

游戏编程权威指南



[游戏编程权威指南_下载链接1](#)

著者:Mike McShaffry 麦克沙福瑞

出版者:人民邮电

出版时间:2016-3

装帧:平装

isbn:9787115410344

全书分为4个部分共24章。首部分是游戏编程基础，主要介绍了游戏编程的定义、游戏架构等基础知识。

第二部分是让游戏跑起来，主要介绍了初始化和关闭代码、主循环、游戏主题和用户界

面等。

第三部分是核心游戏技术，主要介绍了一些*为复杂的代码示例，如3D编程、游戏音频、物理和AI编程等。

第四部分是综合应用，主要介绍了网络编程、多道程序设计和用C#创建工具等，并利用前面所讲的知识开发出一款简单的游戏。

本书适合游戏开发人员、游戏架构设计人员和游戏引擎用户参考阅读，也适合想要进入游戏开发领域的读者阅读。

作者介绍:

Mike McShaffry，在游戏界众人皆知的"Mr. Mike"，参加了创世纪7、8、9和UO（网络创世纪）的开发和项目管理工作。Mike的游戏开发经验以非凡的方式覆盖了整个领域。当团队只有十来个人时他就在那里，然后经历了20、30、50人的团队阶段。他经历过创业，也为业界最大的发行商工作过，开发过从“传统”到绝对“非传统”的游戏——从《创世纪》到Blackjack，单机、多人、在线、离线等你能想到的任何东西。对于PC游戏，他能以每种权威身份发言——程序员、设计师、项目主管、开发主管、工作室领导……

目录: 第1章 什么是游戏编程 1

1.1好的方面 1

1.1.1工作 2

1.1.2游戏玩家 2

1.1.3同事 3

1.1.4工具—软件开发工具包（SDK） 4

1.1.5硬件 5

1.1.6平台 6

1.1.7展会 9

1.2不好的地方 9

1.2.1游戏编程很难 10

1.2.2零碎文件 10

1.2.3那不是bug—而是特性 11

1.2.4工具 12

1.3黑暗的一面 13

1.3.1命中移动的目标 13

1.3.2加班模式（和加班大餐） 13

1.3.3呸！胡扯 15

1.3.4操作系统地狱 15

1.3.5雇员流动的性质 16

1.4这一切都是值得的，对吗 16

第2章游戏中有什么 18

2.1游戏架构 18

2.2使用游戏架构 20

2.3应用层 22

2.3.1读取输入 22

2.3.2文件系统和资源缓存 22

2.3.3内存管理 23

2.3.4初始化、主循环和关闭 23

2.3.5其他应用层代码 24

2.4游戏逻辑 25

2.4.1	游戏状态和数据结构	25
2.4.2	物理学和碰撞	26
2.4.3	事件	26
2.4.4	进程管理器	27
2.4.5	命令解释器	28
2.5	人类玩家的游戏视图	28
2.5.1	图形显示	29
2.5.2	音频	30
2.5.3	用户界面表示	31
2.5.4	进程管理器	31
2.5.5	选项	31
2.5.6	多人游戏	31
2.6	AI代理的游戏视图	31
2.7	网络游戏架构	32
2.7.1	远程游戏视图	33
2.7.2	远程游戏逻辑	33
2.8	必须使用DirectX吗	34
2.8.1	DirectX的设计理念	34
2.8.2	Direct3D或OpenGL	35
2.8.3	DirectSound还是	35
2.8.4	DirectInput或自己实现	36
2.9	其他内容	36
2.10	补充书目	36
第3章	拯救了我的编码趣闻和风格	37
3.1	通用编码风格	38
3.1.1	大括号	38
3.1.2	一致性	39
3.2	智能代码设计实践	40
3.2.1	避免隐藏代码和重要操作	41
3.2.2	类结构：保持简单	42
3.2.3	继承VS.组合	42
3.2.4	变坏的虚函数	43
3.2.5	使用接口类	44
3.2.6	考虑使用工厂	45
3.2.7	封装变化的组件	46
3.2.8	使用流来初始化对象	46
3.3	智能指针和裸指针	47
3.3.1	引用计数	48
3.3.2	C++的shared_ptr	49
3.4	正确使用内存	52
3.4.1	了解不同类型的内存	53
3.4.2	优化内存访问	55
3.4.3	内存对齐	56
3.4.4	虚拟内存	57
3.4.5	编写自己的内存管理器	58
3.5	各种有用的东西	59
3.5.1	一个很棒的随机数生成器	60
3.5.2	集合的伪随机遍历	61
3.5.3	内存池	62
3.6	开发出适合自己的风格	67
3.7	补充书目	68
第4章	生成游戏	69
4.1	一个小动机	69
4.2	创建项目	70

- 4.2.1生成配置 70
- 4.2.2创建坚不可摧的目录结构 71
- 4.2.3将游戏引擎和工具放在何处 73
- 4.2.4设置VisualStudio生成选项 74
- 4.2.5多平台项目 76
- 4.3源代码库和版本控制 77
 - 4.3.1微软VisualSourceSafe的相关历史 79
 - 4.3.2Subversion和TortoiseSVN 79
 - 4.3.3Perforce软件的Perforce 80
 - 4.3.4Avid的AlienBrain 81
 - 4.3.5使用源代码控制分支 81
- 4.4生成游戏：一门黑色艺术 84
 - 4.4.1自动化生成 85
 - 4.4.2生成计算机 85
 - 4.4.3自动化生成脚本 85
- 4.5创建生成脚本 87
 - 4.5.1标准生成 87
 - 4.5.2里程碑生成 88
 - 4.5.3多个项目和共享代码 90
 - 4.5.4最后的建议 91
- 第5章 游戏初始化和关闭 92
 - 5.1初始化10192
 - 5.2C++初始化的一些陷阱 93
 - 5.3游戏的应用层 95
 - 5.3.1WinMain：Windows入口点 95
 - 5.3.2应用层：GameCodeApp 97
 - 5.3.3InitInstance（）：检查系统资源 97
 - 5.3.4检查游戏的多个实例 98
 - 5.3.5检查硬盘空间 99
 - 5.3.6检查内存 99
 - 5.3.7计算CPU速度 100
 - 5.3.8你拥有的是个垃圾袋吗 101
 - 5.3.9初始化资源缓存 101
 - 5.3.10加载文本字符串 102
 - 5.3.11脚本管理器和事件系统 104
 - 5.3.12初始化DirectX并创建窗口 104
 - 5.3.13创建游戏逻辑和游戏视图 105
 - 5.3.14设置游戏保存目录 105
 - 5.3.15预加载从缓存中选定的资源 106
 - 5.4收尾工作：干净漂亮地退出 107
 - 5.4.1我怎样才能离开呢 107
 - 5.4.2强制关闭模态对话框 109
 - 5.4.3关闭游戏 110
 - 5.4.4游戏机怎么样 110
 - 5.5进入和退出 111
- 第6章 游戏主体和组件架构 112
 - 6.1初次尝试创建游戏主体 112
 - 6.2组件架构 115
 - 6.3创建主体和组件 116
 - 6.4定义主体和组件 120
 - 6.5存储并访问主体 122
 - 6.6将它们组合起来 123
 - 6.7数据共享 124
 - 6.7.1直接访问 125

- 6.7.2事件 125
- 6.7.3两全其美 126
- 第7章 主循环的控制 127
 - 7.1组织主循环 127
 - 7.1.1硬编码的更新 127
 - 7.1.2多线程主循环 128
 - 7.1.3一种混合技术 129
 - 7.1.4简单的协同式多任务处理器 131
 - 7.1.5非常简单的进程示例：DelayProcess 135
 - 7.1.6Process派生类的使用 137
 - 7.2良好地适应操作系统 137
 - 7.3使用DirectX11框架 138
 - 7.3.1渲染和呈现显示 138
 - 7.3.2用于更新和渲染的回调函数 139
 - 7.4我现在可以制作游戏了吗 141
- 第8章 游戏数据的加载与缓存 142
 - 8.1游戏资源：格式和存储要求 143
 - 8.1.13D对象网格和环境 143
 - 8.1.2动画数据 145
 - 8.1.3地图／关卡数据 146
 - 8.1.4纹理数据 146
 - 8.1.5位图颜色深度 147
 - 8.1.6声音和音乐数据 149
 - 8.1.7视频和预渲染的过场动画 150
 - 8.2资源文件 152
 - 8.2.1将资源打包到一个文件中 153
 - 8.2.2打包资源的其他好处 153
 - 8.2.3数据压缩和性能 154
 - 8.2.4Zlib：开源压缩 154
 - 8.3资源高速缓存 158
 - 8.3.1ResourceFile接口 161
 - 8.3.2ResHandle：跟踪加载的资源 161
 - 8.3.3ResourceLoader接口和DefaultResourceLoader 163
 - 8.3.4ResCache：简单的资源高速缓存 163
 - 8.3.5将资源缓存入DirectX等 169
 - 8.3.6世界设计和高速缓存预测 170
 - 8.4我的缓存不够用了 173
- 第9章 输入设备编程 174
 - 9.1获取设备状态 174
 - 9.2使用XInput或DirectInput 177
 - 9.3一些安全提示 179
 - 9.4使用双轴控制器 182
 - 9.4.1捕获桌面上的鼠标 182
 - 9.4.2使用鼠标拖拽 184
 - 9.5使用游戏控制器 186
 - 9.5.1非灵敏区 187
 - 9.5.2正常输入 189
 - 9.5.3单杆、双杆、红色拉杆和蓝色拉杆 190
 - 9.5.4增加控制值 190
 - 9.6使用键盘 191
 - 9.6.1Mike的键盘窥探器 191
 - 9.6.2GetAsyncKeyState () 和其他函数 195
 - 9.6.3处理Windows中的Alt键 195
 - 9.7什么？没有跳舞毯 195

- 第10章 用户界面编程 197
 - 10.1 DirectX的文本助手和对话框资源管理器 197
 - 10.2 人类的游戏视图 198
 - 10.3 WASD移动控制器 206
 - 10.4 屏幕元素 208
 - 10.5 自定义的MessageBox对话框 210
 - 10.6 模态对话框 215
 - 10.7 控件 218
 - 10.8 控件识别 219
 - 10.9 命中测试和焦点顺序 221
 - 10.10 控件状态 222
 - 10.11 更多控件属性 223
 - 10.11.1 热键 223
 - 10.11.2 工具提示 223
 - 10.11.3 上下文相关帮助 224
 - 10.11.4 可拖拽 224
 - 10.11.5 声音和动画 224
 - 10.12 最后的用户界面提示 225
- 第11章 游戏事件管理 226
 - 11.1 游戏事件 226
 - 11.1.1 事件和事件数据 227
 - 11.1.2 事件监听器委托 230
 - 11.1.3 事件管理器 231
 - 11.1.4 示例：将所有内容整合在一起 238
 - 11.2 哪些游戏事件是重要的 239
 - 11.3 事件和进程的区别 241
 - 11.4 补充书目 241
- 第12章 使用Lua编写脚本 242
 - 12.1 游戏编程语言的简史 242
 - 12.1.1 汇编语言 243
 - 12.1.2 C/C++ 244
 - 12.1.3 脚本语言 245
 - 12.2 使用脚本语言 246
 - 12.2.1 快速原型法 246
 - 12.2.2 专注于设计 247
 - 12.2.3 速度和内存成本 247
 - 12.2.4 它们之间的界限是什么 247
 - 12.3 脚本语言集成策略 248
 - 12.3.1 自己进行编写 248
 - 12.3.2 使用现有的语言 248
 - 12.3.3 选择一种脚本语言 249
 - 12.3.4 Python 249
 - 12.3.5 Lua 249
 - 12.4 Lua速成课程 250
 - 12.4.1 注释 250
 - 12.4.2 变量 250
 - 12.4.3 函数 252
 - 12.4.4 表 253
 - 12.4.5 流程控制 255
 - 12.4.6 操作符 257
 - 12.4.7 接下来是什么 257
 - 12.5 Lua中的面向对象编程 258
 - 12.5.1 元表 259
 - 12.5.2 创建一个简单的类抽象 261

- 12.6内存管理 263
- 12.7将Lua绑定到C++ 263
 - 12.7.1LuaC API 263
 - 12.7.2tolua++ 263
 - 12.7.3luabind 264
 - 12.7.4LuaPlus 264
- 12.8LuaPlus速成课程 264
 - 12.8.1LuaState 264
 - 12.8.2LuaObject 265
 - 12.8.3表 266
 - 12.8.4全局 267
 - 12.8.5函数 268
 - 12.8.6从Lua调用C++函数 269
- 12.9将所有内容整合在一起 271
 - 12.9.1管理Lua状态 271
 - 12.9.2脚本导出 273
 - 12.9.3进程系统 274
 - 12.9.4事件系统 282
 - 12.9.5脚本组件 287
- 12.10Lua开发和调试 289
- 12.11结语 289
- 12.12补充书目 289
- 第13章 游戏音频 290
 - 13.1声音的工作原理 290
 - 13.1.1数字录音和重现 291
 - 13.1.2声音文件 293
 - 13.1.3线程和同步的简介 293
 - 13.2游戏语音系统架构 294
 - 13.2.1声音资源和句柄 295
 - 13.2.2IAudioBuffer接口和AudioBuffer类 303
 - 13.2.3IAudio接口和Audio类 305
 - 13.2.4DirectSound实现 308
 - 13.2.5声音进程 317
 - 13.2.6启动音效 321
 - 13.3其他技术难题 322
 - 13.3.1声音和游戏对象 322
 - 13.3.2定时和同步 322
 - 13.3.3混合问题 324
 - 13.4一些随记 326
 - 13.4.1数据驱动的声音设置 326
 - 13.4.2背景环境声音和音乐 327
 - 13.4.3语音 328
 - 13.5结语 330
- 第14章 3D图形基础 331
 - 14.13D图形流水线 331
 - 14.23D数学101332
 - 14.2.1坐标和坐标系 333
 - 14.2.2向量数学 335
 - 14.3C++数学类 340
 - 14.3.1向量类 340
 - 14.3.2矩阵数学 341
 - 14.3.3四元数数学 351
 - 14.3.4变换 358
 - 14.3.5几何体 360

- 14.3.6光照、法线和颜色 361
- 14.3.7材质 363
- 14.3.8贴有纹理的顶点 365
- 14.3.9纹理 365
- 14.3.10二次采样 365
- 14.3.11mip映射 367
- 14.3.12ID3D11Device和ID3D11DeviceContext简介 367
- 14.3.13在D3D11中加载纹理 368
- 14.3.14三角形网格 370
- 14.4你还在吗 373
- 第15章 3D顶点和像素着色器 374
- 15.1顶点着色器和着色器语法 375
- 15.2编译顶点着色器 379
- 15.3顶点着色器的C++辅助类 380
- 15.4像素着色器 386
- 15.5像素着色器的C++辅助类 387
- 15.6使用着色器辅助类进行渲染 390
- 15.7着色器—这只是一个开始 391
- 15.8补充书目 391
- 第16章 3D场景 392
- 16.1场景图基础 392
- 16.1.1ISceneNode接口类 392
- 16.1.2SceneNodeProperties和RenderPass 394
- 16.1.3SceneNode—一切都是从这里开始的 396
- 16.1.4Scene类 401
- 16.2特殊的场景图节点 408
- 16.2.1独立渲染通道的实现 408
- 16.2.2一个简单的摄像机 411
- 16.2.3在场景中放入灯光 413
- 16.2.4天空的渲染 416
- 16.2.5在场景中使用网格 420
- 16.3遗漏的内容 424
- 16.4还没满足 425
- 16.5补充书目 425
- 第17章 碰撞和简单的物理学 426
- 17.1物理学中的数学知识 427
- 17.1.1米、英尺、肘尺还是Kellicam 427
- 17.1.2距离、速度和加速度 427
- 17.1.3质量、加速度和力 428
- 17.1.4转动惯量、角速度和扭矩 431
- 17.1.5距离和交集的计算 431
- 17.2选择一种物理SDK 432
- 17.3对象属性 434
- 17.4碰撞体 435
- 17.4.1良好的碰撞几何体的要求 436
- 17.4.2可见几何体VS碰撞几何体 437
- 17.4.3人类角色的碰撞体 437
- 17.4.4特殊对象：楼梯、门道和树 439
- 17.5碰撞系统的使用 439
- 17.6集成一个物理SDK 441
- 17.6.1BulletSDK的组件 444
- 17.6.2初始化 445
- 17.6.3关闭 446
- 17.6.4物理系统的更新 447

- 17.6.5创建简单的物理对象 449
- 17.6.6凸面网格的创建 451
- 17.6.7触发器的创建 452
- 17.6.8力和力矩的应用 453
- 17.6.9物理调试渲染器 454
- 17.6.10接收碰撞事件 455
- 17.6.11物理SDK集成的最后内容 457
- 17.7等一下，我还有话要说 458
- 第18章 游戏AI简介 459
 - 18.1AI技术 459
 - 18.1.1硬编码AI 460
 - 18.1.2随机化 461
 - 18.1.3加权随机 462
 - 18.2有限状态机 463
 - 18.3决策树 467
 - 18.4模糊逻辑 471
 - 18.5效用理论 474
 - 18.6以目标为导向的行动计划 477
 - 18.7路径查找 478
 - 18.7.1A* (A—Star) 479
 - 18.7.2动态规避 481
 - 18.8补充书目 482
- 第19章 多玩家游戏的网络编程 483
 - 19.1互联网的工作原理 483
 - 19.1.1Winsock还是Berkeley 484
 - 19.1.2Internet地址 484
 - 19.1.3域名系统 486
 - 19.1.4有用的程序和文件 487
 - 19.2套接字API 488
 - 19.2.1套接字效用函数 488
 - 19.2.2域名服务 (DNS) 函数 490
 - 19.2.3套接字初始化和关闭 491
 - 19.2.4创建套接字和设置套接字选项 491
 - 19.2.5服务器函数 495
 - 19.2.6套接字读取和写入 498
 - 19.3使用套接字制作一款多玩家游戏 499
 - 19.3.1数据包类 500
 - 19.3.2核心套接字类 501
 - 19.3.3用于监听的套接字类 506
 - 19.3.4套接字管理器类 508
 - 19.4核心客户端类 515
 - 19.5核心服务器端类 516
 - 19.6将套接字连接到事件系统中 517
 - 19.7如果真的这么简单就好了 522
- 第20章 多道程序设计简介 523
 - 20.1多道程序设计是什么 523
 - 20.2创建线程 525
 - 20.3进程同步 527
 - 20.3.1测试与置位、信号量和互斥 528
 - 20.3.2Windows临界区 528
 - 20.4有趣的线程问题 530
 - 20.5线程安全 531
 - 20.6GameCode4中的多线程类 531
 - 20.6.1RealtimeProcess类 532

- 20.6.2从实时进程发送事件 534
- 20.6.3接收实时进程中的事件 537
- 20.7Zip文件的后台解压缩 538
- 20.8进一步工作 540
- 20.9关于硬件 541
- 20.10关于未来 541
- 20.11补充书目 542
- 第21章 “茶壶大战” 游戏 543
 - 21.1制作游戏 544
 - 21.2核心类的创建 545
 - 21.2.1茶壶大战的应用层 545
 - 21.2.2游戏逻辑 546
 - 21.2.3人类玩家的游戏视图 553
 - 21.3游戏事件 556
 - 21.4游戏玩法 556
 - 21.4.1关卡的加载 557
 - 21.4.2主体管理器 558
 - 21.4.3发送和接收事件 560
 - 21.4.4进程 562
 - 21.5留给读者的练习 563
- 第22章 C#中简单的游戏编辑器 565
 - 22.1为什么要使用C# 565
 - 22.2如何将编辑器组合起来 565
 - 22.3编辑器架构 566
 - 22.3.1应用层 566
 - 22.3.2编辑器的逻辑类 567
 - 22.3.3编辑器视图 568
 - 22.3.4访问游戏引擎的函数 569
 - 22.3.5创建DLL 578
 - 22.3.6编辑器架构的封装 578
 - 22.4C#编译器应用程序 579
 - 22.4.1托管代码和非托管代码之间的区别 580
 - 22.4.2NativeMethods类 581
 - 22.4.3Program类 582
 - 22.4.4MessageHandler类 583
 - 22.5C#编辑器用户界面 585
 - 22.5.1EditorForm类 585
 - 22.5.2ActorComponentEditor类 595
 - 22.6后续工作 603
 - 22.7补充材料 604
- 第23章 对游戏进行调试和分析 605
 - 23.1处理错误的艺术 606
 - 23.2调试基础 607
 - 23.2.1调试器的使用 609
 - 23.2.2安装Windows符号文件 611
 - 23.2.3对全屏游戏进行调试 612
 - 23.2.4远程调试 613
 - 23.2.5对小存储器转储文件（Minidump）进行调试 615
 - 23.3图形调试和着色器调试 616
 - 23.4调试技术 617
 - 23.4.1调试是一次实验 617
 - 23.4.2重现bug 619
 - 23.4.3降低复杂度 620
 - 23.4.4设置下一条语句 620

- 23.4.5汇编级调试 621
- 23.4.6给代码添加调料 623
- 23.4.7提取调试信息 624
- 23.4.8Lint和其他代码分析器 625
- 23.4.9Nu—Mega的BoundsChecker和运行时分析器 625
- 23.4.10消失的bug 625
- 23.4.11调整数值 626
- 23.4.12caveman调试 626
- 23.4.13当一切方法都失败时 627
- 23.5创建错误日志系统 628
- 23.6不同类型的bug 633
 - 23.6.1内存泄漏和堆损坏 634
 - 23.6.2游戏数据损坏 637
 - 23.6.3堆栈损坏 638
 - 23.6.4剪切和粘贴bug 639
 - 23.6.5空间不足 639
 - 23.6.6只在发布模式（ReleaseMode）中出现的bug 640
 - 23.6.7惹是生非的多线程 640
 - 23.6.8奇怪的bug 641
- 23.7性能分析 642
 - 23.7.1性能的测量 642
 - 23.7.2代码的优化 642
 - 23.7.3折中方案 643
 - 23.7.4过度优化 644
- 23.8结束小思 644
- 23.9补充书目 644
- 第24章 驶向结束 645
 - 24.1问题的整理 645
 - 24.1.1质量 646
 - 24.1.2代码 650
 - 24.1.3内容 653
 - 24.2应付大麻烦 655
 - 24.2.1项目进度严重拖延 655
 - 24.2.2人事相关问题 661
 - 24.2.3竞争对手会置你于死地 662
 - 24.2.4到底有没有出路 663
 - 24.2.5最后一个建议：不要惊慌 664
 - 24.3光明就在前方—毕竟这不是一场训练 664
 - 24.3.1测试存档 664
 - 24.3.2补丁build或产品演示 665
 - 24.3.3事后分析 665
 - 24.3.4如何利用你的时间 666
- • • • • ([收起](#))

[游戏编程权威指南_下载链接1](#)

标签

游戏编程

游戏

游戏开发

编程

软件开发

计算机

DirectX

Lua

评论

基于 Low-level 基础库 (Windows API, DirectX, Lua, .NET WinForms) 的 Windows 游戏开发的编程技术、程序设计和软件工程实践。无愧于 Windows 游戏开发领域的 Coding Complete 之称。

这是我读过的最好的游戏编程书，

[游戏编程权威指南_下载链接1](#)

书评

刚看这本书是在一年前，刚接触游戏开发，在图书馆看到这本书以为是那种overview的书，所以借来了一读，然后发现至少50%的内容读不懂……后来做了一些小项目之后再次重读，才发现这本书原来如此有价值。
这本书写的主要是作者在做《创世纪》游戏的过程中积累下的点滴经验和感悟...

[2018.10 更新] 这里的评论仅针对本书第一版[《游戏编程全接触》(2006)] ([Game Coding Complete, 1Ed (2003)])。本书的第四版[《游戏编程权威指南》(2016)] ([Game Coding Complete, 4Ed (2012)]) 采用的基础库更新到 D3D11 及同时代的其它技术 (XInput, DirectSound [replace...

很早看过这本书，好像还是英文版的，很多都忘记了，印象最深的好像是cache那一节，因为我的第一个游戏就是把所有图片都载入内存中，当时好像用了100多兆内存，然后放在别人的机器上，速度就很慢，我的开发机器是256兆，别人的机器是128的，哈哈。当时很迷茫，因为刚接触编程， ...

我读的是第四版的翻译版。
原书介绍了游戏开发的方方面面，虽然内容有些旧，但仍是作者开发的切身经验，有很高的参考价值。内容大致是一个概览，说了游戏开发中一些常用技术的基本概念和注意事项，不算太深入。至于风格，就是废话特别多。比如作者除了描述主题之外，还加了一些...

我对本书[第一版的评论]是在略读第一版[《游戏编程全接触》(2006)] ([Game Coding Complete, 1Ed (2003)]) 后写的。在读过本书的第四版[《游戏编程权威指南》(2016)] ([Game Coding Complete, 4Ed (2012)]) 后，感觉原评论已不合适，所以第一版评论仅针对本书第一版。 作者 Mr....

书是好书，很多地方都讲了，但歪个题，作者附赠的源码严重过时了，里面的库都是vs 2012了，第三方lib无法直接参与编译，要重新编译，但其中的luaplus无法生成vs2019的sln，github上提供的最新源码也无法生成项目文件，已经有issue但无回复，effects框架的源码无法编译，语法过时...

这本书是写游戏，更是写一个程序开发的全过程，并且不是空谈，而是实践者写就的。游戏开发需要注意的问题，包括GUI的，是其它可视软件都需要考虑到的。对整理开发思路也很有用处。

[游戏编程权威指南 下载链接1](#)