以及赛道边缘）。

关于输入表示的选择在设计一个游戏时意味着许多东西。如果你希望学习一个用于在某条赛道上驾驶的策略，并且对策略的输入是与速度以及到赛道左右边缘的距离有关的三个连续变量，学习一个适合的策略是比较简单的。然而如果你的输入是一个未处理过的视觉反馈——例如，成千上万的像素值——找到一个适合的策略可能会更为困难。在后一种情况中不仅策略搜索空间巨大的多，与有着适合的表现的策略对应的搜索空间的比例很可能也会小的多，因为更多无意义的策略成为了可能（例如，偶数像素比奇数像素更亮则左拐；这个策略没有以任何有意义的方式映射到游戏局面，即便它起作用了也只是侥幸而已）。为了基于视觉输入来学习驾驶——至少在照明，路边景色等显著变化的情况下——你可能需要学习某种类型的视觉系统。鉴于此，大部分应用到完整的视觉输入的原生策略很可能不会表现的非常好。继续车辆竞速的案例，即便在你有着非常少的输入的案例中，其如何被表示也存在问题；举例来说，如果输入是由车辆的参考系而不是赛道的参考系来表示的，将更容易学习到一个好的驾驶策略[707, 714]。一种在某种程度上更为容易理解的向神经网络表示低维度的输入的方式的探讨可以在[566]中找到。

近年来，一些研究人员集中于学习使用完整的视觉来作为输入的策略。例如，Koutnik 等人逐步发展了神经网络来通过高分辨率视频玩 The Open Racing Car Simulator (TORCS)[352]，Koutnik 等人使用评估的像素来作为一个深度 Q 网络的输入，其被训练来玩《*毁灭战士*》(GT Interactive, 1993)的某个版本[463]，还有 Mnih 等人训练了深度网络来使用

Q-learning 来玩雅利达 2600 游戏[463]。使用原始像素输入通常是以给予 AI 与人类能拥有的相同条件来作为目的的，从而实现人与 AI 之间的公平竞争。另一个动机是，如果你希望使用你的算法来玩一个“盒外”的游戏，不带有任何 API 或者额外的工程来暴露游戏的内部状态，你将可能不得不诉诸于使用原始的视觉反馈。然而，在你拥有游戏的源代码或者一个有效的 API 的情况时——就像你在为一个新游戏开发 AI 时几乎总是会有的那样——没有理由不去使用在无论什么情况下都会使得 AI 算法的任务更简单的“消化过的”游戏状态。而是否呈现人类玩家无法获取的信息，也就是“作弊”，是一个单独的问题。

3.2.2.2 是否存在一个前向模型？

在设计一个 AI 来玩某个游戏时一个非常重要的因素是是否存在一个游戏的模拟器，一个所谓的**前向模型**是否可行。一个前向模型是一个模型，其给定一个状态s与一个动作a，会达到状态s'，就像真实的游戏如果在s给定a会达到的那样。换句话说，这是在模拟中玩游戏的方式，所以多个动作的结果可以在实际地在真实游戏中执行这些动作的一部分之前被探索。为了能够使用任何基于树搜索的方法来玩一个游戏，拥有一个游戏的前向模型是必须的，因为这个方法依赖于模拟多个动作的结果。

一个正向模型令人满意的特性，除了它的存在之外，就是它是**飞快的**。为了能够有效地使用一个树搜索来控制一个实时游戏，通常需要能够比真实时间快至少一千倍来模拟游戏，最理想的是快数万或者数十万倍。

为棋盘游戏，例如国际象棋或者围棋构建一个前向模型是非常简单的，因为游戏状态简单地就是棋盘局面，并且规则非常容易编码。对于许多视频游戏来说，构建一个前向模型可以通过简单地复制（或者是重用）被用于控制游戏本身的相同代码来完成，但是不用等待用户输入或者显示图像，并且不需要执行所有涉及到图像渲染的计算。对于一些视频游戏来说——特别是一些最初实现在更老的硬件上的游戏，例如经典的实现于 8 位或者 16 位处理器上的街机游戏——前向模型可以被实现为比真实时间快很多，因为核心的游戏循环在计算上并不复杂。（这也是可能发生在某些现代游戏上，只要通过在可以将图形循环置换为替换代码的模拟器中运行它们。）

然而对许多游戏来说，是不可能或至少难以得到一个快速的前向模型的。对于大部分商业游戏，源代码是不开放的，除非你工作在开发这个游戏的公司。即便源代码是开放的，当前游戏工业界中的软件引擎实践也使得从游戏代码中抽取前向模型非常困难，因为核心控制循环与用户界面管理，渲染，动画，某些时候还有网络代码，是非常紧密地捆绑的。通过软件引擎实践中的改变，将核心游戏循环更为纯净地从多种输入/输出功能中剥离出来，让前向模型被更容易地构建，将会是视频游戏中的高级 AI 方法最重要的推动者之一。然而，在某些情况下核心游戏循环的计算复杂度仍然可能过高，使得任何构建于核心游戏代码的前向模型会变得太慢而无法使用。在该情况的一部分中，实际上可能可以构建以及/或者学习一个简化过的或是接近的前向模型，此时在前向模型中一系列动作导致的结果并不能保证与在实际游戏中相同的动作序列导致的状态相同。一个**近似的前向模型**是否是可以接受的取决于 AI 使用的特定用例与本意。要注意一个稍微低一点准确度的前向模型仍然可以是令人满意的。例如，当存在明显的隐藏信息或者随机性时，AI 设计者可能不想要提供给 AI 智能体一个可以让隐藏信息变得可观察的预言并且告诉智能体哪一个随机动作将会发生。采取这样的一个设计决策可能导致不合理的优秀表现以及作弊的浮现。

当一个前向模型不能被生成时，**树搜索**算法不能被应用。仍然有可能手动地构建智能体，并且也可以通过**监督学习**或者**强化学习**的某些形式来学习智能体，例如时序差分学习或者进化强化学习。然而请注意，虽然强化学习方法通常不需要一个完整的前向模型，也就是在任

何状态下采取任何动作的结果都可以被预测，它们仍然需要一种方式来比真实时间更快的运行游戏。如果游戏不能明显的超越它在自然游戏情况下的步调，这将会导致算法要一个非常长的时间来学习玩法。

3.2.2.3 你是否拥有时间进行训练？

在人工智能中一个粗略但有用的区分，就是通过检查可能的行动和未来状态来决定**在一个给定的情况下应当如何做**的算法——大致上就是多种类型的树搜索——以及随着时间而**学习一个模型**（例如一个策略）的算法——例如机器学习。同样的区分存在于用于玩游戏的 AI 之中。有已经被研究过的，不需要学习关于游戏的任意知识，但是需要一个前向模型的算法（树搜索）；也有不需要一个前向模型，而是学习了一个像是从状态映射到动作的策略的算法（基于模型的强化学习）；而且也有同时需要一个前向模型以及训练时间的算法（基于模型的强化学习以及带有自适应超参数的树搜索）。你想要使用什么类型的算法很大程度上取决于你使用 AI 来进行游戏的动机。如果你正在使用游戏来作为你的 AI 算法的测试平台，你的选择将会由你正在测试的算法类型所决定。如果你正在使用 AI 来在你开发的游戏中解决玩家体验——例如，作为一个非玩家角色——那么你很可能会不希望 AI 在游戏正在进行时表现出任何学习能力，因为这要冒着干扰制作者设计的游戏体验的风险。在其他情况中你正在寻找一种可以很好地玩一定范畴的游戏的算法，并且没有时间去为每个游戏重新训练智能体。

3.2.2.4 你正在进行多少种游戏？

AI 设计者可能希望达到的一个目标是**全局游戏策略**。此时，我们不是在寻找一个用于某个单独游戏的策略，我们是在寻找一个更为通用的智能体，其可以玩任意的它所代表的游戏——或者至少任何来自一个特定的分布或种类中的游戏，并且其附加到一个给定的接口。全局游戏策略是通常是出于一种使用游戏来推进人工通用智能发展的理念，也就是开发不仅仅擅长一件事情，而是许多不同的事情的 AI[597, 679, 745]。这个想法是为了避免过拟合(overfitting)一个给定的游戏，并且提出了能够很好地适应许多不同游戏的智能体；在为一个特定游戏开发智能体是这是一种普遍的现象，例如，对于一个基于游戏的 AI 竞赛，许多专门的解决方法被设计为不能很好地转换到其他游戏[701]。

出于这个原因，普遍是在未见过的游戏上评估全局游戏策略，也就是，那些它们从未在其之上被训练过以及智能体的设计者在开发智能体时并不知晓的游戏。也有一些用于全局游戏策略的框架，包括了全局游戏策略竞赛(General Game Playing Competition)[221]，通用视频游戏 AI 竞赛[527, 526]以及 Arcade Learning Environment[39]。这些将稍后在本章进行谈论。

通用的视频游戏策略，其中 AI 是为其以玩家身份贯穿许多游戏（理想的是在所有游戏中）的表现而进行游戏而开发的，可以看出是与在商业游戏开发中用于进行游戏的 AI 的普遍应用截然不同的，其中 AI 是在一个非玩家角色下为了体验而执行的，并且被仔细地调整到了某个特定的游戏。然而，全局游戏策略的 AI 方法的发展最终肯定也会有益于商业游戏 AI。并且即便是在作为游戏开发的一部分而开发游戏 AI 时，开发在某种程度上可以被重用的方法也是很好的工程实践。

3.3 AI 如何可以进行游戏？

在第 2 章，我们回顾了许多重要的 AI 方法。那些方法的大部分都能以一种或多种方式被用于玩游戏。这个章节将会集中于核心的 AI 方法，并且针对每种算法族来介绍它们如何可以被用于玩游戏。

3.3.1 基于规划的方法

通过在一个状态空间中规划一系列未来的动作来选择动作的算法对游戏来说通常是可行的，并且通常不需要任何训练时间。它们的确需要一个快速的前向模型，如果是在游戏的状态空间中搜索，而不是简单地使用它们它们来在物理空间（路径规划）中搜索。树搜索算法被广泛地用于玩游戏，无论是在它们自身还是在进行游戏的智能体结构中扮演支撑角色。

3.3.1.1 传统的树搜索

传统的树搜索方法，其拥有少数或没有随机性，从 AI 以及游戏的研究上的早期阶段开始就已经在玩游戏的角色上得到使用。正如本书的导论介绍的那样，极大极小算法以及 α-β 剪枝一开始被发明是为了玩经典的棋盘游戏，例如国际象棋与国际跳棋[725]。尽管对抗性树搜索的基本概念自那之后并没有真正地改变，但是现有的算法以及新的算法已经有了大量的调整。通常，传统的树搜索方法可以被简单地应用到有着完全可观察性，一个低分支因子以及一个快速前向模型的游戏中。理论上来说它们可以解决任何对于玩家来说有着完全可观察性的确定性的游戏(见图 3.2 的红色方块)；在实践中，它们仍然败于包含了巨大状态空间的游戏。

最佳优先搜索，特别是 A*算法的无数变种，在现代的视频游戏中被非常普遍地用于**路径规划**。当一个现代 3D FPS 或 RPG 中的某个 NPC 决定如何从点 A 到达点 B 时，这通常会使用 A*的某种版本完成。在这一的情况中，搜索通常完成于（游戏中的）物理空间而不是搜索空间，所以前向模型并不是必须的。由于空间是伪连续的，搜索通常完成于覆盖于要被穿越的区域上的一个网格或是点阵的节点。请注意最佳优先欧诺并不仅仅用于导航，并且不被用于智能体的完整决策制定；如行为树或者有限状态机这样的方法（其通常是被手动编写的）被用于决定前往何处，而 A*（或者它的某些变种）被用于决定如何到达那里。事实上，在玩家输入是通过指向并且点击位置来前往的游戏中——想想一个头顶的指针，像《*暗黑破坏神*》Diablo (Blizzard Entertainment, 1996)，或者一个 RTS，像《*星际争霸*》(Blizzard Entertainment, 1998)——玩家命令的执行通常也涉及到一个路径规划算法。最近加入到最佳优先算法族中的包括了跳点式搜索(JPS)，其在适当的环境下比起 A*算法能够以数量级的水平来改善性能[662]。分层式寻路(hierarchical pathfinding)是它自身的一个小型研究方向，基于将一个区域分割为数个子区域并且使用单独的算法来决定如何在区域间以及区域内如何到达的想法。为一个现代视频游戏选择一个路径规划算法通常是一个要选择在给定 NPC 穿越的环境的形状，一个覆盖于这个空间顶部的网格或移动图形，以及动画算法的需求时表现最好的算法的问题。通常来说，一种尺寸并不是适合所有；某些致力于面向工业界的游戏 AI 的教科书更佳深入地探讨了这个问题[460]。

除了路径规划，如 A*这样的最佳优先算法可以被用于**控制** NPC 行为的所有方面。要达成这的关键在于在游戏状态空间中的搜索，而不只是物理空间。(很明显，这需要一个快速的前向模型) 举个例子来说，2009 年的马里奥 AI 竞赛的冠军是完全基于在状态空间中的 A*搜索的[705]。这个竞赛要求选手开发可以玩一个经典平台游戏《*超级马里奥兄弟*》(Nintendo, 1985)基于 Java 的克隆版的智能体——它之后变成了一个多元化的竞赛[320]。尽管原始的竞赛软件并没有提供一个前向模型，竞赛的冠军，Robin Baumgarten，通过调整部分核心游戏

代码来创造了一个。他之后建立了一个 A*智能体，其在任何时间点都只是简单地尝试到达屏幕的右边缘。（这个智能体的一个示例图可以通过图 3.3 与 2.5 看到）这表现的非常优秀：产生的代理似乎是以最佳方式来运作的，并且尝试设法到达了竞赛软件包含的所有关卡的尽头。一个展示了智能体对关卡之一进行导航的的视频在 Youtube 上收到了超过一百万次浏览；观看智能体玩游戏的吸引力部分是由于智能体在多个敌人之间进行导航的极端技巧。

值得注意的是，这个智能体的成功是出于几个因素。一个是关卡几乎是线性的；在这个竞赛之后的一个版本中，需要回溯的有着死亡结尾的关卡被引入，其挫败了单纯的 A*智能体[320]。另外两个因素是《*超级马里奥兄弟*》(Nintendo, 1985)是确定性的，并且拥有局部的完美信息（当前屏幕的信息在任何时刻都是完全知晓的），因此一个好的前向模型自然是可行的：如果 A*将不能使用游戏的一个完整模型，包括敌人的移动，它是不可能规划这些敌人的路径的。

3.3.1.2 随机的树搜索

MCTS 算法在 2006 年进入了围棋研究的舞台[140, 76]，预示了围棋游戏 AI 性能的飞跃。传统的对抗搜索在围棋上表现不佳，部分是因为分支因子实在过高（比国际象棋高一个数量级），部分是因为围棋的性质使得它非常难以从算法上来判定一个棋牌状态的价值。MCTS 通过构建一棵不平衡的树部分地战胜了这些挑战，其中并非所有的招式都需要被探索到相同的深度（降低有效的分支因子），并且进行随机推演直到游戏的结束（降低了对一个局面评估函数的需求）。在 2016 年和 2017 年战胜了世界上最好的围棋选手中的两位的 AlphaGo[628]程序，就是围绕着 MCTS 算法建立的。

MCTS 在围棋上的成功让研究人员与从业者探索它在玩其他多种种类的游戏上的用途，包括了交易卡牌游戏[747]，平台游戏[292]，即时战略游戏[309, 645]，竞速游戏[201]等等。当然，这些游戏与围棋有着许多不同之处。虽然围棋是一个确定性的完美信息博弈，一个像《星际争霸》(Blizzard Entertainment, 1998)这样的即时战略游戏，一个像万智牌这样的交易卡牌游戏，或者任何扑克变种都同时拥有隐藏信息与随机性。例如信息集蒙特卡罗树搜索这样的方法是一种解决这些问题的方式，但是增加了它们自身的计算开销[145]。

另一个问题是，在有着精细的时间粒度的游戏中，一次随机推演可能会需要一个望而却步的长时间来达到一个末端局面（失败或胜利）；在许多视频游戏中有可能可以采取任意数量的动作而没有游戏的胜利或者失败，甚至在实质上影响游戏的结果。例如，在《*超级马里奥兄弟*》(Nintendo, 1985)中，大部分随机产生的动作序列并不会看到马里奥离开最初的屏幕，而基本是是来回摆动，直到在成千上万的时步之后时间耗尽。这个问题的一种应对是使用一个状态评估函数[76]。其他的思路包括了为动作选择剪枝，以时算法搜索的更深[292]。考虑考对 MCTS 算法的所有组件的大量改动，将 MCTS 视为一种通用的算法框架而不是一个单独的算法可能更有意义。

许多游戏可以通过 MCTS，非启发式搜索（例如宽度优先搜索）或者启发式搜索（例如 A*）来玩。决定使用哪一种方法并非总是直截了当的，但幸运的是这些方法的实现与测试相对简单。通常来说，极大极小算法只能被用于（双人）对抗性游戏，与此同时非启发式搜索的其他形式最适用于单人游戏。最佳优先搜索需要某种形式的到目标状态的距离的估计，但这不需要是一个物理位置或者游戏的最终目标。MCTS 的变种既可以被用于单人游戏也可以被用于双人游戏，并且在分支因子很高时候经常有着超越无启发式搜索的表现。

3.3.1.3 进化规划

很有趣的是，通过规划进行决策并不需要基于树搜索。相反，可以为规划使用优化算法。其基本思路是，你可以优化整个动作序列，而不是从一个初始点开始搜索某个动作序列。换句话说，你是在完整动作序列的空间中搜索那些有着最大效用的动作序列。评估一个给定的动作序列的效用是通过在模拟中简单地执行所有序列中的动作，或者关卡在采取所有这些动作之后得到的状态的价值而实现的。

这个想法的吸引力在于，优化算法可能会以与树搜索算法非常不同的方式来搜索规划空间：所有的树搜索算法始于树的根节点（初始状态）并且从这个点开始构建一棵树。而进化算法将规划简单地视为一个序列，并且可以在字符串中的任何点执行变异或交叉。这可以帮助在树搜索算法会针对相同问题而探索的规划空间的不同区域中为搜索进行引导。

尽管许多不同的优化算法可以被应用，但为数不多的可以在文献中被找到的关于在游戏中的基于优化的规划上的研究使用了进化算法。Perez 等人提出为单人动作游戏使用进化规划，称这个方法为“旋转水平进化(rolling horizon evolution)”[525]。在物理旅行商问题(Physical Traveling Salesman Problem)（一种经典旅行商问题与一种竞赛游戏之间的混合）的特定实现中，一种进化算法在每一步都被用于生成一个规划。这个规划被表示为一个包含 10-20 个动作的序列，并且一个标准的进化算法被用于搜索规划。在某个规划被找到之后，这个规划的第一步就是被执行，就像在树搜索中会出现的那样。基于进化规划的智能体通常在通用视频游戏 AI 竞赛中表现的很有竞争力。

进化规划作为一种用于解决非常大的分支因子的技术十分有潜力，就像我们已经看到的拥有多个独立单位的游戏那样。Justesen 等人[307]将进化计算应用到在回合制策略游戏《*英雄学院*》Hero Academy (Robot Entertainment, 2012)中用于选择动作，称这个方法为“在线进化(online evolution)"。鉴于玩家控制的单位数目以及每个单位可用的动作的数量，分支因子大约为一百万；因此，只有之后的单个回合会被规划。在这个游戏中，进化规划被证明为以比较大的差异超越了蒙特卡罗树搜索的表现。Wang 等人[746]，以及 Justesen 和 Risi[308]之后应用了这个技术的变种到《*星际争霸*》(Blizzard Entertainment, 1998)的战术中。鉴于游戏的连续空间性质，如果所有单位所有可能的移动方向都被考虑为一个单独的动作，分支因子将变得非常极端。因此所演化的并非一个动作序列，而是在一个给定的时刻每个单位将会使用哪一个简单的的脚本（战术）（这个思路借用于 Churchill 与 Buro，他们将一个脚本的“文件夹”与简单的树搜索相结合[122]）。Wang 等人[746]表明，在这个简单的《星际争霸》场景中进化规划比树搜索算法的几种变化表现的更好。

游戏中的进化规划是一个近期的发明，并且目前为止只有优先数目的关于这项技术的研究。暂时还没有很好地了解在什么条件下表现的很好，甚至是在此时为什么它表现的非常好。一个主要的未解决问题是如何来表现进化对抗规划(evolutionary adversarial planning)[585]（例如参照极大极小算法），暂时不清楚如何将这整合为一种属型(genotype)。也许是通过不同参与者所采取的动作的竞争性共同进化？换句话说，在这个领域中仍有大量的研究余地。

3.3.1.4 带有符号化表示的规划

尽管在游戏内动作这一水平上的规划需要一个快速的前向模型，但也有其他的方式来在游戏中使用规划。特别是，可以以一种游戏的状态空间的抽象表示进行规划。自动规划的领域已经研究几十年在符号化表示这一水平上的规划[226]。特别是，一种基于用来表示事件，状态和动作，以及树搜索方法的一阶逻辑的语言被应用来寻找从当前状态到某个结束状态的路径。这种风格的规划起源于被世界上第一个数字化移动机器人 Shakey 所使用的 STRIPS 表示[493]；符号化的规划已经被广泛地应用于多个领域。

恐怖主题的第一人称射击游戏《*极度恐慌*》(Sierra Entertainment, 2005)由于它使用规划

来调整 NPC 行为而在 AI 社区中知名。这个游戏的 AI 也在游戏媒体上得到了很好的评价，因为玩家能够听到 NPC 互相沟通攻击计划，提升了沉浸感。在《*极度恐慌*》(Sierra Entertainment, 2005)中，一个类似 STRIPS 的表示被用于规划为了打败玩家角色哪个 NPC 要表现哪个动作（侧翼，掩护，压制，设计等）。这个表示是建立在单独房间的水平之上的，其中一个方面与下一个方面之间的移动通常是一个单独的动作[506]。使用这种高层级的表示，可以比在使用单独的游戏动作的规模来进行规划时规划的更为向前。然而，这样的表示需要手动地定义状态与动作。

3.3.2 强化学习

正如第 2 章所讨论的，一个强化学习算法就是任何解决了某个强化学习问题的算法。这包括了来自时序差分或者相近的动态规划家族的算法（为简单起见，我们将这样的算法称为经典强化学习方法），进化算法到强化学习的应用，如神经进化和遗传编程(genetic programming)，以及其他方法。在这个章节，我们将会同时探讨被应用在进行游戏的经典方法（包括哪些设计到深度神经网络的）与进化方法。另一种描述这些方法之间的差异的方式是个体发育(ontogenetic)（其在”生命期”内学习）与系统发育(phylogenetic)（其在”生命期”之间学习）方法之间的差异。

强化学习算法适用于存在学习时间的游戏 :大部分强化学习方法为了精通游戏将需要反复第执行游戏成千上万，甚至上百万次。因此，拥有一种比真实时间快很多的执行游戏的途径（或者一个非常大的服务器群）是非常有用的。某些，但不是全部的强化学习算法也需要一个前向模型。一旦训练完毕，一个强化学习到的策略通常可以被非常快地执行。

很值得注意的是，用于进行游戏的基于规划的方法（描述于先前的章节中）不能被直接地与描述于本章的强化学习方法进行比较。他们解决了不同的问题 : 规划在每个时间步骤都需要一个前向模型和明显的时间；而强化学习需要学习时间，可能需要也可能不需要一个前向模型。

3.3.2.1 传统以及深度强化学习

正如本书的导论中已经介绍的那样，传统的强化学习方法很早就被应用于游戏上，在某些情况下有着很可观的成功。Arthur Samuel 在 1959 年设计了一种算法——其可以说是第一个传统的强化学习算法——以创造一个自我学习的国际跳棋玩家。尽管当时计算资源极度有限，但是这个算法学习的足够优秀，可以打败它的制作人[590]。经典强化学习算法在游戏上的另一个成功出现在数十年之后，当时 Gerald Tesauro 使用时序差分学习的现代形式来教一个简单的神经网络去玩西洋双陆棋，称之为 TD-gammon；在不带有信息的开始以及简单地与自身对弈之后，它学习到了令人吃惊的优秀程度[689]（TD-gammon 被更详细地涵盖在了第 2 章）。在 90 年代与 21 世界初，这个成功极大地激发了对强化学习的兴趣。,

然而，这个进展被由于缺乏优秀的值函数（例如，Q 函数）的逼近器而被限制。尽管例如 Q-learning 这样的算法在正确的条件下很可能会收敛到最佳策略，但各种正确的条件实际上是非常有限制的。特别举个例子来说，它们包括了在一个表中分别存储所有状态或{ 状态，动作 }的值。然而，对于大多数有意思的游戏来说，在这一点的可行性上来说是有着远远多的可能状态——几乎任何视频游戏都至少有着数以十亿的状态。这意味着表格将会变得过于巨大而无法被内存容纳，并且大部分状态在学习时将永远不会被访问到。很显然，有必要使用一个压缩过的值函数的表示，其占用了更少的内存并且为了计算它的值也不需要每个状态都被访问过。而且可以基于临近的已经被访问过的状态来计算它。换句话说，所需要的就是

一个**函数逼近器**，例如一个神经网络。

然而，同时使用神经网络与时序差分学习的结果是并不平凡的。它非常容易遇到“灾难性的遗忘”，其中复杂的策略未被学习过并且偏向退化的策略（例如总是采取相同的动作）。其中原因是复杂的并且超出了本章的讨论范畴。然而，为了直观地理解其中的一个机制，考虑一下对一个玩某个游戏的强化学习智能体来说什么是会经常发生的。奖赏是非常系数的，并且智能体经常会长时间的没有奖赏，或者负的奖赏。当长时间碰到同样的奖赏时，反向传播算法将只被这个奖赏的目标值所训练。就监督学习来说，这等同于在一个单独的训练样例中进行了一次长期训练。可能的结果就是，网络只学习到了这个目标值的输出，无论输入如何。关于使用 ANN 对一个值函数进行逼近的方法的更多细节可以在章节 2.8.2.3 中找到。

时序差分的变化与函数逼近器相结合的强化学习在使用上的一个重要成功出现于 2015 年，当时谷歌 DeepMind 公布了一篇论文，在其中它们尝试训练深度神经网络来玩多种来自雅利达 2600 游戏主机的不同游戏[463]。每个网络都被训练来玩某个单独的游戏，而输入是游戏在视觉上的原始像素，带着得分，而输出是控制器的方向与开火按钮。被用于训练深度网络的方法是深度 Q 网络，其在本质上是应用到了具有多个层级的神经网络上（在这个结构中使用的某些层是卷积的）的标准 Q-learning。关键的是，他们设法通过一种称为经验回放(experience replay)的技术来克服了与时序差分学习和神经网络的结合相关的问题。在这里，游戏玩法的短暂序列会被保存，并且以不同的顺序重放到网络，以打破相似状态与奖赏的长链条。这可以被视为类似于监督学习中的基于批次的训练方法，其同样使用小的批次。

3.3.2.2 进化强化学习

另一个主要的强化学习方法家族是进化方法。特别是，使用进化算法来演化神经网络（神经进化）或者程序的，特别是结构为表达树（遗传编程）的程序的权重以及/或者拓扑结构。适应性评估函数来自于使用神经网络或者程序去进行游戏，并且使用结果（例如，分数）作为一个适应度函数。

这个基本思路已经出现了很长时间了，但是在很长一段时间内却令人吃惊地被人低估了。John Koza，一位遗传算法的知名研究者，在他 1992 年出版的书中使用了一个玩吃豆人的进化程序案例[355]。在几年之后，Pollack 以及 Blair 表明进化计算可以用于训练西洋双陆棋玩家，就像 Tesauro 在他的 TD 学习实验中那样使用相同的设置，并且具有类似的结果[536]。在游戏之外，一个在 90 年代正在成形的研究者社区探索了使用进化计算来为小型机器人学习控制策略的结果；这个领域逐渐被称为进化机器人学(evolutionary robotics)[495]。训练机器人来解决例如导航，避障以及条件学习这样简单的问题与训练 NPC 来玩游戏也非常相似，尤其是二维的类街机游戏[566, 768, 767]。

从 2005 年左右开始，在应用神经进化来玩不同种类的视频上，取得了许多提升。这包括了在车辆竞速上的应用[707, 709, 391, 352]，第一人称设计[517]，策略游戏[78]，即时战略游戏[654]以及经典的街机游戏，例如吃豆人[767, 402]。可能从这项工作中得到的主要一点是，神经进化是十分地全才，并且可以被应用到广泛类型的游戏上，通常对每种游戏来说都只有少数不同的途径。例如，对于一个简单的车辆竞速游戏来说，这显示了用做状态评估的进化神经网络，即便结合了一个简单的单步向前搜索，通常也能超越做动作评估的进化神经网络的表现[407]。输入特征也是一个问题；就像在章节 3.2.2.1 中讨论的那样，以自我为中心的输入通常来说值得强烈倾向的，与此同时对单独的游戏类型也有一些额外的考量，例如如何表示多个对手[654]。

神经进化已经在那些状态可以被相对少的维度（比如，在神经网络的输入层少于 50 个单位）所表示的情形中进行学习策略上取得了巨大的成功，并且比时序差分变化的传统强化

学习算法更易于调整与见效。然而，神经进化似乎在扩大到有着需要巨大并且深层的神经网络的极大输入空间的问题时存在一些问题，例如那些使用高维度像素输入的问题。关于这个问题的可能的原因是，在权重空间中随机搜索遇到了一种梯度下降搜索（例如反向传播）所没有的维度灾难(curse of dimensionality)。目前，几乎所有从高维度像素输入中直接学习的成功案例都使用 Q-learning 或者类似的方法，尽管也存在将神经进化与无监督学习结合的方法，来让控制者去学习使用一个视觉反馈的压缩表示来作为输入[352]。

关于更多的神经进化的通用方法与文献指引，读者可以参阅章节 2.8.1，以及近期关于游戏中的神经进化的综述文献[566]。

3.3.3 监督学习

游戏也可以通过使用**监督学习**来玩。或者说，用于进行游戏的策略或者控制器可以通过监督学习来学习。这里的基本思路是记录人类玩家玩某个游戏的轨迹并且训练某些函数逼近器做出类似人类玩家的行为。这个轨迹被记录在元组<特征，目标>的列表中，其中特征代表了游戏状态（或者智能体可用的一个对它的观察）而目标是人类在这个状态下做出的动作。一旦函数逼近器被充分地训练，游戏就可以进行了——以它所训练的人类的风格——通过简单地在呈现当前游戏状态时采用训练过的函数逼近器返回的任意动作。或者，即便不去学习预测什么动作会被采用，也可以去学习预测状态的值，并且使用训练过的函数逼近器与一个搜索算法相结合来玩这个游戏。关于可能的可以在游戏中被使用的监督算法的更多细节被描述于第 2 章中。

3.3.4 嵌合式游戏玩家

尽管规划，强化学习与监督学是从基础上就不同的玩游戏的方法，在不同的限制下解决了进行游戏的问题，其并不意味着它们不能被结合。事实上，存在许多来自这三个种类的方法的成功**混合**或是**嵌合**的案例。一个案例是**动态脚本**(dynamic scripting)[650]，其可以被视为一个**学习分类系统**(learning classifier system)[362]的某种形式，在其中它涉及到了基于规则（这里称为基于脚本）的表示加上强化学习。动态脚本通过强化学习在运行期调整了脚本的重要性，并且是基于当前游戏状态以及获得的瞬时奖赏的。动态脚本已经在游戏中有不少应用，包括了格斗游戏[416]以及即时战略游戏[408, 153]。这个方法主要用于要适应玩家技能的 AI，因此目标在于玩家的体验，而不一定要赢得游戏。

另一个很好的案例是 AlphaGo。这个性能卓越的围棋游戏智能体实际上是通过搜索，强化学习与监督学习来结合规划[628]。在这个智能体的核心是一个蒙特卡罗树搜索算法，其在状态空间中搜索（规划）。然而，随机推演结合了来自一个岁状态的值进行估计的神经网络的评估，而节点选择被一个位置评估网络所启发。状态网络与位置网络最初都是在围棋大师之间的对弈的数据库上训练的（监督学习），后来又通过自我对弈进行进一步训练（强化学习）。

3.4 AI 可以玩哪一个游戏？

对于让 AI 玩游戏来说，不同的游戏有着不同的挑战，于此同时他们也为人类玩游戏带来了不同的挑战。不仅是 AI 玩家拥有的对游戏的联接方式上的不同，也是游戏类型之间的不同：一个用于玩国际象棋的策略几乎不可能在《*侠盗猎车手*》(Rockstar Games, 1997–2013)系列中有效的进行游戏。这个章节是根据游戏种类所组织的，并且针对每个游戏种类，它探

讨了这个特定种类的游戏通常拥有的特定的在认知，感知，行为以及动觉上的挑战，并在之后给出了一个 AI 方法已经如何被应用于玩这个特定游戏种类的概述。它也包括了一些拓展案例，给出了一些关于特定实现的系列。再一次强调的是，这个列表并不包含了所有可能的 AI 可以扮演一个玩家或者非玩家角色的游戏种类；这个抉择是基于游戏种类的流行度基础，以及在每个种类中已有的用于玩游戏的 AI 的公开工作之上而做出的。

3.4.1 棋盘游戏

就像本书的导论所介绍的那样，最早的在游戏 AI 上的工作是完成于传统棋盘游戏中的，并且很长一段时间内其是 AI 被应用于玩游戏的唯一方式。特别是，国际象棋被非常普遍的被用作 AI 研究，被称为“人工智能的果蝇”[192]，暗指在基因研究中将普通的果蝇用作一种模范生物。这一点的原因似乎是由于棋盘游戏易于实现，实际上，完全可能是由于其是在 AI 研究的早期时可用的有限计算机硬件上实现的，并且那些游戏被认为需要某些类似“纯粹思维”的东西。而像国际象棋或是围棋真正需要的其实是对抗性规划(adversarial planning)。传统的棋盘游戏通常对感知，反应，运动技能或是对连续移动的评估能力没有任何要求，这意味着它们的技能要求十分狭隘，特别是与大多数视频游戏相比较。

大部分棋盘游戏有着非常简单的离散状态表示以及确定性的前向模型——游戏的完整状态经常可以被表示为小于 100 个字节，并且计算游戏的下一个状态就像应用一组小规则一样简单——以及合理小的分支因子。这使得它非常容易应用树搜索，并且几乎所有成功的棋盘游戏智能体都使用树搜索算法的某种形式。就像第 2 章中对树搜索的讨论那样，极大极小算法最初是在国际象棋的背景下发明的。几十年的集中于在国际象棋上的研究（而在国际跳棋与围棋上的研究较少），以及致力于这类研究的特定会议，导致了许多算法的进步，改善了极大极小算法在一些特定棋盘游戏上的性能。这里面的许多在它们被发展的特定游戏之外都只有有限的应用性，而在这里深入了解这些算法变种的话将会花费我们太多的时间。对于国际象棋的研究的一个概述，读者可以参阅[97]。

在国际跳棋中，卫冕的人类冠军在 1994 年被 Chinook 程序所击败[593]，并且游戏在 2007 年被解决，意味着双方玩家的最佳招式集合已经被找到（如果你使用最佳策略，那么它会是一个和局）[592]；在国际象棋上，Garry Kasparov 在 1997 年被深蓝十分出名所击败[97]。直到 2016 年，谷歌 DeepMind 击败了使用它们的 AlphaGo 程序击败了一位人类围棋冠军[682]，主要是由于必要的算法上的提升。尽管国际象棋和国际跳棋可以使用极大极小算法结合相对浅的状态评估来有效的进行，围棋的巨大分支因子仍然刺激并且使得 MCTS 的发展[76]成为了必要。

尽管 MCTS 可以被用于玩棋盘游戏，并且不使用某种状态评估函数，为这个算法补充状态以及动作评估函数仍然可以大量地提高性能，就像在 AlphaGo 的例子中看到的那样，其使用了深度神经网络作为状态与动作评估。换句话说，在使用了一些极大极小的变种时，必须要使用状态评估函数，因为所有有意义的棋盘游戏（比井字棋更复杂的）要在可接受的时间内一直搜索到游戏的结尾的话，有着过于巨大的状态空间。这些评估函数可以被手动构造，但使用学习算法的某种形式来学习它们的参数通常都会是一个好主意（尽管算法结构是由算法设计者所特化的）。就像上面所讨论的那样，Smuel 是第一位使用强化学习的某种形式来在一个棋盘游戏（或任何类型的游戏）中学习一个状态评估函数的[590]，并且 Tesauro 之后使用了 TD 学习在西洋双陆棋中取得了非常好的效果[689]。进化计算也可以被用于学习评估函数，例如，Pollack 展示了共同进化在使用与 Tesauro 非常类似的设置下可以表现的非常良好[536]。值得注意的基于进化评估函数的强大棋盘游戏玩家的案例是 Blondie24 [205] 以及 Blondie25 [206]，分别是一个国际跳棋程序以及一个国际象棋程序。评估函数基于五层

的深度卷积网络，并且 Blondie25 在对抗非常强大的国际象棋玩家时表现的非常出色。

尽管传统的棋盘游戏，例如国际象棋与围棋，已经存在了数百年甚至数千年了，但过去十几年中，已经见到了一个棋盘游戏设计的复兴。许多更为近期设计的棋盘游戏将传统棋盘游戏的公式与来自其他游戏种类的设计思考相结合。一个很好的案例是《*车票之旅*》(Days of Wonder, 2004)，其是一个包括了卡牌游戏元素的棋盘游戏，例如可变的玩家数目，隐藏信息以及随机性（在卡牌的抽牌阶段）。由于这些原因，难以基于标准的树搜索方法来构建性能优异的 AI 玩家；最知名的包括了实质性的领域知识的智能体与人类玩家相比也表现不佳[159]。为这种类型的游戏创造表现良好的智能体也是一个有趣的研究挑战。

考虑到使用树搜索来进行棋盘游戏的简便程度，就不会奇怪我们目前为止已经讨论过的每一个方法都是基于一个或另一个树搜索算法之上。然而，也有可能不使用前向模型来玩棋盘游戏——通常有着"有趣"而不是良好的结果。例如，Stanley 和 Miikkulainen 开发了一个用于玩围棋的"游走的目光"方法，在其中一个进化神经网络自我扫描围棋棋盘并且决定在哪里摆放下一个棋子[656]。与此相关的是，根据报道 AlphaGo 的局面评估网络也可以凭借自身来进行一场高质量的围棋博弈，但很自然的是它结合搜索将会表现的更为强大。

3.4.2 卡牌游戏

卡牌游戏是以一副或几副卡牌为中心的游戏；这可能是也可能不是标准的 52 张的法国套牌，其被普遍用于传统卡牌游戏。大部分卡牌游戏牵涉到了玩家持有不同的卡牌，其可以在玩家间，或玩家与某副牌间，或是桌上的其他位置间转移所有权。另大部分卡牌游戏的另一个重要元素是某些卡牌对持有它们的玩家是可见的，但对其它玩家是不可见的。因此，几乎所有的卡牌游戏都具有很大程度的隐藏信息。事实上，卡牌游戏可能是隐藏信息最为主导游戏玩法的游戏类型了。

用传统的卡牌游戏扑克举例来说，现在它的德州扑克(Texas hold 'em)变体非常受欢迎。其规则也相对简单：在几轮结束之后持有最佳牌面（也就是"最佳手牌"）的玩家获胜。在回合之间，玩家可以交换多张牌以从桌上抽取新牌。如果存在完美信息，也就是所有玩家可以看见其他人的手牌，这会变成一个索然无味的游戏，可以根据查询表来进行游戏。让德州扑克——以及类似的扑克变体——变得充满挑战性并且有趣的是每个玩家并不知道其他玩家拥有什么牌。玩这类游戏的认知挑战设计到了在信息匮乏的情况下做决定，这意味着从不完美的证据中推测游戏状态，并且可能影响玩家其他玩家对真实游戏状态的感知。换句话说，一场扑克游戏在很大程度上关系到了猜测与虚张声势。

在玩扑克以及类似的游戏时一个重要的进步是虚拟遗憾最小化(Counterfactual Regret Minimization, CFR)算法[798]。在 CFR 中，算法通过一个与差分学习以及其他强化学习算法是如何被应用在西洋双陆棋和国际跳棋这样的完美信息博弈上相类似的方式来通过自我对弈而学习。基本原理是在每个动作之后，当某些隐藏信息被揭露，根据新揭露的信息，它转而计算所有其他之前可以采取的动作的奖赏。在被实际采用的动作与之前可以采用的最佳动作的奖赏之间的差异被称为*遗憾*(regret)。策略在之后会被调整以最小化遗憾。这是迭代地完成的，缓慢地收敛到某个最佳的策略，也就是大量的游戏之后，它会失去越来越少的遗憾。然而，对于像德州扑克这样的复杂的博弈，为了在实践中使用 CFR 算法，不得不做出一些简化。

DeepStack 是一个近期发布的智能体，其在德州扑克上达到了世界级别的表现[466]。类似 CFR，DeepStack 使用了自我对弈以及递归推理来学习策略。但是，它在开始游戏前并没有计算一个明确的策略。相反，它使用树搜索结合一个状态值估计来在每轮中选择动作。从这个意义上来说，它更像是 AlphaGo 的启发式搜索（但是是在大量不完美信息的情况下）而

不是像 TD-gammon 那样增强学习而来的策略。

另一个更为近期并且从搜索社区吸引了越来越多兴趣的卡牌游戏是《*炉石传说*》(Blizzard Entertainment, 2014)；见图 3.4。这是一种以万智牌为传统的收集类卡牌游戏，但是带有某种程度上更为简单的规则并且只能在计算机上进行游戏。一场炉石传说的游戏发生在两个玩家之间，并且每个玩家的牌组中有 30 张卡牌。每张卡牌代表了一个随从或者一个法术。每个玩家手中都只有几张牌（<7，对其他玩家不可见），并且每个回合抽取一张新的卡牌并且可以选择使用一张或者更多的卡牌。随从卡牌转换为被放置在玩家一方桌上的随从（对双方玩家可见），并且随从可以被用于攻击敌人的随从或者玩家角色。法术拥有许多不同的效果。游戏中存在的数以百计的卡牌，每个回合选择多种动作的可能性，玩一场游戏所需要消耗的漫长时间（20 到 30 回合是普遍的），随机性以及想当然的隐藏信息的存在（主要是在对手的手上有什么牌），合在一起让《*炉石传说*》(Blizzard Entertainment, 2014)变成了一个对人类和机器来说都很难玩的游戏。

可能最简单的用于玩《*炉石传说*》(Blizzard Entertainment, 2014)的方法是简单地忽略隐藏信息并且以某种方式来执行每个回合，也就是说，搜索在一个单独回合内的可行动作的空间，并且只根据已有的信息来选择最大化某些准则的那一个动作，例如在回合结束时的生命值优势。实现了这样的贪婪策略的智能体被包括在了某些开源的炉石传说模拟器中，例如 *Metastone*。标准的树搜索算法，例如 Minimax 或者 MCTS，在这里由于高程度的隐藏信息，一般是无效的（就像在扑克中那样）。一种用于构建高性能智能体的方法是转而手动编写领域知识，例如，通过建立卡牌的本体并且在一个抽象的符号化空间中搜索[659]。

不同于玩家不能控制将会被发到什么牌的扑克，在《*炉石传说*》(Blizzard Entertainment, 2014)中，玩家也可以构建一副牌并且使用其来进行游戏。这为玩这个游戏增加了另一种等级的挑战：除了在每回合要选择采用什么动作之外，成功的玩家还必须构建让她的策略得以实现的牌组。牌组的组成有效地限制了可以被选择的策略，以及之后通过动作选择在战术上的实现。虽然这两个层次相互影响——一个强大的玩家在选择一个招式时会将卡牌的组成以及它所提供的策略一同纳入考虑，而反之亦然——牌组构建与动作选择的问题的确可以在某种程度上被分别对待，并且实现于不同的智能体中。一种构建牌组的方式是使用进化计算。牌组被视为基因组，并且适应度函数包含了使用这个牌组来进行游戏的简单启发式智能体[216]。一种类似的方法也已经被用于多人卡牌游戏*皇舆争霸*[415]。

3.4.3 传统街机游戏

传统街机游戏，也就是那类出现于 20 世纪 70 年代初与 80 年代早期的街机游戏机，家用视频游戏主机以及家用电脑，已经在过去的几十年中被普遍被用作 AI 基准。这类游戏具有代表性的平台是雅利达 2600，任天堂 NES，Commodore 64 以及 ZX Spectrum。大部分传统街机游戏的特点是在一个二维空间中移动（某些时候被等体积地表示来提供三维移动的错觉），图形逻辑的重度使用（其中游戏规则由精灵(sprites)或者图像的交集而触发），连续时间的进度，以及连续空间或离散空间的移动。

玩这类游戏在认知上的挑战因游戏而异。大部分游戏需要快速的反应与精确的计时，并且少数游戏，特别是如《*田径*》(Konami, 1983)以及《*十项全能*》Decathlon (Activision, 1983)这样早期的运动游戏几乎完全依靠速度与反应（图 3.5(a)）。非常多的游戏需要为几个同时发生的事件分出先后，这需要某种能力来在游戏中预测其他实体的行为轨迹。例如在《*轻敲者*》(Bally Midway, 1983)中，这种挑战就是很明显的——见图 3.5(b)——但是在部分平台游戏中也会有不同的方式，例如《*超级马里奥兄弟*》(Nintendo, 1985)，以及《*打鸭子*》(Nintendo, 1984) 或《*导弹指挥官*》(Atari Inc., 1980)这样的射击场游戏，还有如《*防卫者*》(Williams

Electronics–Taito, 1981)或《*异型战机*》(Irem, 1987)这样的滚动式射击游戏。另一个普遍的需求是导航迷宫或者其他复杂的环境，不仅被像《*吃豆人*》(Namco, 1980)，《*吃豆人小姐*》(Namco, 1982)，《*青蛙过河*》(Sega, 1981)以及《*推石小子*》(First Star Software, 1984)等游戏最为突出地进行了例证，并且在许多平台游戏中也是很普遍的。一些游戏，例如《*蒙特祖玛的复仇*》(Parker Brothers, 1984)，需要长期的规划，并涉及到了对暂时无法观察的游戏状态的记忆。一些游戏具有不完整的信息与随机性，而其他游戏是完全确定性并且可以充分观察的。

3.4.3.1 吃豆人以及吃豆人小姐

经典的《*吃豆人*》(Namco, 1980)游戏的多种版本以及克隆已经被频繁地用于人工智能的研究与教学，这是由于它是一个有深度的挑战，并且伴随着概念上的简洁以及实现上的容易。在这个游戏的所有版本中，玩家角色在躲避追逐的鬼魂的同时要移动贯穿一个迷宫。当所有分布在关卡各处的豆子被收集之后，将获得这个关卡的胜利。特殊的能量药丸短暂地给予了玩家角色能力来消灭鬼魂而不是被它们所消灭。就像在第 2 章所看到的那样，原版的《*吃豆人*》(Namco, 1980)以及它的后继者《*吃豆人小姐*》(Namco, 1982)之间的差异看起来很小，但实际上却是根本性的；自重要的一点是在《*吃豆人小姐*》(Namco, 1982)中的鬼魂有着非确定性的行为，让学习一个固定的动作序列来作为游戏的一个解变成了不可能。这个游戏对研究群体的吸引力被一份近期的调查所佐证，其涵盖了 20 多年来使用着两种游戏作为的测试平台的活跃 AI 研究[572]。

存在多种用于以吃豆人为基础的实验的框架，其中一些与竞赛有关。吃豆人屏幕捕获竞赛(The Pac-Man Screen Capture Competition)是基于原始游戏的 Microsoft Revenge of Arcade 版本的，并且没有提供一个用来加速游戏的前向模型或是工具[403]。吃豆人小姐对鬼魂团队竞赛(Ms Pac-Man vs Ghost Team Competition)框架由 Java 编写，并且包括了一个前向模型以及为游戏明显地加速的能力；它也包括了一个界面用于控制鬼魂团队而不是玩家角色吃豆人小姐[573]。《*吃豆人小姐*》(Namco, 1982)的雅利达 2600 版本作为 ALE 框架的部分是可用的。在加州大学伯克利分校还有一个用于教授 AI 的基于 Python 的吃豆人框架。

就像预期的那样，AI 玩家的表现根据前向模型的可用性而有所不同，其允许对鬼魂行为的模拟。没有提供一个前向模型的基于屏幕捕捉的竞赛被启发式方法所主导（其中一些包括了在迷宫中进行寻路而不考虑鬼魂移动）其表现出了初学的人类玩家的水平[403]。已经观察到的是，即便向前搜索一步，并且使用一个基于某种进化神经网络的局面评估器，都可以成为用来玩这个游戏的有效方法[402]。当然，搜索的比单层更深会产生额外的好处；然而，在《*吃豆人小姐*》(Namco, 1982)中引入的随机性使得即使在前向模型存在的情况下也会导致挑战。MCTS 已经被证明在这种情况下表现的很好[589, 523]。强化学习中的无模型方法也已经被用于玩这个游戏，并且取得了一些成功[56]。一般来说，在吃豆人小姐对鬼魂团队竞赛中最好的选手是以中级技能的人类玩家的水平进行游戏的[573]。在写下这本书的时候，微软的 Maluuba 团队报告说《*吃豆人小姐*》(Namco, 1982)已经在实践上被解决了（达到了最高的可能分数 999,990 分）。这个团推使用一个称为混合奖赏架构(hybrid reward architecture)的强化学习技术[738]，其将环境的奖赏函数分解为不同的，需要由对应数目的智能体所解决的强化学习问题（一组奖赏函数）。每个智能体根据所有智能体对每个动作的 Q 值的汇总来选择它的动作。

《*吃豆人*》(Namco, 1980)也可以针对体验而不是表现。在一系列的试验中，在游戏的复制版中控制鬼魂的神经网络的进化使得游戏对人类玩家来说更有趣味性[767]。这个实验在概念上是基于 Malone 对游戏中的乐趣的定义的，也就是挑战，好奇，以及幻想维度[418]，并且尝试寻找让这些特定最大化的鬼魂行为。特别是，适应度被分解为三个因素：1)挑战的

适合程度（也就是在游戏不会太难也不会太简单时），2)鬼魂动作的多样性，以及 3)鬼魂的空间多样性（也就是在鬼魂的行为是探索性而不是静态的时候）。这个用于发展有趣的鬼魂行为的适应度函数通过用户研究得到了交叉验证[771]。

3.4.3.2 超级马里奥兄弟

任天堂的标志性主机游戏《*超级马里奥兄弟*》(Nintendo, 1985)对于 AI 研究来说已经是非常流行了，包括了在进行游戏，内容生成以及玩家建模上的研究（使用这个游戏的研究在本书的其他部分进行了阐述）。其中很大的一个原因是从 2009 年开始的马里奥 AI 竞赛，并且包括了几个不同的轨迹，分别聚焦于游戏表现，以人类方式进行游戏以及生成关卡上[320, 717]。用于这个竞赛的软件框架基于《*无限马里奥兄弟*》(Notch, 2008)，一个基于 Java 并带有简单的关卡生成的《*超级马里奥兄弟*》(Nintendo, 1985)克隆版[706, 705]。这个竞赛软件的不同版本通常被称为马里奥 AI 框架或者马里奥 AI 基准，目前已经被用于多个研究项目。在接下来，我们将简单地将用于玩《*超级马里奥兄弟*》(Nintendo, 1985)，无限马里奥兄弟或者马里奥 AI 框架/基准的多个版本的方法称为玩“马里奥”。

马里奥 AI 的第一个版本可以可以比真实时间快数千倍进行模拟，但并不包括一个前向模型。因此学习一个马里奥兄弟智能体的第一次尝试是从对马里奥动作的直接状态观察中学习到某种函数[706]。在这个计划中，神经网络被发展来引导马里奥通过简单的程序化生成的关卡。输入是在以马里奥为中心的一个粗糙矩形中存在或不存在环境特征或是敌人，而输出被解释为在任天堂手柄（上，下，左，右）上的按下按钮。状态表示的展示请参见图 3.6。一个标准的前馈 MLP 架构被用于这个神经网络，并且适应度函数被简单地表示为这个控制者能够在每个关卡上前进多远。使用这个设置以及一个标准的进化策略，神经网络发展到了可以赢得部分关卡，但不是全部关卡的地步，并且通常以一个人类初学者的水平进行游戏。

然而，正如以往一样，拥有一个前向模型将会产生一个巨大的不同。在 2009 年，第一次马里奥 AI 竞赛由 Robin Baumgarten 赢得，其通过重新使用部分开源游戏引擎代码来为这个游戏构建了一个前向模型[705]。使用这个模型，它构建了一个基于在状态空间中进行 A* 搜索的智能体。在每个时间帧，智能体搜索到屏幕右边的最短路径，并且执行在产生的规划中最前面的动作。正因这个搜索利用了前向模型并且因此发生在了状态空间中而不只是物理空间中，它可以在它的规划中包含预测过的敌人移动（确定性的）。这个智能体能够完成所有使用于 2009 马里奥 AI 竞赛中的关卡，并且产生了在完成关卡的时间周期上显然是最佳的（它并不关注收集金币或是杀死敌人）。关于这个算法的解释以及一个展示了它在马里奥上的使用的图，参见章节 2.3.2。

鉴于 2009 竞赛中基于 A*的智能体的成功，这个竞赛下一年的版本更新了关卡生成以使的它产生更为有挑战性的关卡。重要的是，新的关卡生成器创造了包括“死路”的关卡，马里奥在其中可以采取错误的路径，并且一旦发生了就要回退去采用其他路径[320]。一个像这样的死路的例子，参见图 3.7。这些结构有效地“困住”了依赖简单的最佳优先搜索的智能体，因为它们的搜索将会大量的终止于接近当前位置的路径，并且在找到一条回溯到结构开始的路径之前将会超时。而 2010 马里奥 AI 竞赛的胜利者是 REALM 智能体[55]。这个智能体使用一个进化形成的基于规则的系统来在关卡的当前部分中决定子目标，并且在之后使用 A* 导航到这些子目标。REALM 在 2010 竞赛中成功地解决了存在于部分关卡中的死路，并且也是我们所知的当前拥有最佳表现的马里奥智能体。

在除了 A*以外的其它已经被用于玩马里奥的搜索算法中，也包括了蒙特卡罗树搜索[292]。已知的是，MCTS 的标准模式表现的并不好，因为算法搜索的不够深，还因为一个分支平均回报的方式会导致厌恶风险的行为。然而，经过一定的修改来解决这些问题后，一个

MCTS 变体被证明可以将马里奥玩的与纯粹的 A*算法一样好。。在后续的实验中，噪声被添加到马里奥 AI 基准中，并且发现 MCTS 比 A*更好的解决了这个添加的噪声，可能是因为 MCTS 依赖于回报的统计平均，而 A*假设了一个确定性的世界。

所有上述工作都集中于以表现为目标而进行游戏。在为体验而进行马里奥游戏上的工作主要集中于模仿人类玩家风格，或者是创造与类似人类玩马里奥的智能体。马里奥 AI 竞赛的一种图灵测试分支被创造来用于深入这项研究[618]。在这个分支中，参赛者提交智能体，并在它们在多个关卡中的表现被记录下来。智能体进行不同关卡的视频在这里与其他人类玩家玩相同关卡的视频一同被展示给人类观众，然后观众被要求指出视频中哪一个是人类玩家。智能体的得分基于它们可以多频繁地欺骗人类，与原始图灵测试的设置相同。结果指出，简单的，包含了硬编码的活动的启发式解决方法，例如有些时候静止不动（给出一种“思考下一个动作”的印象），或是偶尔误判一个跳跃可以非常有效地呈现人类的游戏风格。提供类人行为的另一种方式是从游戏轨迹中学习来显式地**模仿**人类。Ortega 等人描述了一种用于创造以特定人类玩家风格进行马里奥游戏的智能体的方法[510]：生成神经网络，其中适应度函数是基于智能体在面临相同状况时是否会与人类玩家表现相同的动作。比起生成一个有着更为直接的适应度函数的神经网络架构，其被证明可以产生更为类人的行为。在类似的创造类人马里奥 AI 玩家的努力中，Munoz 等人[469]同时使用玩家轨迹以及玩家在屏幕上的眼睛位置的信息来作为一个 ANN 的输入，其被训练来逼近在每个游戏步骤中哪个键盘动作将会被执行。它们的结果产生了一个很高的对玩家行为的预测准确性，并且显示了基于信息的类人马里奥控制者的发展超越了游戏玩法数据的可能性。

3.4.3.3 ALE 框架

Arcade Learning Environment (ALE)是一个基于对经典的视频游戏主机雅利达 2600 的模拟的用于通用游戏研究的环境[39]。（虽然在这个环境在技术上也可以适应其他模拟器，但雅利达 2600 模拟器是一个已经在实践中被使用的模拟器，以至于 ALE 框架有时候被简单地称为“雅利达”）雅利达 2600 是一种主机，带有 128 字节的 RAM，最大每个游戏 32KB 的 ROM，并且没有屏幕缓存，对系统上可以实现的游戏类型有着严重的限制[465]。ALE 提供了一个借口用于智能体通过标准操纵杆来控制游戏，但是不提供任何内部状态的处理版本；相反，它向智能体提供了一个 160×210 像素的屏幕输出，其将需要在某种程度上解析这种视觉信息。这里存在一个正向模型，但它相对较慢并且一般不被使用。

早期一些使用 ALE 的工作使用了神经进化；特别一份在 61 个雅利达游戏上比较了几种神经进化算法的研究[249]。他们发现他们可以使用流行的神经进化算法 NEAT 为单独游戏生成质量不错的玩家，前提是那些算法可以得到由某个计算机视觉算法所识别的各个游戏内目标的位置。HyperNEAT 算法，一个非直接编码的神经进化算法，其可以创造任意大的网络，可以基于原始像素输入而学习到一个能够进行游戏的智能体，甚至在被测试的游戏中的三个中超越了人类的表现。在这篇论文中，神经进化方法的通常比经典的强化学习方法在表现上要好得多。

之后 ALE 被用在了 Google DeepMind 在深度 Q-learning 上的研究中，在 2015 年其被报道于一篇《自然》上的论文中[463]。就像第二章所仔细阐述的那样，这项研究展示了通过训练一个深度深度神经网络（五层，其中最初的两层是卷积的）加上 Q-learning，再加上经验回放，在 49 个被测试的雅利达游戏中，可以有 29 个达到人类玩游戏的水平的表现。这项研究激发了一系列尝试改进这篇论文中提出的核心深度强化学习的实验。

值得注意的是，几乎所有的 ALE 工作都集中于为多个单独游戏学习神经网络（或者是偶然混杂的其他智能体表示）。也就是说，网络架构与输入表示在所有游戏中都是相同的，但是参数（网络权重）是针对单个游戏学习而来并且只能被用于玩这个游戏。这似乎与全局

游戏策略的思想有一些不一致，其的思想是，你可以学习玩不限于单个游戏，而是任意你给予它们的游戏的智能体。也可以注意的是，比起在全局游戏策略上的研究，ALE 自身更适合用于玩多个单独游戏的研究，因为只存在一个有限数量的雅利达 2600 游戏，并且为这个平台创造新游戏是非常不易的。这使得它可能会将架构甚至智能体调节到多个单独游戏上。

3.4.3.4 通用视频游戏 AI

通用视频游戏 AI(GVGAI)竞赛是一个基于游戏的 AI 竞赛，从 2014 年开始举办[527]。它部分是被设计来作为对某种可以在许多现有的基于游戏的 AI 竞赛上看到的倾向的回应，例如在那些组织于 CIG 与 AIIDE 会议的竞赛上，各种提交通过包含越来越多的领域知识而变得更为针对游戏特化。因此 GVGAI 的一个中心思想是竞赛的各个提交会在未见过的游戏上进行测试，换句话说，就是之前尚未放出给选手的游戏，因此其不可能为提交做出特殊调整。在撰写本文时，GVGAI 仓库包括了大约 100 种游戏，并且每次竞赛都会有多于十种的游戏被添加。

为了简化 GVGAI 中的游戏的开发，一个称为视频游戏描述语言(Video Game Description Language, VGDL)的语言被开发[179, 596]。这个语言允许使用一种类似 Python 的语法简洁地描绘游戏；一种特定的游戏描述大约 20-30 行，在不同的文件中指定级别。鉴于对 2D 移动以及图像逻辑的潜在猜想，在 GVGAI 语料库中的大部分游戏是由经典街机游戏所改造（或所启发）而来，例如《*青蛙过河*》(Sega, 1981)（见图 3.8(b)），《*推石小子*》(First Star Software, 1984)或《*太空侵略者*》(Taito, 1978)（见图 3.8(a)），但是也有一些是现代独立游戏的版本，例如《*雪人难堆*》(Hazelden and Davis, 2015)。

GVGAI 竞赛的最初路线是单人规划路线，其大部分结果是已知的。在这里，智能体被给定一个游戏的快速前向模型，以及 40 毫秒来使用它为下一个动作做出规划。鉴于这些条件，多种类型的规划算法成为主流也是有原因的。在这个路线中有着突出表现的大部分都是基于 MCTS 上的变体或者类 MCTS 算法，例如开环最大期望树搜索(Open Loop Expectimax Tree Search)算法[527]或者带有选择性搜索的 MCTS[160]。一个令人惊讶的高性能智能体使用了迭代宽度(Iterative Width)算法，其中的核心思想是为所有事实建立一个命题数据库，然后使用它将一个宽度优先搜索算法剪枝到只会探索那些有着一定量的新奇性的的分支，以第一次看到的事实的最小集合的大小来衡量。基于进化规划的智能体也表现的非常好，但没有那些基于随机树搜索或是带有新奇性剪枝的搜索所表现的那么好。

尽管一些智能体总体上比其他的更好，但在排名中却有着很明显的非传递性，也就是针对一个特定游戏的最佳算法可能对另一个游戏来说不是最好的——事实上，似乎存在一些模式，某种算法家族在某种游戏家族上表现的更好[211, 58]。鉴于这些模式，一种取得更高的变现性能的自然而然的想法是去使用**超启发式**(hyper-heuristics)或者**算法抉择**(algorithm selection)来选择在运行时哪一个算法被用于哪一个游戏，目前为止一个方法已经取得了一些成功[452]。,

另外两个 GVGAI 路线与游戏玩法有关，就是双人规划路线与学习路线。双人规划路线类似单人规划路线，但是具有一定数量的双人游戏，其中一些是合作性的而某些是竞争性的。在本文撰写的时候，在这个路线中的最佳智能体时单人路线智能体的轻微修改版本，其对其它玩家的行为做出了简单的猜想[214]；预计有着更为复杂的玩家模型的智能体最后会表现的更好。相比之下，学习路线以单人游戏作为特征，并且为玩家提供了时间去学习一个策略，但却没有提供一个前向模型。预计深度强化学习以及神经进化等算法将会在这里取得良好的效果，但目前暂时没有结果；可以想象，学习了一个前向模型并执行树搜索的算法将会占据主导地位。

3.4.3.5 其他环境

除了 ALE 以及 GVGAI，也有一些其他的可以被用于带有街机风格游戏的 AI 实验的环境。Retro Learning Environment 是一个在概念上与 ALE 类似的学习环境，不过其是基于对超级任天堂主机的模拟的[44]。一个更为通用，但更少关注度的系统是 OpenAI Universe，其可以作为大量不同游戏的统一接口，从简单的街机游戏都复杂的现代冒险游戏。

3.4.4 策略游戏

策略游戏，特别是计算机策略游戏，是指玩家控制多个角色或单位的游戏，并且游戏的目标是赢得某种形式的征服或是冲突。通常，但不会全部，叙述与图像都反映了一场军事冲突，其中单位可能是骑士，坦克或者战舰。在策略游戏中最重要的区别可能是回合制与即时战略游戏，其中前者为玩家留下了充足的时间来决定在每次采取哪一个动作，而后者则施加了时间压力。著名的回合制策略会游戏包括了史诗级策略游戏，例如《*文明*》(MicroProse, 1991) 以及《*幽浮*》(MicroProse, 1994)系列，也有更短的游戏，例如《*英雄学院*》(Robot Entertainment, 2012)。突出的即时战略游戏包括了《*星际争霸*》(Blizzard Entertainment, 1998)与《*星际争霸 II*》(Blizzard Entertainment, 2010)，《*帝国时代*》(Microsoft Studios, 1997–2016)系列以及《*命令与征服*》(Electronic Arts, 1995–2013) 系列。其他的区别是在集中于探索的单人游戏，例如文明游戏，以及如《*星际争霸*》(Blizzard Entertainment, 1998)这样的多人竞技游戏之间。大部分，但不是全部的策略游戏含有隐藏信息。

在策略游戏中的认知挑战是制定与执行涉及到了多个单位的复杂计划。这个挑战通常比其他传统棋盘游戏中的规划挑战更为困难，例如国际象棋，主要是因为在每个回合有多个单位必须被移动；一个玩家控制的单位数量可以很容易地超过短期记忆的极限。规划范畴也可以变得非常的长，例如在《*文明 V*》(2K Games, 2010)中，你做出的关于建立单个城市的策略将会在几百个回合内影响游戏进程。单位被移动的顺序可以很明显地影响单个移动的结果，尤其是因为某个单独的动作可能在一次移动中揭露新的信息，导致了一个对先后顺序的挑战。除此之外，还有在预测一个或多个对手的移动上的挑战，其经常拥有多个单位。对于即时策略游戏来说，还有额外的与游戏速度相关的感知以及肌肉运动挑战。这种认知复杂性反映在了智能体进行这些游戏的计算复杂度上——就像在章节 3.2.1.4 中探讨的那样，一个策略游戏的分支因子可以轻松地达到上百万甚至更多。

策略游戏的海量搜索空间与巨大的分支因子对于大多数搜索算法来说都是很严重的问题，因为仅向前搜索单个回合可能都已经是无能为力的了。处理这个问题的一种方式是分解这个问题，让单位各自为政；这为每个单位都创造了一个搜索问题，其分支因子相当于个体单位的分支因子。这具有易于处理的优点，以及阻止了单位间协调的缺点。尽管如此，单位被分开处理的启发式方法被用于许多策略游戏中内置的 AI。不巧的是，许多策略游戏的内置 AI 通常被认为是不足的。

在玩策略游戏的研究中，一些解决方法涉及到了巧妙地对回合空间进行子采样，一遍可以使用标准的搜索算法。这种方法的一个案例是一个基于使用朴素采样(Naive Sampling)来进行分解的 MCTS 变体[502]。另一种方法是非线性的蒙特卡罗(Monte Carlo)，其被应用于《*文明 II*》Civilization II (MicroProse, 1996)，并且有着非常有效的结果[64]。在这里的基本思路是随机地抽样回合空间（其中每个回合由所有单位的动作所组成），并且通过执行一次直到某个特定点的随机推演（采取随机动作）来为每个回合得到一个估计。基于这些估计，一个神经网络被训练来预测回合的值；然后回归可以被用来搜索有着最高预测值的回合。

但是训练并不需要基于树搜索。Justesen 等人将在线进化规划应用到了《*英雄学院*》(Robot Entertainment, 2012)，一个有着完美信息以及每回合相对低的招式数量（在标准设置下为五）的双人竞争性策略游戏[307]。每个染色体由在单个回合内要采用的动作所组成，而适应度函数是在回合结束时的物质差异。可以看出的是，这个方法大大地超越了 MCTS（以及其他树搜索算法），除了一个单独回合的浅搜索深度，可能是分支因子使得树搜索算法完全无法有效地探索甚至这一个回合的空间。

进化也被用于创造可以进行策略游戏的智能体。例如，基于 NEAT 的宏观管理控制器被训练用于游戏《Globulation 2》(2009)。然而，在这个研究中，NEAT 控制器的目标并不是为了赢得胜利，而是为了游戏上的体验；特别是，它演变为采取宏观行动（例如，建立计划，战斗规划），以便为所有玩家提供一个平衡的游戏[498]。基于人工进化的 AI 方法也已经被用于为许多其他的策略游戏用做游戏测试机制[587, 295]。

3.4.4.1 星际争霸

最早的星际争霸由暴雪娱乐于 1998 年发行，仍然被广泛地作为竞技游戏，也是一种对它强大的游戏设计，特别是它在挑战上罕见的深度（见图 3.10）的一种证明。它通常伴随着母巢之战拓展来玩，并称为 SC:BW。母巢之战 API(Brood War API , BWAPI)的存在，也就是一个使用人工智能体来玩这个游戏的接口，已经成就了一个繁荣的，共同致力于一个用于《*星际争霸*》(Blizzard Entertainment, 1998)的 AI 的研究者与爱好者社区。每年都会会有少数基于 SC:BW 与 BWAPI 的竞赛被举办，包括了在 IEEE CIG 与 AIIDE 会议[86]上的竞赛。TorchCraft 是一个建立于 SC:BW 与 BWAPI 上来促进使用星际争霸的机器学习研究的环境，特别是深度学习[681]。与此同时，一个相似的 API 在最近被发布，用于将 AI 智能体与后续游戏《*星际争霸 II*》(Blizzard Entertainment, 2010)进行接口，其在机制与概念上非常相似，但有着许多技术上的不同。鉴于现存的 API 与竞赛，几乎所有现有的研究都已经使用 SC:BW 来完成。

由于 SC:BW 是如此复杂的一个游戏，并且将精通它的挑战是如此巨大，大部分研究关注在这个问题的部分之上，最常见的是通过某种层次的抽象来进行游戏。通常会将 SC:BW（以及类似的即时战略游戏）中的决策的不同等级划分为三个等级，取决于时间规模 :策略，战术以及微观操作（见图 3.9）。目前为止，尚没有智能体能够以甚至一个人类中级玩家的水平完成一场完整的游戏；不过，已经在进行这个游戏的几个非常棘手的问题的某些子问题上取得了明显的进步。

关于针对玩 SC:BW 游戏的 AI 研究的完整概述，读者可以参阅一份近期的综述[503]。下面，尽管没有完整覆盖，但我们将举例说明这个领域内已经完成的一些研究。

在**微观**层次，AI 通常在小于一分钟的时间范畴内进行游戏，其中采取动作相间隔的时间通常是一秒上下。当集中于这个 SC:BW 战役的最低级形式时，没有必要考虑基地建筑，研究，战争迷雾，探索以及完整 SC:BW 游戏的许多其他方面。通常来说，两个阵营相互对峙，每一边都有一组的几个或是几十个单位。目标是去摧毁敌人的单文，这个游戏模式可以在实际游戏中进行，其并不允许明显的加速而且不提供一个前向模型，还有就是在模拟器 SparCraft[122]中，其提供了一个前向模型。（也存在一个 SparCraft 的 Java 版本，称做 JarCraft[309]。）

在无模型的情况中，没有奢侈的前向模型，智能体必须基于手动定制的策略或是通过强化学习或监督学习学习而来的策略。手动定制的策略可以基于潜在领域等来实现，在其中，为了创造有效的战斗模式，不同的单位被其他单位所吸引或排斥[240]，或者以模糊逻辑(fuzzy logic)[540]。对于无模型常见下的机器学习方法，标准的[621]以及深度强化学习[729

已经被小有成效地被用于学习策略。通常情况下，问题会被分解，一遍学习单独的 Q-函数，其之后会被分别应用于每个单位[729]。

使用 SparCraft 模拟器，由于前向模型的可用性，我们可以做的更多。Churchill 与 Buro 开发了一个称为项目组合贪婪搜索(Portfolio Greedy Search)的简单方法来处理巨大的分支因子[122]。核心思想是，不为每个单位在动作之间进行选择，为数不多的（一个项目组合(portfolio)）称为脚本的简单启发式被采用，并且一个贪婪搜索算法被用于将这些脚本赋予每个单位。这个方法戏剧性地修剪了分支，但将可探索的策略的空间限制到了可以被描述为脚本组合的策略的空间中。随后，发现脚本的项目组合思想可以与 MCTS 结合并产生良好的结果[309]。而通过进化规划来完成更好项目组合选择甚至可以获得更好的结果[746]；这些想法很可能由于它们能够同时控制多个单位而被推广到许多策略游戏或其他有着高分支因子的游戏中。

转到微观操作-战术-策略连续体的另一端，大规模的**策略**适应仍然是一个非常困难的问题。现有的 SC:BW 机器人很少能够实现多种策略，更不用说根据游戏的进度来调整它们的策略了。为了完成这一点，需要基于有限的证据来创造一个关于对手正在尝试做什么的模型。在这里，Weber 与 Mateas 的前沿工作集中于挖掘 SC:BW 匹配的日志，来从游戏早期的动作中预测一个玩家将会采取什么策略[751]。

还有一些更为雄心壮志的尝试，在创造完整的，能够以一个原则方式处理策略，战术与微观操作的智能体。例如，Synnaeve 与 Bessière 构建了一个基于贝叶斯编程的智能体，其有着合理优秀的表现[680]。

3.4.5 竞速游戏

竞速游戏是玩家负责控制某种形式的车辆或是角色以在最短内到达某个目标的游戏，或者在一个给定的时间内沿着赛道尽可能远地移动。通常这个游戏使用一个第一人称视角，或是玩家控制的车辆之后的一个有利点。竞速游戏大部分都采用一个连续的输入信号作为一个转向输入，类似于一个方向盘。某些游戏，如那些在《*竞速飞驰*》(Microsoft Studios, 2005–2016) 或《*真实赛车*》(Firemint and EA Games, 2009–2013)系列中的，允许包括变速杆，离合器与手刹在内的复杂输入，而更像《*极品飞车*》Need for Speed (Electronic Arts, 1994–2015)系列中那些更为街机化的游戏中，通常有着一个简单的输入集合，因此有着更低的分支因子。像《*反重力赛车*》(Sony Computer Entertainment Europe, 1995–2012) 以及《*马里奥赛车*》(Nintendo, 1992–2017)系列中的那些竞速游戏引入了额外的元素，例如可以被用来短暂地瘫痪对手车辆的武器。

尽管在进行竞速游戏时的认知挑战可能看起来很简单，但大多数竞速欧系实际上需要同时执行多个任务，并且有着明显的技能深度。在最基本的关卡的中，智能体需要根据车辆的位置来进行控制并且调整加速或刹车，使用良好调整的连续输入，从而尽可能快地通过赛道。要将这个做到最优至少需要短周期的规划，向前一或两圈（赛道）。如果在游戏中存在要被管理的资源，例如燃油，损伤或者速度提升，这需要长周期的规划。当其他车辆出现在赛道上时，在尝试或阻止超车时，还会加入一个对抗性规划的角度；这个规划经常完成于存在隐藏信息的情况下（在赛道的不同部分的其他车辆的位置与资源），并且有着可观的时间压力，可以受益于对手驾驶员的建模。

一个相对早期的突出商业游戏应用是用于《*竞速飞驰*》(Microsoft Studios, 2005)的 AI，其以 Drivatar 的名字进行推广[257]。Drivatar 智能体构建在有监督的惰性学习(lazy learning)的一种形式上。为了训练这个智能体，人类驾驶一定次数的比赛赛道，其被分解为多个部分；游戏中的所有赛道需要被分解为来自相同“字母”的部分。在驾驶时，智能体选择最为接近玩

家在相关部分完成的比赛线路的驾驶指令。这个方法成功地实现了个性化的驾驶智能体，也就是能够以它们被训练的人类人家的风格驾驶新赛道的智能体，但是对赛道的设计也提出了要求。

通过强化学习，也存在多种方法来在没有监督的情况下训练智能体去驾驶。一系列论文展示了神经进化如何在缺乏其他车辆以及一个优秀的人类驾驶员的情况下被用于训练驾驶某个单独赛道的智能体[707]，渐进式进化(incremental evolution)如何被用于训练有着充足通用驾驶技巧的智能体去驾驶未见过的赛道[709]，以及竞争性的共同进化如何被用于在缺乏其他车辆的情况下对抗性地训练智能体在驾驶上有着更多或更少的进攻性[708]。在所有这些实验中，一个相对小的固定拓扑网络的权重由一个进化策略训练而来。对这个网络的输入是车辆的速度以及少量会返回到赛道边缘的距离的测距传感器。由此产生的网络的低维度使得高性能的网络相对容易被找到。

模拟车辆竞速锦标赛(Simulated Car Racing Championship)自 2007 年以来每年举办一次，其部分基于这项工作，并且使用一个类似的传感器模型。第一年，这个竞赛基于简单的 2D 竞速游戏，并且竞赛的胜利者是一个基于模糊逻辑的控制者[710]。在 2008 年，竞赛软件围绕着 TORCS，一个有着合理的随机物理模型的 3D 竞速软件而进行了重建[392]（见图 3.11）。在接下来的几年中，许多选手提交了基于多种不同架构的智能体到这个竞赛，包括了进化计算，差分学习，监督学习与简单的硬编码的基于规则的系统[392]。在竞赛过程中观察到了一个普遍的趋势，也就是胜利的智能体以硬编码的形式包括了越来越多的领域知识。最佳的智能体，例如 COBOSTAR[89]或是 Mr.Racer[542]在某条赛道上单独驾驶时通常与一名优秀的人类驾驶员旗鼓相当甚至更好，但在超车以及其他形式的对抗性驾驶上仍然在苦苦搏斗。

如上所述，模拟车辆竞速锦标赛以一种相对容易映射到驾驶指令的形式提供了信息，使得至少基本驾驶策略（但并非良好调整的）的学习相对容易。然而，某些作者已经尝试着从原始数据中学习驾驶。在这之上的早期工作包括了 Floreano 等人的工作，其开发了一个带有某种可移动的“视网膜”的神经网络来在一个简单模拟环境中驾驶。这个神经网络的输出同时包括了驾驶命令以及如何移动视网膜的命令，并且只有在视网膜中相对不多的像素被用做到网络的输入[204]。之后，Koutnik 等人尝试在压缩过的权重空间中通过发展网络来开发使用高维度输入的控制者；本质上来说，网络连接的某种 JPEG 编码的各个参数得到了开发，允许进化搜索在巨大的神经网络空间中有效地运作[352]。深度网络的监督学习也已经被应用于视觉驾驶中，产生了从案例中学习而来的高兴能 TORCS 驾驶者。

上面的例子并没有使用任何类型的前向模型。然而，汽车机动对模型来说相对简单，并且很容易为竞速游戏创造一个快速并且接近的模型。给定这样的一个模型，树搜索算法很容易被应用到汽车控制上。例如，Fischer 等人表明了 MCTS 加上一个简单的前向模型可以在 TORCS 中产生很不错的表现[201]。

3.4.6 射击与其它第一人称游戏

第一人称设计自《*毁灭战士*》(GT Interactive, 1993)以及《*德军总部 3D*》(Apogee Software and FormGen, 1992)在 90 年代早期的成功之后就成为了视频游戏的一个重要类别。虽然 FPS 的一个基本准则似乎是要通过一个第一人称视角来观察世界，但此外还有些游戏也被认为是 FOS，例如《*战争机器*》Gears ofWar (Microsoft Studios, 2006–2016)系列，其相机位置略微地在玩家之后和/或之上。类似的，“射击”词也意味着这个游戏以某种武器的子弹射击为主。在这个基础上，像《*传送门*》Portal (Electronic Arts, 2007)这样的游戏可以被视为一个 FPS 游戏，尽管玩家的工具是否真的是一个武器还存在争议。

射手往往被视为快节奏的游戏，其中感知与反应的素质是至关重要的，并且这在一定程

度上是正确的，尽管游戏过程的速度在不同的射击上各自不同。很明显，快速的翻译通常对于计算机程序来说不会是一个问题，这意味着一个 AI 玩家在默认情况下比起一个人类有着确定性的优势。但也存在其它的认知挑战，包括了在一个复杂的三维环境中的定位以及移动，并且在某些游戏模式中也有基于团队的合作。如果使用视觉输入，还有一个额外的关于从像素中提取相关信息的挑战。但针对 FPS 的 AI 的大量工作是由两个竞赛所激发的 :第一是 **2K BotPrize**，以及更为近期的 **VizDoom**。

3.4.6.1 虚幻竞技场 2004 与 2K BotPrize

《*虚幻竞技场 2004*》(UT2k4) (Epic Games, 2004)是发行与 2004 年的 FPS，有着当时最先进的图像技术与游戏玩法。尽管游戏本身没有被开源，但位于布拉格查理大学的一个团队创造了 Pogamut，一个基于 Java 的 API，其允许对游戏进行简单的控制[220]。Pogamut 为智能体提供了基于目标的信息界面，其中智能体能够查询目标与角色的位置，并且也为执行像向特定地点发射子弹这样的动作提供了便利的函数。

某些使用 UT2k4 的工作尝试为一个或多个游戏内的任务实现高性能的智能体，使用像神经进化这样的技术。例如，van Hoorn 等人将进行 UT2k4 的任务细分为三个子任务 :射击，探索以及路径跟踪[734]。使用一个更早的方法[698]，其将神经进化与 Rodney Brooks 的包容结构[69]相结合，他们之后这些任务中的每一个都生成了神经网络。产生的智能体能够相对有效地玩进行些游戏场景。

然而，UT2k4 基准的主要使用是在 **2K BotPrize**（见图 3.12）中。这个竞赛，从 2008 年举办至 2014 年，在基于游戏，但不集中于游戏表现而是游戏体验的 AI 竞赛中脱颖而出。特别是，它是图灵测试的一种形式，其中被提交的智能体被不是根据它们在与其他智能体的交火中生存的有多好来判断的，而是根据它们是否可以欺骗人类的判断（其在之后的比赛设置中也参加了游戏）来认为它们是人类[260, 261, 262]。

2014 年最后一场 2K BotPrize 的胜利者是两个让过半的人都判断它们（机器人）是人类的团队。第一个胜利的团队，UT2，来自德克萨斯大学奥斯汀分校，主要是基于通过多目标进化的神经进化[602]。它由多个独立的控制器所组成，其中的大部分是基于神经网络；在每一帧，它遍历循环所有这些控制器，并且使用一组优先级来决定哪些控制器的输出将会指令智能体的多个方面。除了神经网络，某些控制器是建立在不同的原则之上的，特别是人类追踪控制器，其使用人类玩家的轨迹来帮助智能体走出被卡住的位置。第二个胜利者，Mihai Polceanu 的 MirrorBot，是围绕着观察游戏中的其它玩家并且模范他们行为的思路搭建的。

3.4.6.2 原始屏幕输入与可视化 Doom AI 挑战

VizDoom 框架是围绕着经典的《*毁灭战士*》(GT Interactive, 1993)FPS 游戏的一个版本搭建的，其允许研究者开发仅使用屏幕缓存来玩这个游戏的 AI 机器人。VizDoom 被波兹南理工大学计算机科学学院的一个研究者团队开发为一个 AI 测试平台（见图 3.13）。这个框架包括了几个不同复杂度的任务，从健康包手机和迷宫导航到全力以赴的死亡竞赛。一个基于 VizDoom 的年度竞赛自 2006 年开始举办于 IEEE CIG 会议，并且这个框架也被包括在了 OpenAI Gym，一个可以被用于 AI 研究的游戏集合中。

大部分已发表的在 VizDoom 之上的工作都是基于带有卷积神经网络的深度强化学习，这是鉴于该方法在基于原始像素输入时学习到的行为已经得到了证明的水平而导致。例如，Arnold，一个在首届 VizDoom 竞赛中表现优异的智能体，是基于两个不同的神经网络的深度强化学习的，一个用于探索，一个用于战斗[114]。

但是也可以使用进化计算来训练神经网络控制器。因为非常巨大的输入空间需要巨大的网络，其通常对进化优化表现不佳，必须要在某种程度上压缩信息。这可以通过使用一个训练与伴随游戏进行的视频流上的自动编码器来完成；自动编码器的瓶颈层的激活之后可以被作为到某个决定动作的神经网络的输入，并且这个神经网络的权重可以被进化[12]。先前在相关游戏《*雷神之锤*》(GT Interactive, 1996)中作用在视觉输入之上的进化控制器的尝试只取得了有限的效果[518]。

3.4.7 严肃游戏

严肃游戏类别，或者有着某种超越了娱乐的**目的的游戏**，已经游戏 AI 中成为了近期研究的一个关注领域。有人可能会讨论说，大部分现有的游戏本质上是严肃的，因为它们在游戏过程中为玩家提供了某种形式的学习。例如，像 Minecraft (Mojang, 2011)这样的游戏，并没有特意设计出具体的学习目标；尽管如此，它们已经在教学中被广泛地用于科学教育。除此之外，也有人认为严肃游戏并不具有自己的特定类别；游戏可能拥有一种不限于它们被设计的类别的目的。严格地来说，严肃游戏的设计涉及到了一组特定的学习目标。学习目标可能是教育目标，例如那些在 STEM 教育中被考虑的——这样一类游戏的一个流行案例是《*龙箱*》(WeWantToKnow, 2011)系列，其会教小学生等式的解决技巧，以及基本的加减技巧（在基于游戏的 STEM 教育上的一项重要的学术努力是以叙事为中心的《*Crystal Island*》系列游戏系列，用于有效的科学训练[576, 583]。）然而，这个学习目标可以称为社交技能的训练，例如通过游戏解决冲突或是社会融入；《*Village Voices*》[334]（见图 3.14(a)），《*My Dream Theater*》[99]（见图 3.14(b)），以及 Prom Week[446]是这些技能训练游戏的案例。或者，目标也可以是遭受创伤后应激障碍的战争老兵被训练在面临游戏中的应激物时去协调他们的认知行为显示，像是在《*StartleMart*》[270, 268]以及《*Virtual Iraq*》[225]这样的游戏中。这个学习目标也可以是为科学家募集机体智慧。许多科学游戏在近期已经通过群体游戏(crowdplaying)（或者也可以说人类计算(human computation)）探索到了新的知识；可以说最受欢迎的科学探索游戏是《*Foldit*》[137]，通过其玩家可以共同探索一种新的蛋白质折叠算法。

玩一个研究游戏所需要的认知与情绪技巧很大取决于游戏以及潜在的学习目标。一个关于数学的游戏通常会需要计算以及问题解决能力。而一个关于应激接种与暴露疗法的游戏将需要认知行为应对机制，认知评估的元认知和自我控制。玩家所需要的认知与情绪技巧的广度与一款认知游戏能够融进它的设计中的不同学习目标的数量一样广泛。

许多严肃游戏拥有 NPC，以及可以帮助让这些 NPC 变得更为可信赖，类人，社交化以及具有表达性的 AI。在严肃游戏中的 AI 通常对 NPC 行为建模与作为 NPC 进行游戏来说非常有用，但并不是为了胜利；而是为了游戏的体验。无论这个游戏是用于教育，健康还是出于模拟的目的，为了增强学习或是游戏的参与度，NPC 智能体需要可信赖并且情绪化地进行行动。多年来在情况计算与虚拟智能体的领域中的活跃研究一直致力于这项任务。遵循的常见方法是构建自上到下（被特定设计）的智能体结构，其代表了多种认知，社交，情绪与行为能力。在传统上，关注点一般同时在对智能体行为的建模以及在特定情况下它的适当表达之上。构建一个智能体行为的计算模型的一种普遍方法是将其基于某种理论认知模型，例如 OCC 模型[511, 181, 16, 187, 235]，其依赖一组接收到的刺激来尝试影响人类的决策，评估以及应对机制。对于有兴趣的读者，Marsella 等人[427]彻头彻尾的归纳了大部分常见的用于智能体的情绪计算模型。

类似 Greta [533]与 Rea [104]这样的有表达性以及可信赖性的对话智能体，或是体现了情况表现的虚拟人类[674]可以被考虑在严肃游戏设计当中。像这样对角色模型的使用已经

在用于教育与健康目的的智能辅导系统[130]，实体对话系统[103, 16]以及情感智能体中已经占据了主要优势。像这样的智能体架构系统的知名案例包括了 Lester 团队在《*Crystal Island*》系列游戏的工作[577, 576, 583]，《*Prom Week*》[446] 与《*Façade*》[440]的表达性智能体，《*World of Minds*》[188]游戏的智能体，以及 FAtiMA[167]产生了《*My Dream Theater*》[99]的智能体。

3.4.8 互动小说

尽管存在少数变化，但互动小说类型中的游戏通常包含了一个由类似房间这样的小型区域组成的幻想世界；然而，一个模拟环境并不是不可或缺的。重要的是，玩家需要使用文本命令去玩这个游戏。玩家通常与各个对象以及存在的游戏角色交互，收集对象并将它们存储在他的物品栏中，并且解决多种难题。这个系列的游戏也被称为文字冒险游戏或是文字角色扮演游戏。著名的案例包括了《*魔域帝国*》(Infocom, 1979–1982)系列游戏以及《*Façade*》[440]。

在这个游戏种类中，AI 可以扮演理解在自然语言格式下来自玩家的文本的角色。换句话说，AI 作为玩家的同伴或是敌人进行游戏时可以被赋予**自然语言处理**(NLP)能力。进一步的 NLP 可以被用作与玩家的互动小说中的一段对话，一段文本或是一个故事的的输入。通常情况下，基于文本的输入被用来驱动一个故事（交互式叙事），其经常通过具现化的对话智能体来交流，并且通过某个虚拟摄影机的镜头来表现。与传统的电影摄制相似，摄影机的机位以及所交流的叙事都为观众的体验做出了贡献。然而与传统的电影摄制相反的是（但与交互式戏剧相似），游戏中的故事可以被玩家自身所影响。不言而喻，基于文本的游戏中的研究很自然地会与可信的交谈智能体（就像在前面章节提及的那样），计算与交互性叙事[693, 440, 561, 793]以及虚拟电影摄影中的研究有所交织。交互式叙事以及虚拟电影摄影之间的相互作用的探讨在第 4 章进行了拓展，特别是致力于叙事生成(narrative generation)的章节。

游戏中基于文本的 AI 上的工作开始于早期的与 Eliza[752]的互动以及被用于像《*魔域帝国 I*》(Infocom, 1980)这样的文字冒险游戏的 Z-Machine，到《*Façade*》[437, 440]以及近期用于玩 Q & A 游戏[338]以及文字冒险游戏[351]的 word2vec [458]方法（例如，它的 TensorFlow 实现[2]）。要注意的是，除了 AI 在理解自然语言上的用途，我们也可以使用 AI 来玩基于文本的游戏，关于同时处理这两个任务的一个值得注意的近期案例是一个由 Kostka 等人开发的智能体[351]，称为 Golovin。Golovin 智能体使用相关的语料，例如奇幻书籍来创造适合这个游戏领域的语言模式（通过 word2vec [458]）。为了玩这个游戏，智能体使用五中类型的命令生成器：战役模式，收集物品，物品栏命令，通常行动与移动。Golovin 可以用于 50 个互动小说游戏，展现出了与顶尖水平一较高下的水平。另一个案例是 Narasimhan 等人的智能体[475]，其被用于玩多人地下城游戏，也就是多人或是团体互动小说的一种形式。它们的智能体使用长短时记忆模型(LSTM)网络来将文本表示转化为状态表示。这些表示被提供给一个深度 Q 网络，其学习了一个给定的游戏状态中为每个动作的近似评估[475]。它们的方法在小型甚至中型游戏中就完成的任务而言有着超出其它基准的表现。对于有兴趣的读者，一个专门的在文字冒险游戏 AI 上的年度竞赛与 2016 年的 IEEE CIG 会议一同启动。竞赛的参与者需要提交针对 Z- Machine 来玩游戏的智能体。

针对基于文本的游戏的可用开发工具的例子包括了 Inform 系列的设计系统与编程语言，其受到了 Z- Machine 的启发，并促进了几种基于文本的游戏以及基于自然语言的互动小说的发展。值得注意的是，Inform 7 被用于《*Mystery House Possessed*》(Emily Short, 2005)游戏的设计。

3.4.9 其它游戏

AI 可以玩的游戏的列表并不局限于上面所涵盖的种类。尽管在我们看来，我们更为具体描述的这些类别是最具有代表性的。也存在一些关注其它类别游戏的 AI 游戏，我们在下面概述了其中的一些。

一种人气逐渐上升的游戏类型是**休闲游戏**，这是由于它们近年来不断增长的流行度以及可以通过移动设备访问而产生的。休闲游戏通常很简单，并且被设计为较短的游戏篇章（关卡），以允许在游戏时间上的灵活性。这个特性给予了玩家在短时间内结束一个篇章的能力，而不需要保存游戏。结果就是，玩家能够在短时间内专注于单个关卡，也可以在一整天内玩多个关卡，甚至在多达几个小时的游戏时间内反复地玩新关卡。玩休闲游戏所需要的游戏技巧取决于休闲游戏的类型，从像《*宝石迷阵*》(PopCap Games, 2001)，《*愤怒的小鸟*》(Chillingo, 2009)，和《*割绳*》(Chillingo, 2010)这样的解谜游戏，到《*梦之旅*》(KatGames, 2007)这样的冒险游戏，再到如《*美女餐厅*》 (PlayFirst, 2004)这样的策略游戏，《*植物大战僵尸*》(PopCap Games, 2009)与《*吞食鱼*》(PopCap Games, 2004)这样的街机游戏，《*探索博彩岛*》(funkitron, Inc., 2006)这样的棋牌游戏，其类型有着巨大的差异。

在休闲游戏上值得注意的学术努力包括了 Isaksen 等人的工作[286, 287]，在其中一个 AI 智能体被创建来测试《*Flappy Bird*》(dot- GEARS, 2013)关卡的难度。基本的 AI 玩家遵循一个简单的寻路算法，其在完成《*Flappy Bird*》关卡上表现优异。然而，为了模范人类的玩法，这个 AI 玩家被赋予了人类运动技巧的元素，例如精确度，反应时间以及每秒的动作数目。在这个工作线上的另一个案例是在《*割绳*》(Chillingo, 2010)的一个变体中使用 AI 智能体去测试游戏关卡的生成。Ropossum 创作工具中包含的这个 AI 智能体即表现出了自动化的测试，也使用一阶逻辑来优化了关卡的可玩性[613]。Ropossum 的关卡生成元素在之后的章节有着进一步的讨论。另一个在近期吸引了游戏 AI 研究的兴趣的休闲游戏是《*愤怒的小鸟*》(Chillingo, 2009)。这个游戏已经建立了一个 AI 竞赛[559]，名为愤怒的小鸟 AI 竞赛(Angry Birds AI Competition)，其始于 2012 年，主要与 International Joint Conference on Artificial Intelligence(IJCAI)一同开展。《*愤怒的小鸟*》(Chillingo, 2009)中的 AI 方法到目前主要集中于规划以及推理技术。例如一个定性的空间推理方法，其评估了关卡结构属性以及游戏规则，并且推测这些中的哪一个满足这个关卡中的每个建筑块[797]。每个建筑块的有用性（例如，击中它会有多好）在之后根据这些需求进行计算。其他的方法模型离散了关于《*愤怒的小鸟*》(Chillingo, 2009)当前游戏状态的知识，之后基于回答集编程(answer set programming)的拓展来尝试满足建模得到的世界的约束[91]。

在休闲游戏之外，**格斗游戏**类型也从学术界以及工业界玩家中收到了客观的兴趣。格斗游戏需要的认知技能主要与动觉控制以及空间导航有关，但也与反应时间以及决策有关，着二者都需要很快[348]。流行的用于格斗游戏的方法包括了经典的强化学习——特别是，用于借由线性或是 ANN 函数逼近器表示的 Q 值的在线学习的 SARSA 算法——就像被微软研究组应用在《*道风：莲花之拳*》(Microsoft Game Studios, 2003)游戏中[233]那样。强化学习被应用于格斗游戏中的自适应难度调节是也有着不同级别的成功[157, 560, 467]，也就是针对玩家体验的 AI。进化强化学习变体也针对这个任务进行了研究[436]。一个值得注意的在格斗游戏 AI 研究上的工作是由格斗游戏 AI 竞赛(Fighting Game AI Competition)[401]（见图 3.15）所提供的基于 Java 的格斗游戏 FightingICE。这项竞赛由日本的立命馆大学所组织，始于 2013 年，旨在推出最强的格斗机器人；也就是以胜利为目的的 AI。用于 FightingICE 的方法包括了动态脚本[416]，k-近邻[761]，蒙特卡罗树搜索[791]，神经进化[356]以及其它一些方法。到目前为止，基于 MCTS 的方法似乎最有利于在格斗游戏中取胜。

我们在这个章节要涵盖的最后一个游戏是《*我的世界*》(Mojang, 2011)，这是由于它作为一个用于游戏 AI 研究测试平台的独特属性。《*我的世界*》(Mojang, 2011)是一个在某个玩家可以游历的 3D 程序化生成世界中进行的**沙盒**游戏。这个游戏赋予了让玩家从立方体建立建

筑的游戏机制，但它并没有一个特定的目标来让玩家去达成。除了探索以及建筑，玩家也可以收集资源，制作物品以及与对手战斗。这个游戏在所有平台上已经卖出了超过 1.21 亿份[50]，让它成为了有史以来第二畅销的视频游戏，仅次于《*俄罗斯方块*》(Alexey Pajitnov and Vladimir Pokhilko, 1984)之后。《*我的世界*》(Mojang, 2011)的 3D 开放世界形式以及对具体目标的缺乏为玩家提供了以不同方式去畅玩以及探索世界的终极自由。玩《*我的世界*》(Mojang, 2011)的益处似乎有很多，并且其中的某些已经被教育研究社区所报告[479]。例如，在游戏中可用的方块可以被安排来生成一个玩家可以想象到的任何物品，并因此培养了玩家的创造性[479]以及图式水平思考方法[775]。除此之外，方块可以被组合与拓展的和功能可以导致玩家逐渐地获取新知识。还有就是，游戏简单的三维像素风格图像使得玩家在一个简单也美观的环境中集中于游戏体验以及探索任务。总的来说，让《*我的世界*》(Mojang, 2011)能够吸引成百万用户的诸多理由也是让这个游戏成为一个 AI 以及 AI 研究的优秀测试平台的理由。特别是，这个游戏为 AI 玩家提供了一个准备去探索的开放世界，有着无限度的游戏时间以及多种多样的可能性。除此之外，对一个 AI 智能体来说游戏内的任务也千面万化，从探索以及寻找财富到制作物品，建造建筑，无论是单独或或是作为一个智能体团队。

在《*我的世界*》(Mojang, 2011)AI 上一项值得注意的近期工作是 Project Malmo[303]，其由微软研究院所支持。Project Malmo 是一个基于 Java 的 AI 实验平台，作为原始游戏的一个游戏 mod，被设计来支持在机器人，计算机视觉，机器学习，规划和多智能体系统，以及通用游戏 AI 领域的研究[303]。注意这个开源平台可以通过 Github 获取。在 Project Malmo 中的早期 AI 实验包括用于在 3D 迷宫中导航的深度神经网络[464]以及与游戏中的对手进行对战[726]。除了 Project Malmo，还有一点值得注意的是这个游戏的许多 mod 已经被直接地用于教授机器人学[11]——包括用于迷宫导航与规划的算法——以及教授常见的 AI 方法[37]。

3.5 进阶阅读

用于玩游戏的方法在文献中有着更详细的拓展，其在第 2 章以及这里都有着详细的涵盖。我们在这个章节涵盖的不同游戏种类包含了对应的文献，有兴趣的读者可以使用它作为进一步探索的出发点。

3.6 练习

在转到关于内容生成的下一章之前，在这里我们提供了一个通用的练习，能够让你将第 2 章中涵盖的任意算法应用到一个单独领域中。联系的目的是让你在更为麻烦与复杂的领域与任务中测试它们之前先熟悉这些方法。

如上所述，吃豆人小姐对鬼魂团队竞赛是一个在全球多个 AI 会议上举行的比赛，其中用于吃豆人小姐以及鬼魂团队的 AI 控制者互相争夺最高的排名。在这个练习中，你必须开发一些吃豆人小姐 AI 玩家来与包括在软件包内的鬼魂团队控制者进行对抗。这是一个完全由 Java 编写的模拟器，有着良好文档化的接口。尽管在一个学期内选择两到三个不同的智能体已经被证明是一种良好的教育实践，但我们仍将留下你或是你的课程指导者所开发的吃豆人小姐智能体的最终数目。

本书的网站包含了用于游戏的代码以及许多不同的 Java 示例代码类来帮助你开始。所有涵盖于第 2 章的 AI 方法都能被用于控制吃豆人小姐的任务中。不过你可能会发现，其中的某一些比起其它的来说可能是更为贴近并且有效。所以哪一个方法将会运作的最好呢？就表现而言，它们如何相互比较？你应当使用哪一种状态表示？哪一个效用函数是最适合的？

你认为你能否做出正确的实现决定，来让你吃豆人小姐表现出职业水平？如何表现出世界冠军，甚至超越人类的水平？

3.6.1 为什么是吃豆人小姐？

我们的某些读者可能会反对对这个游戏的选择，并且认为应该有更有趣的 AI 可以玩的游戏。尽管《*吃豆人小姐*》(Namco, 1982)非常老，但它可以说一款经典的游戏，其仍然十分有趣并且在游戏内容上仍然充满挑战，而且也是一个简单的用于开始尝试本章以及前面章节介绍的各种 AI 方式与方法的测试平台。《*吃豆人小姐*》(Namco, 1982)易于理解与进行游戏，但并不容易精通。玩法的简单程度上的争议元素与温度复杂度的结合使得《*吃豆人小姐*》(Namco, 1982)成为用于控制主角的不同 AI 方法的理想测试平台。这个游戏另一个令人兴奋的特点是它的非确定性质。随机性并不只是增加了游戏的娱乐元素，并且也为任何被考虑过的 AI 方法增加了挑战性。就像前面所说的，选择《*吃豆人小姐*》(Namco, 1982)的另一个依据是，这个游戏以及它的变体已经在文献中得到了很好的研究，被几个游戏 AI 竞赛所检验，并且在许多年内被全球各地的多个大学涵盖于 AI（或是游戏 AI）课程之中。读者也可以参考近期一份由 Rohlfshagen 等人所写的综述[572]，涵盖了 20 年来在吃豆人中的研究，还有一个关于吃豆人在游戏 AI 研究中的重要性的 YouTube 视频。

3.7 总结

在本章中，我们讨论了 AI 可以扮演的不同角色，以及游戏与 AI 方法拥有的不同特征，多种可以用于进行游戏的方法以及它可以玩的不同游戏。特别是，AI 可以以获胜作为目标，也可以是为一个人类玩家或者观察者创造一种特殊的体验。前一个目的涉及到了最大化一个映射到游戏表现上的效用，而后一个目的涉及到了在单纯的胜利之外的目标，例如参与度，可信度，平衡性以及有趣性。AI 作为一个参与者可以扮演游戏中存在的玩家角色或是非玩家角色。一个 AI 方法需要在进行游戏时需要考虑的游戏特性包括了玩家数量，游戏的随机程度，可用的观察程度的大小，动作空间与分支因子，以及时间粒度。除此之外，在我们设计一个算法来进行某个游戏时，我们也需要站在算法的角度进行考虑，例如状态表示，前行模型的存在与否，可用的训练时间，以及 AI 可以进行的游戏数量。

上面的角色与特性我们在这章的第一部分有着详细的阐述，因为无论应用何种 AI 方法，它们都是非常重要并且相关的。在设计到本章所涵盖的方法时，我们集中于用于进行游戏的树搜索，强化学习，监督学习以及混合方法。从某种意义上来说，我们将在第 2 章中列出的方法定制为了游戏过程中的任务。本章最后对基于游戏类型的研究以及相关方法进行了详细回顾与总结。特别是，我们阐述了 AI 如何进行棋盘游戏，卡牌游戏，街机游戏，策略游戏，竞速游戏，射击游戏还有严肃游戏，以及互动小说，与多种其他的游戏种类，例如休闲游戏与格斗游戏。