HZ BOOKS 华章IT
[PACKT] PUBLISHING
游戏开发与设计技术丛书

Unity 3D 人工智能编程

[美] 昂·斯尤·基奥 (Aung Sithu Kyaw) 克利福德·彼得斯 (Clifford Peters) 斯特·奈·斯瑞 (Thet Naing Swe) 著 李秉义 译

Unity 4.x Game AI Programming

- 如何使游戏中的角色看上去像真实的人或动物，如何让游戏更惊险、刺激、有趣味性，使人玩不释手
- 本书通过大量的示例项目，详细展示如何将人工智能技术应用到Unity 3D游戏，系统讲解如何使游戏角色更具智能性，增强游戏的可玩性

机械工业出版社
China Machine Press

游戏开发与设计
—技术丛书—

Unity 3D
人工智能编程

[美] 昂·斯尤·基奥 (Aung Sithu Kyaw) 克利福德·彼得斯 (Clifford Peters) 斯特·奈·斯瑞 (Thet Naing Swe) 著 李秉义 译

Unity 4.x Game AI Programming

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Unity 3D 人工智能编程 /（美）基奥（Kyaw, A. S.），（美）彼得斯（Peters, C.），（美）斯瑞（Swe, T. N.）著；李秉义译 . —北京：机械工业出版社，2015.6
（游戏开发与设计技术丛书）
书名原文：Unity 4.x Game AI Programming

ISBN 978-7-111-50389-7

I. U… II. ① 基… ② 彼… ③ 斯… ④ 李… III. 游戏程序 – 程序设计 IV. TP311.5

中国版本图书馆 CIP 数据核字（2015）第 115325 号

本书版权登记号：图字：01-2015-1909

Unity 3D 人工智能编程

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：陈佳媛 责任校对：董纪丽
印 刷：北京市荣盛彩色印刷有限公司 版 次：2015 年 6 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：13
书 号：ISBN 978-7-111-50389-7 定 价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

华章IT
HZBOOKS | Information Technology

Preface 前 言

本书旨在帮助你把各种人工智能技术应用到你的游戏中。我们将会讨论决策技术，比如有限状态机和行为树；也将探讨运动、避开障碍和群组行为；还将演示如何跟随一条路径，如何使用 A* 寻路算法来创建一条路径，以及如何使用导航网格到达目的地。作为额外收获，你将详细了解随机性和概率，并把这些概念应用到最后一个综合项目中。

本书内容

第 1 章讨论什么是人工智能，如何将其应用到游戏中，以及游戏中使用的各种实现人工智能的技术。

第 2 章讨论人工智能中需要用到的一种简化决策管理的方法。我们使用有限状态机来确定人工智能在特定状态下的行为，以及这种状态下人工智能如何转换为其他状态。

第 3 章讨论概率论的基础知识，以及如何改变特定输出的概率。然后学习如何给游戏增加随机性，让游戏中的人工智能更难以预测。

第 4 章介绍怎样让游戏角色在某些情况下能够感知他们周围的世界。当他们具有视觉和听觉时，游戏角色会知道敌人就在附近，他们还会知道何时发起攻击。

第 5 章讨论多个对象组队同时行进的情况。该章将探讨两种实现群组行为的方式，以及这两种方式是怎样使这些对象同时行进的。

第 6 章学习人工智能角色如何跟随一条给定的路径到达目的地。我们将了解人工智能角色如何在不知道路径的情况下找到目标，以及如何使其移向目标的同时避开

障碍。

第 7 章讨论一个流行的算法，即寻找从指定位置到目标位置的最优路径。有了 A* 算法，我们可以扫描地形并找到到达目标的最优路径。

第 8 章讨论如何利用 Unity 的能力使寻路更易于实现。通过创建一个导航网格（需要使用 Unity Pro 版），我们能够更好地表示周围的场景，然后就能使用图块和 A* 算法。

第 9 章讲解从有限状态机扩展而来的行为树，即使在最为复杂的游戏中我们也可以使用它。我们将使用免费插件 Behave 来帮助在 Unity 中创建并管理行为树。

第 10 章把我们在本书中所学的各种原理整合在最后一个项目中。在这里你能够应用所学的人工智能原理，设计出一个令人难忘的车辆战斗游戏。

本书要求配置

学习本书，要求读者安装 Unity 3.5 或更高版本。第 8 章讨论导航网格，顾名思义涉及创建一个导航网格，这需要你安装 Unity Pro 版本；第 9 章讨论行为树，要求下载 Behave——一个免费的行为树插件，这需要你拥有一个 Unity Store 账号。不过这些需求都是可选的，因为本书配备的资源中已经为你准备好了导航网格和 Behave 插件，可登录华章网站下载，网址为 www.hzbook.com。

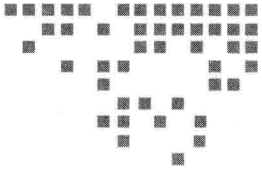
本书的读者对象

本书面向任何想要学习将人工智能应用到游戏中的读者，并侧重于之前有 Unity 使用经验的读者。我们会用 C# 语言编写代码，所以我们希望你熟悉 C#。

下载示例代码和书中的彩色插图

你可以在华章网站的本书页面中下载示例代码文件和书中的彩色插图。

Contents 目 录



第 1 章 Chapter 1

人工智能导论

本章将会在学术领域、传统领域以及游戏的具体应用上给你提供一些人工智能的背景知识。我们将会学习人工智能在游戏中的实现和应用与其他领域中的人工智能的不同，以及游戏人工智能的一些重要且特殊的需求，还将探索在游戏中应用人工智能的基本技术。本章也将作为后面章节的参考。在后面的章节中，我们将会在 Unity 中实现这些人工智能技术。

1.1 人工智能

一些类似于人类和其他动物的生命体具有某种智能，这种智能有助于我们在完成一件事时做出特定的选择。然而计算机只是台可以接收数据的电子设备，它以很高的速度执行逻辑和数学运算并输出结果。所以人工智能（AI）的主旨本质上是让计算机能够像生物体一样，具有思考和决定的能力来执行某些特定操作。

显而易见，人工智能是一个巨大的课题。而这样一本小书并没有办法涵盖所有与人工智能有关的内容。但是了解人工智能在不同领域中的基础知识是非常重要的。人工智能只是一个总称，对于不同的目的，它的实现和应用是不同的 ，人工智能可以用

来解决不同的问题。

在开始研究游戏的专用技术之前，我们先来看看人工智能在下面这些研究领域中的应用：

- **计算机视觉**：这是一种从视觉输入源（比如视频和摄像机）获取信息并对它们进行分析，以执行特定操作（比如脸部识别、对象识别、光学字符识别）的能力。
- **自然语言处理（NLP）**：这是一种让机器能够像我们平常那样阅读和理解语言的能力。问题是，我们今天使用的语言对于机器来说是难以理解的。表达同一件事情有很多种不同的说法，同一个句子依据不同的上下文也有不同的含义。自然语言处理对于许多机器来说是非常重要的一个步骤，因为它们需要了解我们使用的语言和表达方式。幸运的是，在网络上有大量可以获取到的数据集合，可以用来帮助研究人员对语言进行自动分析。
- **常识推理**：在那些我们并不完全了解的领域中，我们的大脑可以用常识推理来很容易地得出问题的答案。常识性知识是我们用来尝试理解某些问题的一个常用和普遍的方式，因为我们的大脑可以混合上下文、背景知识和语言能力，让它们综合影响、相互作用。但是让机器来应用这些知识是件非常复杂的事，对于研究人员来说这仍是一个重大的挑战。

1.2 游戏中的人工智能

游戏人工智能需要去完善一个游戏的品质。为此，我们需要了解每个游戏必须满足的基本需求。答案应该是显而易见的，就是让游戏好玩。那么，是什么决定了一个游戏是否好玩呢？这其实是游戏设计的主旨（Jesse Schell 所著的《The Art of Game Design》是一份极佳的参考资料），让我们试着在不深入讨论游戏设计的话题的情况下来解决这个问题。你会发现一个具有挑战性的游戏一定是好玩的。重申一遍：让游戏具有挑战性。这意味着一个游戏不应该太过困难让玩家没有击败对手的可能性，也不应该让玩家轻而易举地取得胜利。让游戏好玩的关键因素是为之找到合适的难度等级。

而这正是人工智能发挥作用的地方。人工智能在游戏中的作用是通过提供富有挑战性的竞争对象来让游戏更好玩，而在游戏世界中行动逼真的有趣的非玩家角色（NPC），也会让游戏更好玩。所以，我们的目的不是复制人类或其他动物的整个思维过程，而是通过让这些 NPC 对游戏世界里不断变化的情形，产生对玩家来说足够合理、有意义的反应，来让它们看起来更加智能。

我们不希望游戏中的人工智能系统花费过多的计算代价，因为人工智能计算所需要的处理器能力，比如图形渲染和物理学仿真，要同其他的操作共享。另外，别忘了它们都是实时发生的，并且，在整个游戏中保持稳定的帧率也非常重要。甚至有人试图制造专门用于人工智能运算的处理器（AI Seek 公司的 Intia 处理器）。随着处理器的处理能力与日俱增，我们现在拥有了越来越多的人工智能计算的空间。然而，像所有其他的游戏开发规范一样，优化人工智能计算仍然是人工智能开发者所面临的巨大的挑战。

1.3 人工智能技术

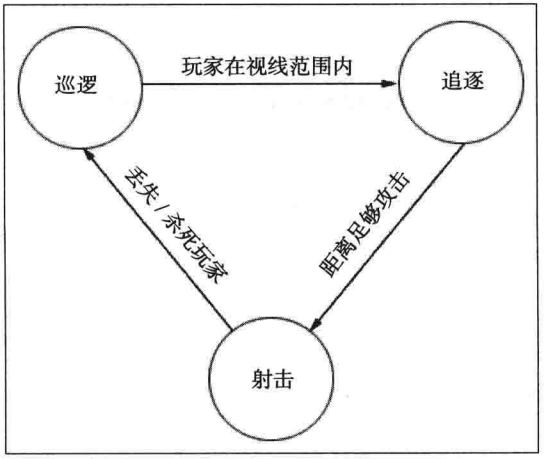
在本节中，我们将简单了解部分人工智能技术在不同类型的游戏中的应用。在后面的章节中，我们将学习如何在 Unity 中实现这些功能。由于这本书不是专注于人工智能技术本身，而是这些技术的在 Unity 中的应用，所以在这里我们不会深究过多的细节。就让我们把它当作一个速成班，然后再开始研究这些应用。如果你想了解关于游戏人工智能的更多内容，也有一些非常棒的书值得推荐，如由 Mat Buckland 所著的《Programming Game AI by Example》、由 Ian Millington 和 John Funge 合著的《Artificial Intelligence for Games》。《AI Game Programming Wisdom》系列丛书也蕴藏了大量最新人工智能技术的实用资源和文章。

1.3.1 有限状态机

有限状态机（FSM）可以认为是最简单的人工智能模型之一，并且普遍应用于大部分游戏中。一个状态机主要由一组数量有限的状态组成，这些状态之间可以相互转换，

并以图表的形式展示出它们之间的联系。一个游戏实体以初始状态开始运行，然后就开始不断地查找能够导致其转换到另一个状态的事件和规则。一个游戏实体在任一给定时间内只能出现在一个确定的状态中。

举个例子，让我们来看一个典型的射击游戏中的人工智能守卫。它的状态可以非常简单，如巡逻（Patrol）、追逐（Chase）和射击（Shoot）。



一个简单的人工智能守卫有限状态机

一个简单的有限状态机主要由四部分组成：

- 状态：该组件定义了一组状态，一个游戏实体或 NPC 可以选择（巡逻、追逐和射击）
- 转移：该组件定义了不同状态之间的关系
- 规则：该组件用来触发状态转移（玩家在视线范围内、距离足够攻击、丢失 / 杀死玩家）
- 事件：该组件用于触发检查规则（守卫的可见区域、与玩家的距离等）

所以，《雷神之锤 2》里的怪物可能具有以下状态：站立、行走、跑步、躲避、攻击、空闲和搜索。

有限状态机广泛地应用于游戏人工智能中，因为它们易于实现，不管对于简单还是相对复杂的游戏都游刃有余。通过使用简单的 if/ else 语句或 switch 语句，我们就能轻松地实现有限状态机。在我们设计出更多的状态和转换时，它看上去可能会有点混乱。在下一章内容中，我们将学习如何使用一个简单的有限状态机。

1.3.2 人工智能中的随机性和概率

想象一下，一个敌方机器人在第一人称射击游戏中（FPS）可以随时暴头杀死玩家，在一个赛车游戏中，对手总是能选择最佳路线，在超车时从不会撞到任何障碍物，具有如此智能水平的对手会让游戏非常困难，玩家几乎不可能取胜。或者说，想象一下敌方人工智能总是选择同样的路线，或总是见到玩家就逃跑……如果人工智能控制的实体每次遇见玩家都表现出同样的方式，就会让玩家轻易地预测游戏并取得胜利。

以上两种情况显然会降低游戏的乐趣，并让玩家感到游戏不具备挑战性，或不够公平。解决这种完美或愚蠢的人工智能的一种方法是，在他们的智能系统中引入一些错误。在游戏中，将随机性和概率应用于人工智能的计算过程中。以下是一些我们想让我们的人工智能实体做出一个随机决定的情况：

- 非故意：这种情况是游戏代理（或者也可能是 NPC）需要随机地做出一个决定，因为它不具备足够的信息来做出完美的决定，以及 / 或者无论做出什么样的决定其实都无关紧要。在这种情况下的策略就只是简单地随机做出决定，并希望最好的结果发生。
- 故意：这种情况是完美的人工智能和愚蠢的人工智能。正如我们在前面的例子中所讨论的，我们需要人为地添加一些随机性，好让它们更逼真，并向玩家匹配适合的难度级别。这种随机性和概率可能被用于命中率、在基础伤害上加减的随机伤害。使用随机性和概率，我们可以给游戏添加现实的不确定感，让我们的人工智能系统难以预测。

我们也可以用概率来定义不同类型的人工智能角色。让我们看看远古捍卫者（DotA）中的英雄人物。DotA 是一个受欢迎的《魔兽争霸 III》中的即时战略（RTS）

游戏模式。游戏中有三类基于三种主要属性的英雄，这三种主要属性分别为：力量、智力和敏捷。力量是英雄的物理力量的度量，智力与英雄控制法术的能力有关，敏捷则决定了英雄躲避攻击和迅速进攻的能力。力量英雄的人工智能将具备在近战过程中造成更多伤害的能力，而一个智力英雄将有更多的概率成功使用法术造成更高伤害。仔细平衡不同类型的英雄之间的随机性和概率，能够让这个游戏更具挑战性，并让DotA 玩起来更加有趣。

1.3.3 感应器系统

我们的人工智能角色需要了解其周围的环境，以及与它们互相影响的游戏世界，以便做出某个特殊的决定。这些信息可以是下面的这些。

- 玩家的位置：该信息用于决定是否进行攻击、追逐或继续巡逻。
- 建筑物和附近的对象：该信息用于躲藏或寻找掩护。
- 玩家的健康和自己的健康：这些信息用于决定是否撤退或继续前进。
- 在一个 RTS 游戏地图上的资源的位置：该信息用于占领和收集资源，建设和生产其他单位。

正如你所看到的那样，根据我们所建立的游戏的类型，我们对这些信息的需要可能有很大的差异。那么，我们如何收集这些信息呢?

1. 轮询

轮询就是一种收集这些信息的方法。我们可以用 if / else 或 switch 语句简单地在人工智能角色的 FixedUpdate 方法中进行检查。人工智能角色只在游戏世界中轮询它们感兴趣的，并进行检查，之后根据结果采取行动。如果没有太多事情需要检查，轮询是一个很好的方法。然而，有些角色可能并不需要轮询每一帧的世界状态，不同的角色可能需要不同的轮询率。所以通常情况下，在一些具备更复杂的人工智能系统的大型游戏中，我们需要部署一个由事件驱动的全局消息系统。