



资深游戏开发专家兼Lua开发工程师10余年工作经验结晶，Lua语言创始人亲自作序推荐，权威性毋庸置疑

内容全面而深入，既深入阐述了Lua语言的核心要素，又全方位地讲解了利用Lua进行游戏开发和设计的各种技术细节、方法和最佳实践

实战性强，不仅为各个知识点精心设计了大量辅助读者理解的小案例，而且包括完整的大案例，可操作性极强

华章程序员书库

Game Development with Lua

Lua游戏开发 实践指南

(美) Paul Schuytema Mark Manyen 著

田剑 译



CD-ROM



机械工业出版社
China Machine Press

华章程序员书库

Lua 游戏开发实践指南

Game Development with Lua

(美) Paul Schuytema Mark Manyen 著

田剑 译



机械工业出版社
China Machine Press

本书是资深 Lua 游戏开发工程师 10 余年工作经验和智慧的结晶，Lua 语言创始人亲自作序推荐，是 Lua 游戏开发领域最具实战意义和代表性的著作之一。它不仅详细讲解了在游戏开发中使用 Lua 的各种技术细节、方法技巧和最佳实践，而且讲解了如何使用 Lua 作为主要工具将游戏设计转化为代码实现的过程。此外，它还重点阐述了 Lua 语言的核心要素。最重要的是，本书包含大量精心设计的案例，并附赠了完整的源代码，可操作性极强。

全书一共 15 章：第 1~3 章简单地介绍了 Lua 语言的特性、授权，以及在游戏开发中的强大用途；第 4~5 章详细讲解了 Lua 语言的基本语法和核心要素；第 6~7 章讲解了 Lua 与 C/C++ 程序的整合以及与 C++ 的交互相关的技术细节；第 8~9 章介绍了开发前需要做的准备工作，以及如何设计 Lua 版本的实现；第 10 章讲解了如何使用 Lua 来处理游戏数据；第 11 章讲解了 Lua 驱动的 GUI；第 12 章详细讲解了两个完整的游戏开发案例；第 13 章结合实例讲解了如何使用 Lua 定义和控制 AI；第 14 章展示了 Lua 在图形绘制和图像处理方面的强大功能；第 15 章探讨了 Lua 与多媒体、Lua 脚本的调试、Lua 应用的资源管理以及 Lua 代码的发布等内容。

Paul Schuytema and Mark Manyen: Game Development with Lua (ISBN 978-1-58450-404-7).

Copyright © 2005 by CHARLES RIVER MEDIA, INC., a part of Cengage Learning.

Original edition published by Cengage Learning. All Rights reserved.

China Machine Press is authorized by Cengage Learning to publish and distribute exclusively this simplified Chinese edition. This edition is authorized for sale in the People's Republic of China only (excluding Hong Kong, Macao SAR and Taiwan). Unauthorized export of this edition is a violation of the Copyright Act. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Cengage Learning Asia Pte. Ltd.

5 Shenton Way, # 01-01 UIC Building, Singapore 068808

本书原版由圣智学习出版公司出版。版权所有，盗印必究。

本书中文简体字翻译版由圣智学习出版公司授权机械工业出版社独家出版发行。此版本仅限在中华人民共和国境内（不包括中国香港、澳门特别行政区及中国台湾）销售。未经授权的本书出口将被视为违反版权法的行为。未经出版者预先书面许可，不得以任何方式复制或发行本书的任何部分。

本书封面贴有 Cengage Learning 防伪标签，无标签者不得销售。

封底无防伪标均为盗版

版权所有，侵权必究

本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2012-7564

图书在版编目 (CIP) 数据

Lua 游戏开发实践指南/(美) 斯库特玛 (Schuytema, P.), (美) 马尼恩 (Manyen, M.) 著；田剑译. —北京：机械工业出版社，2013.1

(华章程序员书库)

书名原文：Game Development with Lua

ISBN 978-7-111-40335-7

I. L… II. ①斯… ②马… ③田… III. 游戏程序－程序设计－指南 IV. TP311.5

中国版本图书馆 CIP 数据核字 (2012) 第 265407 号

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：马 超

印刷

2013 年 1 月第 1 版第 1 次印刷

186mm×240mm · 16.25 印张

标准书号：ISBN 978-7-111-40335-7

ISBN 978-7-89433-699-6 (光盘)

定价：59.00 元 (附光盘)

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) 88378991；88361066

购书热线：(010) 68326294；88379649；68995259

投稿热线：(010) 88379604

读者信箱：hzjsj@hzbook.com

译 者 序

在程序开发领域里，各种编程语言层出不穷，在不同的场合选择合适的工具才能事半功倍。对于现在的游戏开发来说，已经不是那种小作坊的工作模式，也不是从零开始一步步打造自己作品的时代了。目前市场上的大型游戏往往需要数十人甚至上百人的团队至少经过数月甚至几年的艰苦开发才能完成。这样的开发规模没有合理的团队分工协作、科学的工程管理和强有力的开发工具是难以成功的。

游戏开发是一个创造性的工作，需要通过快速开发原型、测试和修改来验证游戏性。因此，需要一个具有良好兼容性、简单而高效的编程语言来帮助游戏设计师完成他们的工作。本书介绍的 Lua 就是这样一种语言，它借助 C/C++ 等底层语言可以无限扩展，而脚本语言的特性又让它十分适合快速原型开发和迭代。近年来，许多大型游戏都采用了 Lua 作为自己的嵌入式脚本语言，以此来实现可配置性和可扩展性。

本书的作者 Paul Schuytema 和 Mark Manyen 作为从事游戏行业多年的技术专家，由浅入深、循序渐进地为大家展示了如何使用 Lua 开发令人激动的游戏。本书从最简单的 Hello World 到复杂的人工智能和路径搜索，使用了大量的例子为初学者详细解释了 Lua 语言的方方面面，并带领大家从游戏设计开始逐步实现游戏的快速原型，展示了完整的游戏开发流程。

本书的翻译经历了三个多月时间，非常感谢华章公司的编辑们在翻译过程中的支持和帮助。由于译者的水平和时间有限，错误和不当之处在所难免，敬请广大读者批评指正。

序

脚本的概念在程序中十分重要，在游戏开发领域，它更是决定性的。脚本语言让程序员可以区分游戏开发的“硬核部分”和“软体部分”。“硬核部分”一般对计算性能要求很高，在开发过程中变更较少、重用性很高。图形引擎和人工智能模块是其中的代表。这些模块最适合使用 C 或者 C ++ 这样的语言开发，可以提供更好的性能。“软体部分”控制“硬核部分”来创建最后的图形和大量的物体。这个部分更适合使用 Lua 这样的脚本语言开发，可以为程序员在尝试、测试和改变游戏代码上提供更多的灵活性。

Lua 作为脚本语言的诞生是因为两个特别的行业需求，它们都和游戏相关。应用程序需要灵活的数据描述语言和简单的行为描述脚本语言（例如，数据验证），这些就是 Lua 最初的目标：可移植性、小巧、无限制。它必须是可移植的，因为目标客户的计算机是多样化的（MS-DOS、Windows 3.0、IBM AIX 及许多其他平台）。它还必须小巧，不能让程序因为使用它而变得庞大，因为一些目标机器空间是很有限的。同时，以往的经验告诉我们，编程语言不能限制程序员，因为在开发程序时，往往会出现我们意想不到的需求和用法。因此，从第一个版本开始，Lua 就因为它自身的简单性而表现出了优异的性能。后来，因为这种简单性被证明是十分有价值的，所以我们将它加到了 Lua 的目标中。

Lua 在 Tecgraf 一举成功之后，用在了许多其他项目中，因此我们向全世界开放了它。这是个成功的决定，Lua 的免费公开，让我们获得了国际化的顾问小组，语言本身也从中获得了长足的发展。

1996 年年末，卢卡斯艺术公司的 Bret Mogilefsky 在 Dobb 博士的网站上读到一篇关于 Lua 的论文后，决定在他开发的游戏中采用它。几个月后，他在游戏开发大会中公开了 Lua 相关的信息。不久以后，许多游戏公司就开始使用 Lua 了。Lua 使用率的快速增加令我们十分惊讶，因为我们从来没想到它能成为游戏开发的一种语言。它曾经在很多图形相关的项目中使用过，但是从未染指游戏领域。不过现在看来，Lua 进入游戏开发领域也是合情合理的。

Lua 所有的优点对于游戏开发都是很重要的：简单性、可移植性以及运行效率高，从

此以后，我们开始关注游戏开发者，陆续为 Lua 增加了许多新的特性，如为游戏开发提供了协同例程（coroutine）的功能。尽管 Lua 的功能不会局限在游戏开发上，但是在 Lua 未来的发展中，必着重考虑游戏开发。

非常欢迎本书成为 Lua 系列书籍的新成员。本书重点关注游戏开发，强烈推荐给有志于开发专业游戏的读者。我们期待本书能在这个不断发展的领域中更好地传播 Lua 语言。

——Lua 语言创始人之一 Roberto Ierusalimschy



前　　言

Lua 游戏开发

游戏开发是一个激动人心的过程，创造出让玩家花费数小时并乐在其中的游戏，给人带来的成就感是任何事情都无法比拟的。然而，这个创造的过程正在变得越来越难。那种奋战几个晚上或者几周就能单枪匹马设计出热门游戏的日子已经一去不复返，现在的游戏往往需要数十人的开发团队工作很多个月甚至几年才能完成。就算是那些可以从网上下载的最简单的“休闲游戏”也通常是由专业开发者组成的团队开发数月的成果。

尽管游戏开发的规模不断增大，但始终有一个不变的追求——测试、更新、调整，以及快速验证游戏性的能力，通常这个部分是设计和开发过程的核心。采用一种像 Lua 这样的脚本语言以及内核级别的语言（如 C++）可以帮助用户开发专业的游戏，并且还能让开发者和设计师快速实现设计想法、测试游戏功能。

适用读者



本书适合三类读者：

游戏程序员。程序员在开发团队中负责实现 Lua 和 C++ 之间的接口，并且通常还要编写部分或者全部的游戏脚本。本书将告诉程序员如何将 Lua 和 LuaGlue 的功能集成到游戏开发项目中。对于程序员来说，最重要的是，使用 Lua 可以在游戏开发的过程中节省很多时间和精力，因为许多游戏的功能可以由设计师和脚本程序员来实现。

游戏设计师。通常，游戏设计师会采用像 Lua 这样的脚本语言在运行环境中来实现部分设计。本书可以作为 Lua 语言的初级读本，为设计师打下坚实的技能基础，从而去构建真实的游戏世界。同时，本书还可以激发设计师的灵感，使用 Lua 开发可以帮助他们使用工具快速开发原型，快速实现并且进行创意的验证。

业余游戏开发者。学习如何开发你自己的游戏是富有成就感和具有挑战性的。游戏行业鼓舞了许多这样的业余爱好者，通过自己的项目来学习更多关于游戏和开发的知识，这样的付出很值得。本书展示了经验丰富的业余游戏开发者如何在他们的项目中使用 Lua，还提供一个已有的框架以便入门并深入学习 Lua，进而快速开发没有任何 C++ 代码的游戏。

(提供完整的控制台和游戏测试环境)。

本书主要内容

在本书中，有一篇关于 Lua 的简介，包括历史背景和脚本编程两方面。此外，读者还将学会如何链接 Lua API 来扩展 C++ 功能。

建立了一定的知识基础后，本书将带领读者使用 Lua 脚本语言开发一个游戏的“快速原型”。这个游戏会为读者展示使用 C++ 功能和 Lua 脚本的方法，例如：

- 存储和载入游戏数据
- 创建模块化的、灵活的 GUI 系统
- 用 Lua 脚本管理游戏的实时事件
- 使用 Lua 定义和控制游戏的 AI (人工智能)

系统要求

- P450 或者更好的处理器
- Windows 2000/XP
- 32MB RAM
- 演示程序要求：
 - DirectX 9 (包含在 CD-ROM 中)
 - DX 兼容的 3D 视频加速器
- DirectX SDK：
 - 操作系统：Microsoft Windows (r) 98, Windows Millennium Edition (Windows Me)，或者 Windows 2000, Windows Server 2003, Windows XP
 - 约 65MB 可用硬盘空间 (安装完成后可以删除安装文件。剩下的 DirectX 文件约占 18MB 硬盘空间，如果安装了更早版本的 DirectX，可能使用空间会有所不同。DirectX 9.0 会覆盖之前的版本。)
- Lua：
 - 兼容大部分可以编译 C 语言的系统
- Ogg Vorbis：
 - Microsoft Windows 95, 98, Me, NT, 2000 或者 XP
 - 声卡
 - 处理器：Pentium 200MHz 以上

- 32MB RAM

■ **Zeus:**

- Microsoft Windows 95, 98, Me, NT, 2000 或者 XP
- 4MB RAM
- 4.5MB 可用硬盘空间

随书光盘内容

随书光盘中的内容包含下列文件夹。

C ++ Code: 包含《Take Away》游戏和本书中其他例子的 Visual Studio 项目和所有 C ++ 代码（使用 .NET 版本的 Visual Studio）。还有一个 VS6_C ++ Code 文件夹，包含了使用 Visual Studio 6 的代码。

Chapters: 包含子文件夹（以章节命名），提供本书中的所有 Lua 脚本和可执行程序。

Documents: 包含了《Take Away》游戏的设计文档和 Lua 脚本编程规范。

DX9.0c: 包含了 DirectX 9 SDK 和最新的运行时发布包。

Figures: 包含了本书中的所有图片，保存在各章节的文件夹下。

License: 包含了 Lua 和 Ogg Vorbis 的发布版本的许可证文档。

Lua: 包含了 Lua 控制台、Lua 手册和 Lua 5.0 的源代码。

OggVorbis: 包含了 Ogg Vorbis 音乐系统的源代码。

Take Away: 本书中《Take Away》游戏的完整版本（部分的版本在各自章节的文件夹下）。

Zeus: 包含了 Zeus 程序编辑器的共享版，是一个不错的 Lua 脚本编辑器。

注意事项

我们目前正在从旧的 Visual Studio 架构转换到 .NET 架构。本书中的大部分代码，包括 DirectX SDK，都能在 .NET 版本的 Visual Studio 下正常工作。

我们知道还有许多程序员在 Visual Studio 6 下工作，因此也提供了这个版本的项目文件。不过需要注意，随书光盘中附带的 DirectX SDK 不能在 Visual Studio 6 下正常工作，用户需要 2004 年夏天的 DirectX SDK，在下面的链接中可以找到：<http://www.microsoft.com/downloads/details.aspx?FamilyId=FD044A42-9912-42A3-9A93-D857199F888E&displaylang=en>。

致 谢

没有 Nick Carlson 提供的优秀脚本，这本书很难顺利完成，他和我们一起完成了书中的例子和大量脚本，他才刚刚开始大学生涯，未来一定前途无量。同样感谢 Chris Listello 为《Take Away》的美术设计和封面设计所做的工作。还要感谢 Roberto Ierusakimschy 为本书作序。感谢所有 Lua Tecgraf 小组的成员，为我们创造了这样一款功能强大且性能优越的脚本语言。最后感谢 Charles River 小组：感谢 Jenifer Niles 的支持，感谢技术编辑们帮助我们修改了文字，感谢所有制作小组的成员，你们的帮助成就了这个令人自豪的项目。



目 录

译者序		
序		
前言		
致谢		
第1章 游戏开发入门	<i>1</i>	
1.1 越来越复杂的开发过程	<i>1</i>	
1.2 更好的开发方式	<i>2</i>	
1.3 为什么使用 Lua	<i>3</i>	
1.4 本章小结	<i>4</i>	
第2章 脚本语言	<i>5</i>	
2.1 脚本语言简介	<i>5</i>	
2.2 Lua 简介	<i>6</i>	
2.2.1 Lua 的历史	<i>7</i>	
2.2.2 Lua 授权	<i>7</i>	
2.3 本章小结	<i>8</i>	
第3章 游戏开发世界的 Lua 语言	<i>10</i>	
3.1 脚本语言和游戏	<i>10</i>	
3.2 游戏项目中的 Lua	<i>11</i>	
3.2.1 游戏界面	<i>11</i>	
3.2.2 管理游戏数据	<i>12</i>	
3.2.3 事件处理	<i>14</i>	
3.2.4 保存和读取游戏状态	<i>14</i>	
3.2.5 人工智能	<i>15</i>	
3.2.6 快速构建原型	<i>16</i>	
3.3 本章小结	<i>16</i>	
第4章 Lua 入门	<i>17</i>	
4.1 使用 Lua 控制台	<i>17</i>	
4.2 Lua 基础	<i>19</i>	
4.3 变量	<i>21</i>	
4.3.1 nil	<i>21</i>	
4.3.2 Boolean	<i>21</i>	
4.3.3 string	<i>22</i>	
4.3.4 Number	<i>22</i>	
4.3.5 table	<i>23</i>	
4.3.6 局部变量和全局变量	<i>23</i>	
4.4 运算符	<i>24</i>	
4.4.1 算术运算符	<i>24</i>	
4.4.2 关系运算符	<i>24</i>	
4.4.3 逻辑运算符	<i>25</i>	
4.5 控制结构	<i>26</i>	
4.5.1 if	<i>27</i>	
4.5.2 while 和 repeat	<i>27</i>	
4.5.3 for	<i>28</i>	
4.5.4 break	<i>29</i>	
4.6 本章小结	<i>29</i>	
第5章 深入学习 Lua	<i>30</i>	
5.1 函数	<i>30</i>	
5.1.1 单一参数	<i>31</i>	
5.1.2 多个参数	<i>31</i>	
5.1.3 返回值	<i>32</i>	
5.2 标准库	<i>34</i>	
5.2.1 assert(myValue)()	<i>34</i>	

5.2.2	dofile(filename)	35	6.2.3	命令处理	51
5.2.3	math. floor()	36	6.2.4	退出程序	52
5.2.4	math. random()	36	6.2.5	cLua 对象和 LuaLib	52
5.2.5	math. min()	37	6.2.6	使用 cLua 的例子	53
5.3	字符处理	38	6.2.7	LuaGlue 函数的优点	55
5.3.1	类型转换	38	6.2.8	LuaGlue 函数：参数和 返回值	55
5.3.2	string. char(n1, n2, ...)	38	6.3	本章小结	56
5.3.3	string. len(myString)	38	第 7 章	Lua 与 C++ 的交互	57
5.3.4	string. sub(myString, start, end)	39	7.1	重新审视 LuaGlue 函数	57
5.3.5	string. format()	39	7.2	C++ 代码和 Lua 的交互	58
5.3.6	string. find(sourceString, findString)	40	7.3	事件驱动的编程	58
5.3.7	字符和格式	40	7.3.1	示例事件	58
5.4	table 数据结构	42	7.3.2	事件的参数	59
5.4.1	table. getn(myTable)	43	7.4	错误处理	60
5.4.2	table. insert(myTable, position, value)	43	7.5	本章小结	61
5.4.3	table. remove(myTable, position)	44	第 8 章	开发准备	62
5.4.4	table 引用	44	8.1	Visual C++ 6.0 工作区	62
5.4.5	多维 table	44	8.2	DirectX 基础	63
5.4.6	pairs()	45	8.3	LuaGUI 简介	65
5.5	I/O 基础	46	8.3.1	启动 GUI	66
5.6	本章小结	47	8.3.2	界面	66
第 6 章	Lua 与 C/C++ 程序的 整合	48	8.3.3	界面控件	66
6.1	初期设计要点	48	8.3.4	事件	67
6.1.1	Lua 环境	48	8.3.5	与 GUI 系统相关的 LuaGlue 函数	67
6.1.2	LuaGlue 函数	49	8.3.6	Shell 程序的扩展	68
6.2	基本实现方式	49	8.4	调试窗口	69
6.2.1	创建 Lua 运行环境	50	8.5	Windows 注册表	69
6.2.2	添加 LuaGlue 函数	51	8.6	本章小结	70
第 9 章	设计 Lua 版本的实现	71			
9.1	游戏设计原则	71			
9.1.1	什么是游戏	71			

9.1.2 了解玩家的想法 ······	72	11.5.4 主菜单界面 ······	125
9.2 基础库设定 ······	73	11.5.5 Controls 界面 ······	130
9.3 设计文档 ······	78	11.5.6 InGame 界面 ······	132
9.4 Lua 编程规范 ······	81	11.6 本章小结 ······	135
9.5 本章小结 ······	83	第 12 章 Lua 游戏编程 ······	136
第 10 章 使用 Lua 处理游戏数据 ···	84	12.1 游戏主循环 ······	136
10.1 简单的游戏数据 ······	84	12.2 井字棋 ······	137
10.1.1 太空飞船的例子 ······	85	12.2.1 游戏的初始化 ······	138
10.1.2 《Take Away》的玩家		12.2.2 游戏回合处理 ······	139
飞船 ······	88	12.2.3 模拟游戏回合 ······	147
10.1.3 敌舰数据 ······	89	12.3 《Take Away》游戏的实现原理 ······	147
10.1.4 补给箱数据 ······	91	12.3.1 InGame ······	147
10.2 大数据集 ······	92	12.3.2 使用计时器 ······	152
10.2.1 表单型数据 ······	93	12.3.3 玩家操作 ······	154
10.2.2 Lua 格式的数据文件 ······	95	12.3.4 子弹运动 ······	156
10.3 使用 Lua 保存游戏数据 ······	96	12.3.5 飞船移动 ······	158
10.3.1 案例 1——《Frontrunner》 ···	106	12.3.6 绘制活动的物体 ······	161
10.3.2 案例 2——健身大亨 ······	107	12.4 本章小结 ······	163
10.4 本章小结 ······	108	第 13 章 使用 Lua 定义和	
第 11 章 Lua 驱动的 GUI ······	110	控制 AI ······	164
11.1 GUI 系统概要 ······	110	13.1 智能的体现 ······	164
11.2 GUI 的 C++ 类 ······	111	13.2 21 点游戏 ······	165
11.2.1 GUI 控件: Sprite ······	112	13.3 井字棋 ······	170
11.2.2 GUI 控件: TextField ······	113	13.4 《Take Away》游戏的实现 ······	175
11.2.3 GUI 控件: Button ······	113	13.4.1 掠夺舰 ······	175
11.2.4 界面 ······	114	13.4.2 攻击舰 ······	176
11.2.5 GUI 管理器 ······	115	13.4.3 冲击舰 ······	176
11.3 GUI LuaGlue 函数 ······	116	13.4.4 混合舰 ······	177
11.4 进一步的说明 ······	118	13.4.5 控制飞行方向 ······	178
11.5 Lua 游戏界面 ······	119	13.4.6 碰撞检测 ······	179
11.5.1 界面设计原则 ······	119	13.5 其他 AI 的例子 ······	183
11.5.2 快速创建界面 ······	120	13.5.1 静态追踪 ······	183
11.5.3 载入界面 ······	121	13.5.2 近距离追踪 ······	185

13. 5. 3 动态追踪	186	14. 3. 2 坦克示例	222
13. 5. 4 预判型追踪	186	14. 4 2D 粒子系统	226
13. 5. 5 炮塔攻击	188	14. 5 本章小结	231
13. 5. 6 躲避攻击	189	第 15 章 最后说明	232
13. 5. 7 防御性射击	190	15. 1 添加音效和音乐	232
13. 5. 8 攻击伤害	191	15. 1. 1 LuaGlue 函数	232
13. 6 有限状态机	192	PlaySound	233
13. 7 路径寻找	194	15. 1. 2 音乐	234
13. 7. 1 算法概要	194	15. 2 使用编辑器	234
13. 7. 2 路径寻找示例	196	15. 3 调试 Lua 脚本	235
13. 7. 3 Lua 实现	197	15. 3. 1 通用原则	236
13. 8 本章小结	205	15. 3. 2 调用 DoFile 函数	237
第 14 章 Lua 和图像	206	15. 3. 3 Lua 错误消息	238
14. 1 运行绘图示例	206	15. 3. 4 使用实时调试窗口	238
14. 1. 1 指纹示例	206	15. 3. 5 使用文本框	239
14. 1. 2 爆炸示例	208	15. 3. 6 使用文件输出	240
14. 2 线性移动	213	15. 4 资源管理	241
14. 2. 1 GetCollisions 函数	216	15. 4. 1 资源的组织	241
14. 2. 2 HitTest 函数	218	15. 4. 2 运行时的文件夹	242
14. 2. 3 进一步的说明	219	15. 5 发布 Lua 代码	242
14. 3 碰撞检测	219	15. 6 许可证	244
14. 3. 1 LuaGlue 函数	220	15. 7 进一步的说明	245
SetTexture	220	15. 8 本章小结	246

游戏开发入门

本章要点

- 越来越复杂
- 更好的方式
- 为什么选择 Lua

创建自己的游戏是最激动人心的事情之一。创造可以传递快乐，在游戏中经历挑战和胜利的快感，对于参与者是一次心灵的冲击。

如果你是一名游戏爱好者，一定体验过第一次玩自己的游戏时的那种快乐，还有当看到伙伴玩你的游戏时，发自内心的喜悦和兴奋的那种满足感。

这种感觉对于资深的游戏开发者来说也是一样，我们同样在乎我们开发的游戏，没有什么比看到别人快乐地游戏更让人感到激动的了，因为其中饱含着爱、血汗、眼泪和真心付出。

1.1 越来越复杂的开发过程

许多年前，大部分游戏是开发者在车库和地下室、利用周末或业余时间开发的。现在若制作能够在当地电子市场售卖的游戏，则需要许多专业的开发者分工协作。

复杂度逐渐增长导致了专业的分工。游戏美术设计人员负责制作 2D 或 3D 动画以及静态模型，程序员实现网络编程、人工智能（AI）和 3D 渲染。在这种专业的分工下，想要保持过去那种灵活并富有创造性的游戏开发过程越来越难。

开发团队规模的不断增长，以及游戏复杂度的不断提高，使得不同游戏系统之间的依赖性也在提高。这些依赖性则直接导致了开发周期延长，游戏设计想法无法验证，创新和游戏灵感也不得不由于开发周期紧张而有所限制。

若干年前，我得到一个机会拜访了一家知名的游戏工作室，他们当时正在开发一款第三人称冒险射击类游戏。这个游戏看起来很“酷”，3D 场景绚丽多彩，与环境的交互也显

得十分真实。

我有幸看到了游戏制作的整个过程。首先，3D 美术设计师用建模软件创建了一个游戏场景，把这些模型导入一个内部工具中，让设计师设置各种触发区域，当玩家角色或者 AI 控制的敌人进入该区域时触发特定的游戏事件。然后，设计师会坐下来和程序员交流每一个触发区域，告诉他们期待发生的结果，程序员做好各种笔记，接着花许多天时间来实现这些代码。完成之后，设计师会检验成果，并提出一些修改意见，然后整个过程再不断重复。

尽管结果是可靠的，但可以想象这个过程非常艰苦，不仅耗时而且呆板。我知道一定有更好的方式，不过在那个时候我对脚本语言还一无所知。

1.2 更好的开发方式

更好的方式是使用中层脚本语言来构建项目，它可以帮助游戏设计师把握整个开发的交互过程，让程序员去做大量更基础的工作。

从游戏开发者的角度看，脚本语言可以帮助用户很容易地返回游戏开发过程。也许需要几个小时来构建一个“干净”的游戏项目，但脚本语言可以帮助用户快速做出修改并且立刻看到游戏的效果。游戏设计师可以独立于程序员尝试新想法，游戏美术设计师可以创建图形界面把游戏流程和功能组合到一起。

脚本语言存在于由软件工程师编写并编译后的代码之上，通常是在运行时编译，是一种方便设计师或者程序员处理和控制数据的简单语言。

为什么要使用脚本语言呢？对于资深从业人员或者业余开发者来说，这都是一个值得关注的问题。从游戏设计师的角度来说，使用脚本语言开发游戏可以很清楚地界定底层代码和游戏玩法代码。通常，在引入了脚本语言的项目中，底层模块交给像 C++ 这样的核心语言，诸如界面交互、数据管理、人工智能和事件处理等，一般使用脚本语言实现。这种职责的划分可以让用户的游戏更加稳定，并且使得并行开发成为可能。

脚本语言还能使开发团队中的非技术成员参与到核心开发过程中。界面美术师不仅只制作界面素材，还能独立于程序员编写让界面运行在游戏中的脚本框架。要实现这些想法，设计师则可以不必麻烦程序员，而直接着手 AI、数据处理或者创建场景脚本。

相对于 C++ 这样的底层语言，脚本语言本身是易学易用的。由于 Lua 语言无须关心复

杂的内存管理、对象渲染或者 TCP/IP 网络通信，所以十分容易上手并投入实际开发。学习脚本语言不需要花费很长时间，开发者可以在几个小时内就掌握它的语法和功能。

在本书中，我们将自下而上地学习如何使用 Lua 语言并利用 C++ 的功能来创建一个完整的游戏。在这个过程中，会特别向读者展示脚本语言是如何优化开发过程的，不管是资深开发者还是业余爱好者，都将会有所收获。

脚本语言有很多，那我们为什么要选择 Lua？纵观整个脚本语言领域，我们可以看到有 Perl、Tcl、Ruby、Forth、Python、Java 和 Lua。尽管所有的脚本语言在特定领域都有自己的一席之地，但在游戏开发的世界中，Python 和 Lua 是非常适合的（因为它们可以直接调用 C++ 的功能）。

许多商业游戏已经成功地使用了 Python 和 Lua，因为它们都有很强的兼容性，所以可以与编译后且基于 C++ 技术的模块协同工作，而且还能扩展。如果读者有机会询问一下程序员对于这两种语言的看法，通常会有非常不同的观点，还可能有激烈的争论，就像体育中的“德比”赛事那样（想象一下芝加哥小熊对白袜，纽约喷气机对巨人，纽约大都会对洋基）。其实，这两种语言都是游戏开发领域中非常出色的工具。

1.3 为什么使用 Lua



对于游戏开发而言，Lua 是较好的选择，其设计的核心目标是可扩展性，因此在最初设计时就考虑到要能够集成在大型应用中。因为有了这样的设计目标，所以非常容易在应用程序中加入 Lua 脚本。Lua 的易集成的特性还使得 Lua 可以很方便地与父程序通信。游戏程序员都希望脚本语言能够简单地实现游戏设计，在这方面，Lua 也能够胜任。

Lua 免费、小巧、快速且易移植。所有的游戏开发者和游戏公司都喜欢“免费”的工具。通常讲，一分钱一分货，但是对于 Lua 来说，它完全超出你的预期。Lua 采用了非常灵活的发布协议，它有极少的源代码，运行轨迹十分紧凑，在编译时间和运行时内存占用上都有很好的性能表现。

要说 Lua 最让人惊喜的地方，应该是它的执行速度。对于任何脚本语言的技术方案，游戏开发者的第一反应就是：“脚本语言太慢了，帧率一定不会很理想。”但这个说法对 Lua 是不成立的，事实上，我们还没有看到任何一个项目因为 Lua 的使用而造成瓶颈。最后，游戏开发界正在迎接一次新的硬件周期，我们将要学习如何使用一组新的平台。因为

Lua 的易移植性，当我们的技术储备转移到新的平台时，至少有一部分是不会过时的。

Lua 是非常容易学习的语言。不需要了解很高级的编程概念（如对象和继承），大部分具有计算机学习背景的人都可以在短期内掌握它的基础知识并且马上投入正式的工作。如果团队成员熟悉其他的语言，那么 Lua 可以轻松上手，很适合那些非程序员背景的团队成员，它们也能对游戏功能和美术部分进行修改或创建。

在我公司，最近刚刚发布了第 13 款使用 Lua 开发的游戏。我们的团队虽然很小，但是也有程序员、美术师和设计师的标准构成。当我们开始一个新的项目时，首先要确定项目的技术需求（什么是我们还没有而需要去实现的？）和设计需要的功能。程序员可以负责技术设计部分，专注于那些他们擅长的技术难题。同时，设计师和美术师可以马上开始着手界面流程和核心游戏功能的设计工作。通常，美术师（包括 2D 和 3D）还会花一部分时间确定游戏的视觉需求。与此同时，3 位熟悉 Lua 的设计师开始构建游戏基础、游戏数据及核心游戏系统。他们甚至都不需要等着程序员，如果需要什么功能一般就直接先用 Lua 代替。这样我们就可以进行非常高效率的游戏开发，因为团队的每个人都能从一开始就“热火朝天”干起来。

有一个项目要特别提起，在我们为 2004 年美国总统大选开发选举游戏模拟器的时候，我们已经可以用 Lua 开发出完整的游戏原型来验证 AI 和游戏流程，然后再回过头来重新用 C++ 实现那些核心的部分。快速的原型开发可以让一个开发者就能够完成设计和开发环节最有价值的部分，这也是业界少有的高效率。

1.4 本章小结

游戏开发是一个令人兴奋的产业，人们一旦有了灵感就想马上开始行动。Lua 赋予了我们这样的能力，让我们可以快速实现游戏核心概念，设计并测试界面，并且方便地管理大量的运行时数据，而这一切都不需要读者有很深厚的技术背景。

在以后的章节中，我们将要学习如何在游戏项目中使用 Lua，并学习语言本身的特性。然后，我们将开发一个完整的游戏，一起经历从开始到完工的整个过程。完成了这些，我们就可以很好地掌握这种灵活、小巧的开发语言，并切身感受到它给你的游戏和游戏开发体验所带来的不同。

脚本语言

本章要点

- 脚本语言简介
- Lua 简介

虽然计算机可以做很多事情，从生成报表到模拟经营属于你自己的主题公园，但是计算机自己不会思考，它需要接受系统化的指令来工作。大部分用户通过像 Word 文字处理器或者电子表格工具这样的应用程序来为计算机指派任务。软件工程师则使用像 C ++ 这样的底层编程语言让计算机工作（通常是为普通用户开发应用程序）。而我们所说的脚本语言，存在于操作便捷的应用程序和开发软件的底层编程语言这二者之间。

2.1 脚本语言简介

脚本语言可以方便地与计算机底层功能交互。这体现在它常常被当做批处理命令工具，即发送一系列重复的指令给命令处理器的工具。所以早期的脚本语言常常称做批处理语言或者作业控制语言。

一个熟悉的例子就是 MS-DOS 时期的的老的 *.bat 文件，这种批处理文件就是简单的文本文件，它包含一系列顺序执行的 DOS 命令。该语言本身就是 DOS 命令集合，通过进一步扩展成为一种伪脚本（参考下面的示例）。

```
copy g:\whitehouserun\working\whr.exe  
copy g:\whitehouserun\working\whrd.exe  
ren *.txt *.lua  
copy *.lua g:\luabank\whitehouserun
```

计算机语言用于解决一些特定的问题，从系统控制级别的 C 和 C ++ 到人工智能处理语言（如 LISP）。脚本语言通常拥有一些共同点，他们一般用在快速开发中（低成本、高效率），并采用接近自然语言的语法，对于非程序员背景的人更易于书写和阅读，这样有

一定基础的用户就可以在没有程序员的帮助下编写和使用脚本语言。脚本语言在调用其他底层语言开发的模块方面十分出色。

脚本文件都是在载入时解释和编译（不是预编译，而是在调用时才处理）。以 Lua 为例，它只有在载入时才被编译成二进制形式并存在于内存中，直到被释放。

在软件开发（特别是游戏开发）领域，结合使用脚本语言和底层语言可以让开发者更好地控制运行环境，使得在开发过程中，在运行环境上的修改和测试都拥有更大的灵活性。

2.2 Lua 简介

Lua 和传统的脚本语言不同，它是一种易整合语言（glue language）。一般的脚本语言用于控制执行重复的任务，而易整合语言可以让使用者把其他语言开发的功能整合在一起。这样就让脚本程序员有了更大的发挥空间，而不仅仅局限于执行命令。程序员可以使用这种脚本在底层语言开发的功能模块基础上创建新的命令。本书将探讨如何使用 Lua 来整合 C++ 的与游戏相关的一些功能，如 GUI、AI、数据等。

Lua 本身是一种简单而又强大的编程语言，它可以让脚本程序员完成大量的处理。这种语言拥有很强大的字符处理和数学运算能力、灵活的数据类型（很快就可以熟悉），以及定义函数的功能。但是如果缺少整合其他环境的组件的“魔力”，这些基础的特性也就丧失了。（没错，你可以在命令行下执行 Lua 脚本并查看运行结果，但除了学习语言本身，对于游戏开发来说，命令行式的输出是没有实际意义的。）学习其他编程语言经典的第一课是如何输出“hello world”，Lua 版本的方法参见代码清单 2.1。

代码清单 2.1 用 Lua 编写的“hello world”程序

```
--Lua's "hello world"  
myString = "hello world"  
print(myString)
```

Lua 非常适合作为更强大的底层编程语言的搭档，如 C++。Lua 能让游戏开发者快速建立游戏原型甚至是完整的游戏。游戏开发者可以在没有程序员帮忙的情况下构建整个图形界面。它还可以用来管理游戏进度文件的保存和载入，而且很容易阅读和调试。在游戏开发领域，Lua 能帮助开发者构建一个高效并且方便验证游戏想法的环境。

按照开发 Lua 的团队的描述，Lua 是一个可以集成在应用程序中的“语言引擎”。它本身是一种编程语言，并且还提供了很多可以和应用程序交换数据的 API（应用编程接口）。另外，Lua 还能够通过整合 C++ 的模块来进行功能的扩展（这个就是我们之前所说的“整合”功能）。和程序开发语言（如 C++）配合使用时，Lua 也可以用来作为特定项目的框架语言。这种易扩展性使 Lua 非常适合作为游戏开发的环境。

作为独立的编程语言（在运行窗口中执行），Lua 功能很有限，只能用做教学工具。（我们会在接下来的章节中使用控制台学习该语言。）Lua 只有集成在其他语言中才能发挥它的价值。它的实现非常简单，仅仅通过一些 LuaGlue 函数就可以和底层语言通信，在用户自定义 LuaGlue 函数的基础上，它还可以进一步被扩展，甚至成为一种新的编程语言。

2.2.1 Lua 的历史

Lua 在葡萄牙语中是“月亮”的意思，1993 年由巴西的 Pontifical Catholic University 开发。该语言是由一个来自计算机图形技术组织（Tecgraf）的团队（Roberto Ierusalimschy、Waldemar Celes 和 Luiz Henrique de Figueiredo）开发，并作为自由软件发行。Lua 开发小组的目标是开发一种小巧、高效并且能够很好地和 C 语言一起工作的编程语言。在脚本语言领域，Lua 是最快、最高效的脚本语言之一，因此它有资格作为游戏开发的备选方案。Lua 的内核小于 120KB（Python 的内核大约 860KB，Perl 的内核大约 1.1MB），当编译和集成到游戏开发系统中时非常小巧。Lua 通常比 Python 这种流行的游戏开发脚本语言运行更快速，完整的性能测试报告可以在计算机编程语言实战性能测试网站中找到（<http://shootout.alioth.debian.org/>）。

计算机图形技术组织（Tecgraf）成立于 1987 年，致力于开发和维护用于技术和科技领域的计算机图形和用户界面。除了 Lua，Tecgraf 小组还开发了 IUP（一种开发用户界面的系统）、CanvasDraw（跨平台的图形库）、TWF（一种用于 Web 页面的图形文件格式）和其他一些系统。读者可以访问相关网站获取更多信息。

2.2.2 Lua 授权

Lua 是免费的开源软件，可以免费用于科研及商业应用。关于开源软件的更多信息可访问 www.opensource.org。

对于游戏开发专业人员，授权费对于开发技术的选择影响很大。通常，对于一个项目，游戏引擎的预算会超过 50 万美元，中间件技术会花费 5 千美元 ~5 万美元不等。因此，开源软件对于开发团队来说是很有吸引力的。

开源软件因为本身不盈利，所以它们的代码通常有很多 bug，又由于没有太多注释，因此难以理解。另外，没人为技术支持付钱，所以也谈不上什么技术支持。

Lua 则避免了这些问题，它小巧，实现简单（而且还在维护），代码简洁、清晰。Lua 的开发团队是由具有计算机工程背景的专家组成，并且一直在关注着它的升级。和其他开源项目不同，设计 Lua 旨在项目中扩展功能，而不是在 API 级别，因此它的内核一直很稳定。

Lua 授权的精神在于用户可以在任何情况下免费使用，并且不需要取得版权所有者的许可。如果用户想知道更多 Lua 授权的信息，那么可以访问 www.lua.org/license.html。



完整的 Lua 5.0 授权如下 (CD-ROM 中也提供了该授权)：

Lua 5.0 License

Copyright ©2003-2004 Tecgraf, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2.3 本章小结

脚本语言最初只是一种简单的工具，目的是为了让资深用户在简单文本文件中能够批

量执行常用命令。现在许多脚本语言，如 Python、Ruby 和 Lua，可以提供像计算机编程语言一样强大而灵活的功能。最近几年，脚本语言已经“走”到了游戏开发业界的前端，作为一个可行的中间件产品，它可以有效地提升开发小组的工作效率以及游戏项目的性能。来自巴西的计算机图形技术组织（Tecgraf）的团队开发的 Lua，由于它的小巧、高速以及与 C 和 C++ 良好的兼容性，成为非常适合游戏开发的工具。Lua 是免费的开源语言并且有良好的技术支持，再加上不断增长的忠实用户群，让它成为专业开发者及业余游戏开发爱好者的不错选择。



第3章

游戏开发世界的 Lua 语言

本章要点

- 脚本语言和游戏
- 游戏项目中的 Lua

现实中的游戏开发常常面临两种互相矛盾的压力，一方面需要测试和验证新想法，另一方面又需要快速开发并且按时交付。如果适当地在项目中引入脚本语言，可以让资深工程师充分发挥他们的能力，开发出优秀的游戏系统和模块。

3.1 脚本语言和游戏

脚本语言可以让美术师直接开始界面设计，让设计师和初级程序员（脚本语言是一种让新手快速进入游戏开发的很好的方式）立即着手游戏流程和逻辑的开发，让关卡设计师能迅速掌控游戏环境和游戏体验。

脚本语言不是非常高效——它们没有原生代码的运行效率，因此不适合作为开发高性能需求处理的工具。但易整合语言能够利用原生语言编写的模块扩展功能，比如 Lua，可以作为控制机制来调用原生代码编写的高性能处理组件。（Lua 是运行效率最高的脚本语言之一，因此大部分性能方面的问题都可以不用担心。）C 函数可以利用自己高性能的特点，并且整合到 Lua 中，让脚本程序员可以利用这些功能。

这种处理的一个例子是在游戏世界中放置一个 3D 模型的功能。渲染系统完全由 C++ 开发，但 Lua 可以调用 C++ 来创建一个特定模型的实体对象，并且设置其在场景中的位置，然后 Lua 还可以为这个 3D 模型指定动画。Lua 并不处理任何实时的复杂运算来改变该模型，而只是告诉底层渲染系统什么时候该做什么。在下面的例子中（参见代码清单 3.1），AddEnvironmentObject() 是一个 LuaGlue 函数——它可以直接调用 C++ 方法并且让 Lua 能够控制底层的 3D 渲染功能。这种函数由 C++ 程序员编写，把底层的功能提供给

脚本程序员和设计师。

代码清单 3.1 使用 Lua 脚本在场景中放置 3D 模型

```
--Lua script to add room, table, and human models
--to a 3D runtime environment
envmID = AddEnvironmentObject("casino_02.mlg",0,0,0,0,0,0)
tableID = AddEnvironmentObject("poker_table_normal_02_5card.mlg",-
2,0,1.5,0,0,0)
if bodyID1 ~= nil or headID1 ~= nil then
    PositionChildEntity(tableID, bodyID1, "anchor_0")
end
-- This sets the color of the Ambient light in the scene
AddLight(LIGHT_AMBIENT, 30, 30, 30)
-- This Adds a directional light source to the scene
AddLight(LIGHT_DIRECTIONAL, 1.0, -1.0, 1.0, 255, 255, 255)
-- set initial camera look at
StartCamera(0,0,.556, 0,3,-5, 0.03,1.5,3,20,.5,2,0.001)
```

3.2 游戏项目中的 Lua

把脚本语言集成到游戏项目中可以提升团队的开发效率，并且可以很好地扩展原生编译语言的能力。Lua 在游戏开发的许多基础领域中都表现得很出色。

在游戏开发团队中，可能有许多成员都使用 Lua 来完成他们的工作。程序员负责将 Lua 整合到游戏开发环境中，通常，他们会需要编写一些 Lua 代码。游戏设计师是脚本语言的主要使用者，因为他们和上层的游戏设计和数据直接打交道。美术师也会经常使用 Lua，进行诸如界面布局、设计和 3D 场景中各种模型的摆放之类的工作。

Lua 是非常强大的工具，可以用来完成下面这些工作：

- 编辑游戏的用户界面
- 定义、存储和管理基础游戏数据
- 管理实时游戏事件
- 创建和维护开发者友好的游戏存储和载入系统
- 编写游戏的人工智能系统
- 创建功能原型，可以之后用高性能语言移植

3.2.1 游戏界面

游戏界面是玩家和你的游戏进行交互的媒介。游戏界面是一个游戏最基本的部分，因

为它负责所有和玩家的交互工作。因为它的重要性，所以需要能够高品质、高性能地运行，并且能够被测试和持续改善，以最大化地提升用户体验。

Lua 可以让界面设计师迅速创建所有主要的界面元素——进行界面布局、管理用户输入并且输出游戏数据。利用游戏程序员开发的一部分核心的界面控件和控制函数，界面设计师不仅能负责界面的美术观感还能控制游戏的交互。这不仅节省了程序员的时间，而且给了美术设计师更大的创作空间，同时还能为界面设计的测试节省出许多时间。代码清单 3.2 展示了使用 Lua 创建文本 GUI 控件的方法。

代码清单 3.2 使用 Lua 创建文本 GUI 控件

```
--Text object example
CreateItem(200, "TextField")
SetItemPosition(200, 300, 20, 200, 28)
SetFont(200, "Arial", 16)
ItemCommand(200, "SetColor", 255,255,255,255)
ItemCommand(200, "SetString", "I am a text object")
```

3.2.2 管理游戏数据

管理游戏数据对于游戏开发者一直都是一种挑战。游戏数据定义了游戏世界中所有对象的参数和特性，如脉冲枪升级费用、气垫船的行驶速度等。对于数据密集型游戏来说，开发者常常利用电子表格工具来输入和保存数据，然后创建解析工具将其转换成游戏中可以使用的格式。这种方法一般用在数据密集型游戏中，如角色扮演类游戏（NPC 或者非游戏玩家角色的信息以表格结构形式存储）或者策略类游戏（角色单位信息保存在数据表中）。

Lua 可以让这个存储系统更为简单，它以 Lua 文件作为存储介质让程序使用相同的数据。通过创建一个简单的数据管理系统，变量和类型可以定义在 Lua 中，然后可以很容易地读取。因为不用关心整个数据处理过程，所以游戏设计师可以按照需要修改、增加和缩减游戏数据，而不需要程序员的协助。因为 Lua 语法的特性和在脚本中添加注释的能力，数据文件是易读的。如果需要转换工具，也非常容易开发，把 Lua 数据输出到 Lua 文件，然后在运行时载入。

Lua 本身并没有可以直接访问外部数据库的能力，但可以用 C++ 开发访问数据库的组件，然后再利用 LuaGlue 函数整合该组件来达到目的。

在代码清单 3.3 中，我们可以看到如何使用 Lua 示例文件来直接保存游戏数据。在这

个例子中，通过使用 LuaGlue 函数，向由核心代码管理的数据结构中写入数据。代码清单 3.3 的输出可参照图 3.1。

代码清单 3.3 使用 Lua 保存美国总统选举模拟游戏（《Frontrunner》）的数据

```
-- Auto-generated LUA campaign data file
-- File created on: 07/28/04 12:02:08
AddCampaign( newID)
SetCash( newID, 38000000)
AddPerson( newID, CANDIDATE, "John Kerry", 70, 85, 100, SELF)
AddPerson( newID, RUNNING_MATE, "John Edwards", 65, 80, 100,
RUNNING_MATE)
SetCampaignData( newID, "Portrait", "ui_kerry.bmp")
SetCampaignData( newID, "RunningMatePortrait", "ui_edwards.bmp")
SetCampaignData( newID, "Name", "Democratic Party")
SetPersonLocation( newID, CANDIDATE, 22)
SetPersonLocation( newID, RUNNING_MATE, 34)
SetPersonHomeState( newID, CANDIDATE, 22)
SetPersonHomeState( newID, RUNNING_MATE, 34)
SetPersonFatigueValue( newID, CANDIDATE, 1)
SetCampaignColor( newID, 0, 200, 0, 255)
SetCampaignData( newID, "ColorName", "blue")
SetCampaignThinkTime( newID, 2)
SetCampaignData( newID, "LastAction", "Rest")
SetCampaignData( newID, "UniOrg", "3")
SetUniversalOrgLevel( newID, 3)
```



图 3.1 游戏《Frontrunner》的截图，Lua 数据的实时显示

3.2.3 事件处理

通常，游戏中的绝大部分重要的处理都是由事件驱动的，要么是来自游戏角色，要么是来自游戏中那些交互的代理。这些事件可以是简单的，如用户按下了 <W> 键，也可以是复杂的，如两个游戏实体同时来到了某个地方。

事件驱动的编程，对于熟悉了 Windows 开发（事件是 Windows GUI 操作的基础）的用户来说不是什么陌生的技术。在 C++ 精心开发的事件系统中，使用 Lua 来接收和处理这些事件，用户可以在游戏内部运行机制、高级 Lua 函数和用户输入之间创建清晰的反馈流程。一个简单的例子是，获取键盘输入然后向 Lua 事件处理器发送该事件，同时将该输入值显示出来。这个概念将一直贯穿于本书之中——事件会返回事件类型 ID 和驱动事件的物体 ID，参照代码清单 3.4。

代码清单 3.4 获取按键事件的例子

```
SetEventHandler("MainMenuEvent")
function MainMenuEvent(id, eventCode)
    if eventCode == GUI_EVENT_BUTTON_UP then
        if ID == OK_BUTTON then
            AddTextToConsole("OK button pressed")
            PlaySound("button_click.wav")
        end
    end
end
```

3.2.4 保存和读取游戏状态

保存和读取玩家的数据是游戏开发项目中最具有挑战的事情之一。因为玩家有时需要暂时离开游戏，所以需要一种保存游戏进度的方法。玩家还需要在某种新的尝试和挑战前保存游戏状态，这样万一尝试失败了还可以恢复游戏进度。同样，在开发和测试过程中，开发者会经常需要读取特定的游戏状态来验证游戏功能或者确认 bug 是否已修复。

如果使用 Lua 保存用户的核心游戏数据，那么就可以用 Lua 作为保存和读取当前游戏状态的系统。在 Lua 中，游戏进度文件是简单的可执行的 Lua 代码的文本文件。程序员或者设计师可以通过进度文件来获取当前游戏的状态（也可以修改），读取游戏进度就和执行 Lua 脚本一样简单。利用 Lua 标准的输入/输出函数，编写一个函数来保存游戏数据到可执行的 Lua 脚本中是最直接的方法。这个系统的优点是可以让设计师在开发过程中，根据

游戏数据的增长和删减来编辑并修改这个函数（不需要特别的读取函数）。在开发的最后阶段，还可以利用脚本编译函数来为游戏数据加密。

3.2.5 人工智能

人工智能（AI）在如今的游戏中非常关键——玩家需要精明的、有挑战性的对手，感觉就像真人一样。游戏开发者明白真实世界的 AI 不是说完全模仿人如何玩和反应，而是为玩家创造这种感觉。多年来，开发者一直在争论计算机对手“作弊”（计算机对手可以比玩家访问更多的游戏数据）的优缺点。这种争论最好用在别的时间和地方。不过，几乎所有的开发者都同意，比起 AI 模拟，AI 行为的玩家感知更为重要。

就开发 AI 判定来说，Lua 是一种非常高效的工具。有许多人工智能组件，如路径寻找，最好留给底层语言来实现。路径寻找（计算机控制的物体在虚拟世界中的路径寻找）是一个数据运算量很大的工作，计算机需要反复测试可能的路径来寻找最短或者最直接的路径。（路径寻找最好整合到上层的 LuaGlue 函数中以便控制相关参数，但还是会后面的章节中介绍一种 Lua 的实现。）另一个例子是用最大最小值方法实现的移动判定，一般被用在计算机象棋游戏中，预测之后几步的移动并尝试计算出最优的移动步骤。一般来说，“能思考的函数”都需要大量的数学计算，如导航树或者尝试错误法运算最好都留给底层代码。依赖有限的数据集合和参数的人工智能才更适合 Lua 的特点。Lua 的优点是设计师可以编写简单的模型来试错，并快速验证和迭代想法而不需要麻烦程序员。想要利用 Lua 高效率实现 AI，需要很仔细地设计函数（C 函数），让 Lua 脚本可以访问和交互游戏数据。使用 Lua 作为事件管理系统同样可以让 AI 设计师能应对游戏开发中的变更，开发出灵活反应的 AI 系统。

在代码清单 3.5 中，用户可以看到使用 Lua 来评估可能的竞选旅行目的地，评估的依据是这个州的选举人票数量和这个州的支持率。

代码清单 3.5 评估虚拟的美国总统候选人应该去哪个州的 Lua 人工智能函数

```
--AI candidate travel
StateTravTable = {}
stateCount = 1
--build a set of potential "travel to" states
while stateCount < 6 do
    pickState = math.random(1,51)
    --make sure there are enough electoral votes to make it worthwhile
    if GetStateData(pickState,"ElectoralVotes") > 12 then
```

```

        StateTravTable[stateCount] = pickState
        stateCount = stateCount + 1
    end
end
--pick the state out of the list with the lowest support
lowSupport = 100
targetState = 1
for indx = 1,5 do
    if TallySupport(StateTravTable[indx], Campaign) < lowSupport then
        lowSupport = TallySupport(StateTravTable[indx], Campaign)
        targetState = StateTravTable[indx]
    end
end
--issue the travel order
Travel(Campaign, Person, targetState)
SetCampaignData(Campaign, "LastAction", "Travel")

```

3.2.6 快速构建原型

商业化的游戏必须为玩家提供高性能的体验，掉帧和处理延迟的现象在如今竞争激烈的市场中是绝不允许的。程序概要分析（Profiling）是行之有效的确定性能瓶颈的方法，但必须在所有功能都开发完成并正确工作的前提下才能进行。原型开发和性能改善一般是不能同时进行的。

Lua 是构建可移植的核心游戏功能原型的不错工具。因为 Lua 可以整合原生语言开发的组件，所以移植单独的 Lua 函数到 C++，对于其他 Lua 函数来说是不需要变更的。它可以让设计师在构建原型时，如果碰到有高性能需求的函数就可以让程序员用底层语言来实现。因为程序的算法结构已经有了，所以 C 函数和 LuaGlue 调用都可以高性能、无缝地集成在项目中。

3.3 本章小结

在游戏开发领域，Lua 和 C++ 是一个功能十分强大的组合。对于 GUI 开发、事件管理、数据保存和获取、游戏进度保存和载入以及人工智能这些游戏开发工作，需要能够快速地构建原型、测试和迭代设计，而这些正是 Lua 所擅长的。在核心底层语言开发的项目中，频繁的变更会导致系统的不稳定，并让开发者不能专注于更重要的开发任务。使用 Lua 环境分担这些工作，可以让设计师和程序员快速而安全地使用脚本来设计、测试和实现游戏功能。