异步图书
www.epubit.com

游戏AI程序设计实战

王磊◎编著

- ★ 用实战的方式讲解 AI 在游戏开发中的应用
- ★ 讲解 AI 的应用场景，介绍寻路算法、有限状态机的实现方法、多层状态机的实现方法、行为树、机器学习算法
- ★ 通过综合案例介绍足球游戏中 AI 知识和技术的具体应用

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

AI游戏程序设计实战

王磊◎编著

人民邮电出版社
北京

图书在版编目（CIP）数据

游戏AI程序设计实战 / 王磊编著. -- 北京 : 人民邮电出版社, 2019.4
ISBN 978-7-115-50004-5

Ⅰ. ①游… Ⅱ. ①王… Ⅲ. ①游戏程序－程序设计
Ⅳ. ①TP317.6

中国版本图书馆CIP数据核字(2018)第248219号

内 容 提 要

本书以实战的方式讲解 AI 在游戏开发中的应用，书中主要内容包括：AI 的基本概念、游戏中常用的寻路算法、Unity 的基本知识、有限状态机、行为树、AI 插件 Behavior Designer、遗传算法、足球 AI 的实现、游戏 AI 设计的扩展技术等。本书适合游戏开发者、程序员阅读。

◆ 编　　著　王　磊
责任编辑　谢晓芳
责任印制　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：12.75
字数：303 千字　　2019 年 4 月第 1 版
印数：1－2 400 册　　2019 年 4 月北京第 1 次印刷

定价：59.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

前　言

本书从是什么、为什么、怎么做这 3 个方面介绍游戏 AI 程序的设计。本书旨在让广大游戏从业者能够快速了解什么是 AI，为什么 AI 在游戏中必不可少，以及如何通过程序实现这些复杂的 AI。具体内容如下。

前 3 章主要介绍游戏 AI 存在的意义和简单的寻路算法。第 4 章与第 5 章主要介绍游戏 AI 中常用的程序设计思路：状态机和行为树。为了便于读者理解，示例由浅入深。第 6 章介绍 Unity 游戏开发中常用的 AI 辅助插件。第 7 章介绍机器学习算法。第 8 章通过一个简单的游戏 AI 项目的完整实现，展示游戏 AI 的设计思路和实现技术。第 9 章介绍游戏 AI 开发中一些必备的知识点。最后一章介绍如何在游戏 AI 开发中提升自己。

本书没有详细地讲解理论知识，主要讨论游戏项目的 AI 实现技术。希望读者通过本书可以了解游戏的 AI 技术，并快速应用到实际项目中，或通过本书的启发，自己设计出想要的游戏 AI。

本书提到的 Unity 示例代码都已开源，读者可以在 GitHub 网站里面查找“onelei”用户，然后使用“Game-AI-Programming-Book”关键字搜索源代码。为便于读者查阅，示例代码在 GitHub 网站上面都是按照章号分类的。

在写作本书的过程中，作者得到许多朋友的帮助，在此表示感谢。同时，感谢人民邮电出版社编辑张涛的细心指导！

由于本人的知识水平有限，书中难免存在错误和遗漏，欢迎读者批评指正，邮箱地址为 ahleiwolong@163.com。

王　磊

资源与支持

本书由异步社区出品，社区（https://www.epubit.com/）为您提供相关资源和后续服务。

配套资源

本书配套资源包括书中示例的源代码。

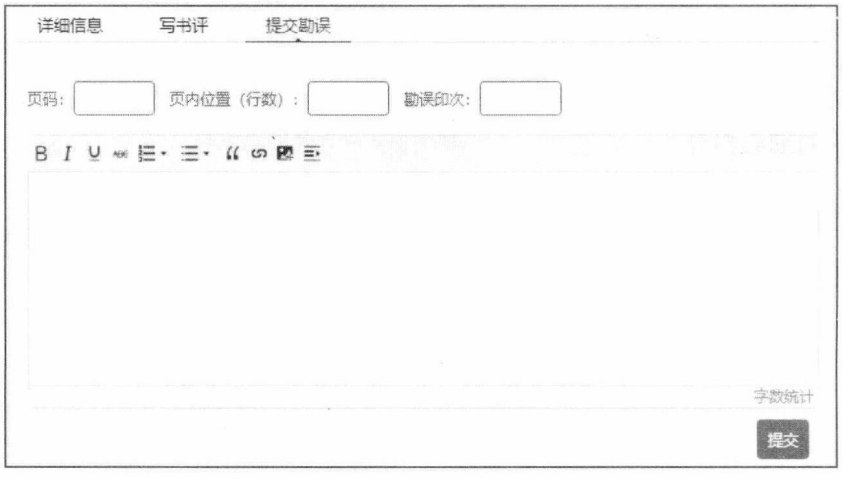
要获得以上配套资源，请在异步社区本书页面中单击 配套资源 ，跳转到下载界面，按提示进行操作即可。注意，为保证购书读者的权益，该操作会给出相关提示，要求输入提取码进行验证。

如果您是教师，希望获得教学配套资源，请在社区本书页面中直接联系本书的责任编辑。

提交勘误

作者和编辑尽最大努力来确保书中内容的准确性，但难免会存在疏漏。欢迎您将发现的问题反馈给我们，帮助我们提升图书的质量。

当您发现错误时，请登录异步社区，按书名搜索，进入本书页面，单击“提交勘误”，输入勘误信息，单击“提交”按钮即可。本书的作者和编辑会对您提交的勘误进行审核，确认并接受后，您将获赠异步社区的 100 积分。积分可用于在异步社区兑换优惠券、样书或奖品。



扫码关注本书

扫描下方二维码，您将会在异步社区微信服务号中看到本书信息及相关的服务提示。

与我们联系

我们的联系邮箱是 contact@epubit.com.cn。

如果您对本书有任何疑问或建议，请您发邮件给我们，并请在邮件标题中注明本书书名，以便我们更高效地做出反馈。

如果您有兴趣出版图书、录制教学视频，或者参与图书翻译、技术审校等工作，可以发邮件给我们；有意出版图书的作者也可以到异步社区在线提交投稿（直接访问 www.epubit.com/selfpublish/submission 即可）。

如果您是学校、培训机构或企业，想批量购买本书或异步社区出版的其他图书，也可以发邮件给我们。

如果您在网上发现有针对异步社区出品图书的各种形式的盗版行为，包括对图书全部或部分内容的非授权传播，请您将怀疑有侵权行为的链接发邮件给我们。您的这一举动是对作者权益的保护，也是我们持续为您提供有价值的内容的动力之源。

关于异步社区和异步图书

“**异步社区**”是人民邮电出版社旗下 IT 专业图书社区，致力于出版精品 IT 技术图书和相关学习产品，为作译者提供优质出版服务。异步社区创办于 2015 年 8 月，提供大量精品 IT 技术图书和电子书，以及高品质技术文章和视频课程。更多详情请访问异步社区官网 https://www.epubit.com。

“**异步图书**”是由异步社区编辑团队策划出版的精品 IT 专业图书的品牌，依托于人民邮电出版社近 30 年的计算机图书出版积累和专业编辑团队，相关图书在封面上印有异步图书的 LOGO。异步图书的出版领域包括软件开发、大数据、AI、测试、前端、网络技术等。

异步社区

微信服务号

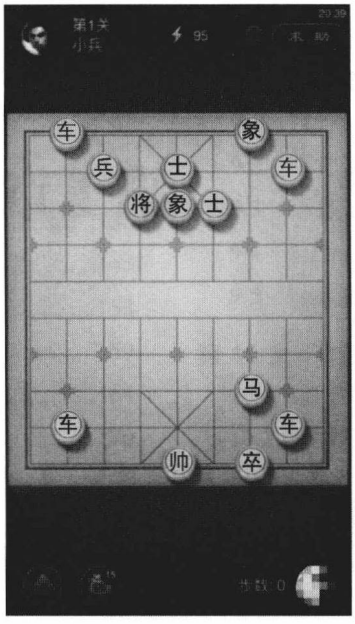
目　录

第1章　游戏AI概述

1.1 AI是什么

AI的英文全称为 Artificial Intelligence，中文翻译为人工智能。提及AI，最让人印象深刻的就是前一段时间热门的AlphaGo。它是由谷歌（Google）旗下DeepMind公司戴密斯·哈萨比斯领衔的团队开发的一个围棋算法。在AlphaGo战胜围棋世界排名第一的柯洁之后，一时间“人工智能是否将取代人类”的新闻报道铺天盖地。其实AI并不像人们想象的那样深不可测，毕竟是人类开发出了这个程序。在有些方面，人类比不过AI是很正常的事。就好比让你和火车赛跑，你肯定跑不过火车。计算一个很长的数学公式，人类可能要花几分钟，但是计算机可以瞬间得出结果。AI说到底其实就是一个程序。像我们平时玩的QQ象棋里面的闯关模式就包含了AI，如图1-1所示。

中国象棋的AI其实很复杂，因为每一个棋子的每次移动都需要考虑自身是否安全，需要判断对手是否会在后面几步就战胜自己，同时也需要设计思路战胜对手。



▲图 1-1　QQ 象棋的闯关模式截图

1.2 为什么在游戏里使用 AI

在实际应用中，AI 可以大大简化人类的工作，将人类从繁重的体力劳动中解脱出来，从而进行更高层次的工作。例如，自动收捡快递的机器人中就有 AI 的应用。在游戏中，我们为什么需要 AI 呢？在我们玩游戏的时候，经常会遇到很多“小怪”，它们在场景中走来走去，一旦敌人靠近，它们就会主动出击，这其实就是 AI。通过提高游戏中非玩家控制角色（Non-Player Character）的智能水平，让它们看起来很“聪明”，这样玩家才会在玩游戏的过程中觉得有趣和富有成就感。“看吧，它们很聪明，但是我还是把它们打败了。”有了这样的心理暗示后，玩家就会获得成就感和满足感，这就是游戏的乐趣。

几乎每一款游戏都应用 AI，只不过有的 AI 很简单，以至于用户不会注意到 AI 的存在。例如，《纪念碑谷》游戏里面的主人公艾达在移动的过程中，如果靠近乌鸦，乌鸦就会叫，这个就是简单的 AI。还有近期很热门的《王者荣耀》游戏，就应用了相对复杂一些的 AI：当敌方的“超级兵”在移动的时候，如果玩家控制的角色靠近，“超级兵”就会追过去；一旦玩家

控制的角色在“超级兵”的攻击范围内，“超级兵”就会执行攻击行为，如果玩家控制的角色逃走，“超级兵”还会在后面追击。这就是一个移动攻击 AI。但是，非玩家控制角色也不会一直追着玩家控制的角色跑，非玩家控制角色会优先攻击离自己最近的故人。

在游戏设计中，需要注意的是，AI 既不能设计得太“聪明”，也不能太“笨”。如果用户控制的角色在每次“打怪物”的时候，“怪物”都能够轻松躲避，最后众多“怪物”采取群体策略，把玩家控制的角色困在一个地方不断地对其进行攻击，使其毫无还手之力，这会增加玩家的挫败感，玩家肯定心情很糟，必定会让游戏玩家流失。反之，如果玩家控制的角色每次都可以轻松地躲避“怪物”的攻击，或者“怪物”总是傻傻地站着不动，让玩家控制的角色攻击，玩家就会觉得很无聊。因此，AI 的平衡性是在设计游戏 AI 的过程中需要注意的。

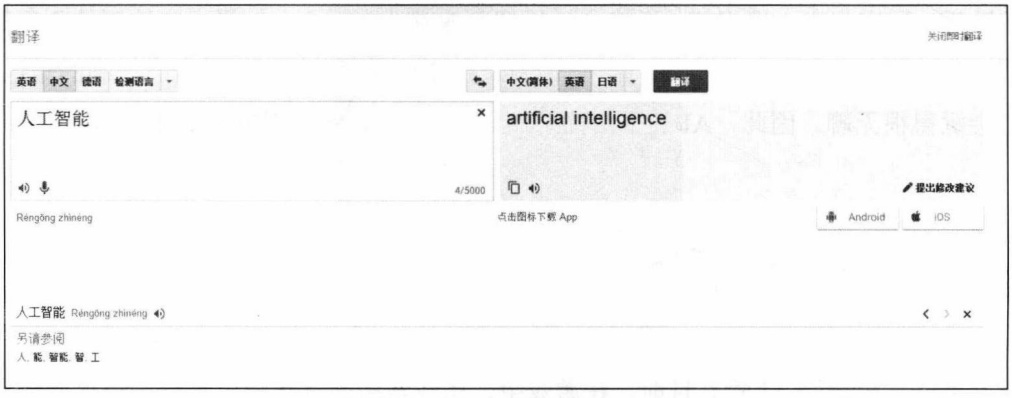
1.3 如何在游戏中实现 AI

那么如何实现游戏 AI 呢？目前，在游戏中，比较常用的有状态机和行为树这两个程序设计，而后面章节将要提到的遗传算法用得不是太多，这主要是考虑到手机游戏的局限性。因为手机性能相对计算机来说不是很高，而且复杂的 AI 里面包含了许多的运算，所以会大幅度降低手机的性能，从而导致游戏卡顿。同时，游戏最后的结果必须是可控的，不然会出现很多莫名其妙的问题，最终导致更多的错误（bug）。因为遗传算法和神经网络算法等机器学习算法在实现过程中都会模拟一些操作，同时包含了概率在里面，所以导致很多东西不可控。因此，在手机游戏中，AI 都是相对弱化的，很少有强策略的 AI。在程序实现中，比较简单的就是使用状态机策略，复杂点的大多是行为树。当然，像多层状态机也是可以进行复杂的 AI 设计的，这就要看个人对项目的把握了。具体在什么时候用哪种框架去实现，可能需要根据工作经验去抉择。

本书提到的算法的主要代码都是用 C#语言实现的，同时基于 Unity 引擎开发。当然，框架和算法都是不依赖引擎本身的，用其他的语言实现也是可以的。本书介绍的这些算法仅仅提供一种设计思想。

1.4 AI 就在我们身边

这里举个应用 AI 的简单例子。我们常用的谷歌翻译就是一个使用人工智能的例子，如图 1-2 所示。



▲图 1-2　谷歌翻译

在输入框中输入想要翻译的内容，就可以知道它对应的其他语种的翻译结果。但很多时候 AI 并不能够准确地将输入的内容翻译成我们想要的结果。中国很多的古诗词使用谷歌翻译都无法得到满意的结果。例如，“春风又绿江南岸”诗句的翻译结果如图 1-3 所示。

▲图 1-3　谷歌翻译中国古诗词

对于中国的古诗词，英语的翻译结果只能够体现其字面的含义，而对更深层次的含义的翻译还要依靠人类。当然，我们平时用谷歌翻译简单的单词都是没什么问题的，而对于整句话的翻译还是要自己斟酌一下。现在看来，AI 其实离我们并不是很遥远，它就在我们身边。接下来，本书将重点介绍几个常见的游戏中 AI 的示例，同时针对不同的算法提供不同的实现，而且会将各个方法进行比对，以供广大的游戏从业者参考。

第2章　游戏中的寻路算法

2.1 寻路算法

游戏中有一个常见的情景是在地图上单击某个位置，然后玩家控制的角色就自动移动过去。如果旁边有障碍物，角色还会从旁边绕开，看起来非常智能。“围住神经猫”就需要通过寻路算法来求出最短路径，这里面就有游戏中常用的寻路算法——A*算法。A*算法是在图的基础上的搜索算法，它是一种启发式的搜索算法。在介绍 A*算法之前，我们先简单了解图的两个常用算法——广度优先搜索（Breadth First Search，BFS）算法和深度优先搜索（Depth First Search，DFS）算法。

2.2 为什么需要图

在设计游戏程序的时候，需要用变量来表示某个数据。例如，有 1000 个金币，在代码中的表示方式如下。

```
Int PlayerCoin = 1000;
```

上述代码定义了一个 int 类型的数据，用它来存取用户的金币信息。但是，随着需求的不断变动，可能发现这个 int 类型不够用了。例如，需要一个背包系统，里面有玩家获得的所有道具信息。背包里面至少需要包含道具 ID 和道具数量这两个信息，因此，又引入了类，示例代码如下。

```
Public class Item
{
    Public int ID;          //道具 ID
    Public int Num;         //道具的数量
}
```

如果还需要根据道具类型排序，就要接着扩充这个 Item 类。

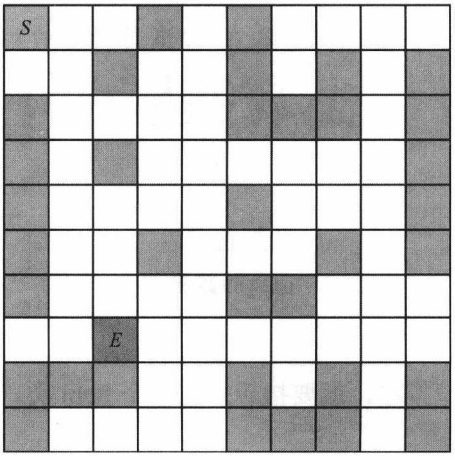
```
Public class Item
{
    Public int ID;          //道具 ID
    Public int Num;         //道具的数量
    Public EItemType Type;  //道具类型
    Public …
    …
}

Public Enum EItemType
{
NONE,
ATTACK,     //进攻型道具
DEFENCE     //防守型道具
}
```

下面就可以根据需要不断地增加变量来丰富这个 Item 类（背包数据）。那这里是不是就可以将这个方法用在所有地方了呢？我们来分析下面这样一个需求。

玩家控制的角色需要从地图的 S 点走到 E 点，具体地图如图 2-1 所示。

该案例中，需要计算出从起点 S 到终点 E 的最短路径，然后玩家控制的角色按照方格一步步移动过去。这里使用之前的数据结构就无法表达出两个地图块之间的数据关系，而图这个数据结构正好满足这种关系。



▲图 2-1　地图俯视图

2.3 地图的简化

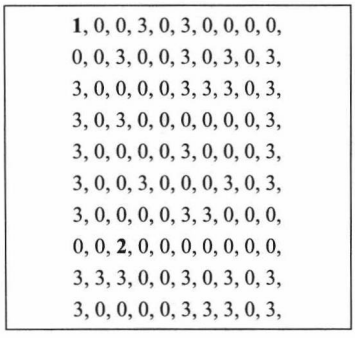
如果一整张大地图呈现在我们面前，那么要如何操作这个地图来让玩家控制的角色自动寻路呢？就好比一个大蛋糕摆在我们面前，我们就直接张口吃吗？吃蛋糕的基础流程大致为首先将蛋糕切分成很多小块，然后一小块一小块地吃。其实，这里面就包含了我们对地图简化的思想。首先，将地图切分成一个个小块。然后，在能够行走的地方放个“灯笼”，在不能够走的地方不放“灯笼”。最后，我们就在这些有“灯笼”的地方找出一条路径。

现在思路很清晰了，需要将实际的地图转换为对应的地图数据来进行操作。按照图 2-1 给出的地图信息，将对应的信息使用数字来代替。其中有“灯笼”的地方（也就是能够行走的地图块）就是 0；黑漆漆的地方（也就是不能够行走的地图块）就是 3。1 和 2 分别代表的是起点和终点，这里对 1 和 2 进行了加粗显示，如图 2-2 所示。

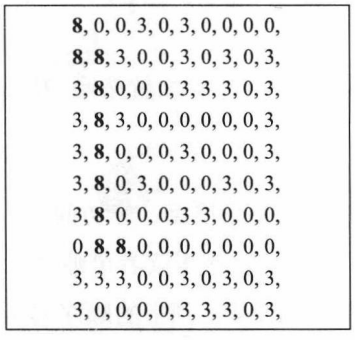
此时，寻路的问题就变成了在这个地图中找出 1→2 的最短路径。通过观察，可得出图 2-3 所示最短路径（其中加粗的 8 表示从起点到终点所经过的路径）。

这里只是演示了一个简单的地图，如果地图设计得比较复杂，那么找起来可能会比较费劲。

也就是说，无论地图复杂与否，最好能通过一个算法在一个给定的地图数据中找出一条路径。此时，就需要使用图的两个遍历算法。按照访问顶点的顺序可以分为 BFS 算法和 DFS 算法。下面开始介绍如何通过 BFS 算法和 DFS 算法来找出路径。



▲图 2-2　用数字表示的地图



▲图 2-3　用数字 8 表示的最短路径

2.4 BFS 算法

BFS 算法的思想是从图中一个顶点 V 出发，然后标记为访问，然后依次访问和 V 相邻的顶点，接着从这些邻接顶点出发，访问它们的邻接顶点，直到所有的邻接顶点都被访问到。如果图中还有未被访问的顶点，那么再选择一个未被访问的顶点，重复以上步骤，直到所有的顶点都被访问到。