

《设计模式与游戏完美开发》

图书基本信息

书名：《设计模式与游戏完美开发》

13位ISBN编号：9787302455988

出版时间：2017-1-1

作者：蔡升达

页数：476

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu111.com

《设计模式与游戏完美开发》

内容概要

初次接触设计模式（Design Patterns）是在求学阶段，第一次看GoF的《Design Patterns: Elements of Reusable Object-Oriented Software》时，感觉犹如天书一般，只能大概了解Singleton、Strategy、Facade、Iterator这几个Pattern的用法，至于为什么要使用、什么时候使用，完全没有概念。

进入职场后，先是跟着几个大型游戏项目一同开发和学习的，到后来，自己可以主持技术项目、开发网络游戏引擎、游戏框架等。在这个过程中，时而拿起GoF的《Design Patterns》或是以设计模式为题的书籍，反复阅读，逐渐地了解了每一种模式的应用以及它们的设计分析原理，并通过不断地实践与应用，才将它们融入自己的知识体系中。

从2004年进入职场，一晃眼，在游戏业也超过了10年的经历，这些年在游戏行业工作的付出，除了得以温饱之外，也从中吸收了不少的知识与经验。记得某天在一个项目开发会议中，我与同仁分享如何将设计模式应用在游戏的开发设计中时，我突然察觉，应该将这些内容写下来，并分享给更多的游戏设计师，于是就有了写这本书的想法。

通过写作将经验与大家分享，希望大家可以了解，在游戏行业中的工程师，不该只是进行着“无意义”程序代码输出的“码农”，而是一群从事高级软件分析实现的设计师。所以，整合多种领域知识于一身的游戏工程师，更需要以优雅的方式来呈现这些知识汇集的结果，设计模式（Design Patterns）是各种软件设计技巧的呈现方式，善用它们，更能表现出游戏设计工程师优雅的一面。

10年的游戏从业过程，接受过许多人的协助及帮忙：Jimmy & Silent兄弟——20年的同学、朋友及合作伙伴们，有你们一路的协助与砥砺才能有今天；Justin Lee——谢谢你的信任，也感谢你的忍受功力，可以让我们一同完成不少作品；Mark Tsai——谢谢你一路的提拔与信任；Jazzdog——感谢你的支持，我一直知道程序与美术是可以同时存在于一个人身上的；Kai——合作伙伴，感谢你的支持。

最后谢谢我的家人，感谢老婆大人这10多年来忍受我在书房内不断地堆积书本、小说及收藏品。感谢我3岁的女儿，因为你的到来，让我知道没什么比你更重要了。

蔡升达

2016年10月

书籍目录

第1篇 设计模式与游戏设计

第1章 游戏实现中的设计模式 2

- 1.1 设计模式的起源 2
- 1.2 软件的设计模式是什么？ 3
- 1.3 面向对象设计中常见的设计原则 4
- 1.4 为什么要学习设计模式 7
- 1.5 游戏程序设计与设计模式 8
- 1.6 模式的应用与学习方式 10
- 1.7 结论 11

第2章 游戏范例说明 12

- 2.1 游戏范例 12
- 2.2 GoF的设计模式范例 15

第2篇 基础系统

第3章 游戏场景的转换——状态模式（State） 20

- 3.1 游戏场景 20
 - 3.1.1 场景的转换 20
 - 3.1.2 游戏场景可能的实现方式 23
- 3.2 状态模式（State） 24
 - 3.2.1 状态模式（State）的定义 24
 - 3.2.2 状态模式（State）的说明 25
 - 3.2.3 状态模式（State）的实现范例 25
- 3.3 使用状态模式（State）实现游戏场景的转换 28
 - 3.3.1 SceneState的实现 28
 - 3.3.2 实现说明 29
 - 3.3.3 使用状态模式（State）的优点 35
 - 3.3.4 游戏执行流程及场景转换说明 36
- 3.4 状态模式（State）面对变化时 37
- 3.5 结论 37

第4章 游戏主要类——外观模式（Facade） 39

- 4.1 游戏子功能的整合 39
- 4.2 外观模式（Facade） 41
 - 4.2.1 外观模式（Facade）的定义 41
 - 4.2.2 外观模式（Facade）的说明 42
 - 4.2.3 外观模式（Facade）的实现说明 43
- 4.3 使用外观模式（Facade）实现游戏主程序 44
 - 4.3.1 游戏主程序架构设计 44
 - 4.3.2 实现说明 45
 - 4.3.3 使用外观模式（Facade）的优点 47
 - 4.3.4 实现外观模式（Facade）时的注意事项 48
- 4.4 外观模式（Facade）面对变化时 48
- 4.5 结论 48

第5章 获取游戏服务的唯一对象——单例模式（Singleton） 50

- 5.1 游戏实现中的唯一对象 50
- 5.2 单例模式（Singleton） 51
 - 5.2.1 单例模式（Singleton）的定义 51
 - 5.2.2 单例模式（Singleton）的说明 51
 - 5.2.3 单例模式（Singleton）的实现范例 52

- 5.3 使用单例模式 (Singleton) 获取唯一的游戏服务对象 53
 - 5.3.1 游戏服务类的单例模式实现 53
 - 5.3.2 实现说明 54
 - 5.3.3 使用单例模式 (Singleton) 后的比较 55
 - 5.3.4 反对使用单例模式 (Singleton) 的原因 55
- 5.4 少用单例模式 (Singleton) 时如何方便地引用到单一对象 58
- 5.5 结论 63
- 第6章 游戏内各系统的整合——中介者模式 (Mediator) 64
 - 6.1 游戏系统之间的沟通 64
 - 6.2 中介者模式 (Mediator) 68
 - 6.2.1 中介者模式 (Mediator) 的定义 69
 - 6.2.2 中介者模式 (Mediator) 的说明 69
 - 6.2.3 中介者模式 (Mediator) 的实现范例 69
 - 6.3 中介者模式 (Mediator) 作为系统之间的沟通接口 72
 - 6.3.1 使用中介者模式 (Mediator) 的系统架构 73
 - 6.3.2 实现说明 73
 - 6.3.3 使用中介者模式 (Mediator) 的优点 79
 - 6.3.4 实现中介者模式 (Mediator) 时的注意事项 79
 - 6.4 中介者模式 (Mediator) 面对变化时 80
 - 6.5 结论 80
- 第7章 游戏的主循环——Game Loop 82
 - 7.1 GameLoop由此开始 82
 - 7.2 怎么实现游戏循环 (Game Loop) 84
 - 7.3 在Unity3D中实现游戏循环 85
 - 7.4 P级阵地的游戏循环 89
 - 7.5 结论 92
- 第3篇 角色的设计
- 第8章 角色系统的设计分析 94
 - 8.1 游戏角色的架构 94
 - 8.2 角色类的规划 95
- 第9章 角色与武器的实现——桥接模式 (Bridge) 98
 - 9.1 角色与武器的关系 98
 - 9.2 桥接模式 (Bridge) 103
 - 9.2.1 桥接模式 (Bridge) 的定义 103
 - 9.2.2 桥接模式 (Bridge) 的说明 107
 - 9.2.3 桥接模式 (Bridge) 的实现范例 108
 - 9.3 使用桥接模式 (Bridge) 实现角色与武器接口 110
 - 9.3.1 角色与武器接口设计 110
 - 9.3.2 实现说明 111
 - 9.3.3 使用桥接模式 (Bridge) 的优点 116
 - 9.3.4 实现桥接模式 (Bridge) 的注意事项 116
 - 9.4 桥接模式 (Bridge) 面对变化时 116
 - 9.5 结论 117
- 第10章 角色属性的计算——策略模式 (Strategy) 118
 - 10.1 角色属性的计算需求 118
 - 10.2 策略模式 (Strategy) 121
 - 10.2.1 策略模式 (Strategy) 的定义 122
 - 10.2.2 策略模式 (Strategy) 的说明 122
 - 10.2.3 策略模式 (Strategy) 的实现范例 123

- 10.3 使用策略模式 (Strategy) 实现攻击计算 124
 - 10.3.1 攻击流程的实现 125
 - 10.3.2 实现说明 125
 - 10.3.3 使用策略模式 (Strategy) 的优点 132
 - 10.3.4 实现策略模式 (Strategy) 时的注意事项 133
- 10.4 策略模式 (Strategy) 面对变化时 134
- 10.5 结论 135
- 第11章 攻击特效与击中反应——模板方法模式 (Template Method) 137
 - 11.1 武器的攻击流程 137
 - 11.2 模板方法模式 (Template Method) 139
 - 11.2.1 模板方法模式 (Template Method) 的定义 139
 - 11.2.2 模板方法模式 (Template Method) 的说明 141
 - 11.2.3 模板方法模式 (Template Method) 的实现范例 141
 - 11.3 使用模板方法模式实现攻击与击中流程 142
 - 11.3.1 攻击与击中流程的实现 143
 - 11.3.2 实现说明 143
 - 11.3.3 运用模板方法模式 (Template Method) 的优点 145
 - 11.3.4 修改击中流程的实现 145
 - 11.4 模板方法模式 (Template Method) 面对变化时 147
 - 11.5 结论 149
- 第12章 角色AI——状态模式 (State) 150
 - 12.1 角色的AI 150
 - 12.2 状态模式 (State) 158
 - 12.3 使用状态模式 (State) 实现角色AI 159
 - 12.3.1 角色AI的实现 159
 - 12.3.2 实现说明 160
 - 12.3.3 使用状态模式 (State) 的优点 169
 - 12.3.4 角色AI执行流程 169
 - 12.4 状态模式 (State) 面对变化时 170
 - 12.5 结论 172
- 第13章 角色系统 174
 - 13.1 角色类 174
 - 13.2 游戏角色管理系统 176
- 第4篇 角色的产生
- 第14章 游戏角色的产生——工厂方法模式 (Factory Method) 183
 - 14.1 产生角色 183
 - 14.2 工厂方法模式 (Factory Method) 188
 - 14.2.1 工厂方法模式 (Factory Method) 的定义 188
 - 14.2.2 工厂方法模式 (Factory Method) 的说明 189
 - 14.2.3 工厂方法模式 (Factory Method) 的实现范例 189
 - 14.3 使用工厂方法模式 (Factory Method) 产生角色对象 195
 - 14.3.1 角色工厂类 195
 - 14.3.2 实现说明 196
 - 14.3.3 使用工厂方法模式 (Factory Method) 的优点 199
 - 14.3.4 工厂方法模式 (Factory Method) 的实现说明 199
 - 14.4 工厂方法模式 (Factory Method) 面对变化时 203
 - 14.5 结论 205
- 第15章 角色的组装——建造者模式 (Builder) 206
 - 15.1 角色功能的组装 206

- 15.2 建造者模式 (Builder) 213
 - 15.2.1 建造者模式 (Builder) 的定义 213
 - 15.2.2 建造者模式 (Builder) 的说明 214
 - 15.2.3 建造者模式 (Builder) 的实现范例 215
- 15.3 使用建造者模式 (Builder) 组装角色的各项功能 217
 - 15.3.1 角色功能的组装 218
 - 15.3.2 实现说明 219
 - 15.3.3 使用建造者模式 (Builder) 的优点 226
 - 15.3.4 角色建造者的执行流程 226
- 15.4 建造者模式 (Builder) 面对变化时 227
- 15.5 结论 228
- 第16章 游戏属性管理功能——享元模式 (Flyweight) 229
 - 16.1 游戏属性的管理 229
 - 16.2 享元模式 (Flyweight) 236
 - 16.2.1 享元模式 (Flyweight) 的定义 236
 - 16.2.2 享元模式 (Flyweight) 的说明 237
 - 16.2.3 享元模式 (Flyweight) 的实现范例 238
 - 16.3 使用享元模式 (Flyweight) 实现游戏 242
 - 16.3.1 SceneState的实现 242
 - 16.3.2 实现说明 245
 - 16.3.3 使用享元模式 (Flyweight) 的优点 250
 - 16.3.4 享元模式 (Flyweight) 的实现说明 250
 - 16.4 享元模式 (Flyweight) 面对变化时 252
 - 16.5 结论 252
- 第5篇 战争开始
- 第17章 Unity3D的界面设计——组合模式 (Composite) 254
 - 17.1 玩家界面设计 254
 - 17.2 组合模式 (Composite) 259
 - 17.2.1 组合模式 (Composite) 的定义 259
 - 17.2.2 组合模式 (Composite) 的说明 260
 - 17.2.3 组合模式 (Composite) 的实现范例 261
 - 17.2.4 分了两个子类但是要使用同一个操作界面 264
 - 17.3 Unity3D游戏对象的分层式管理功能 265
 - 17.3.1 游戏对象的分层管理 265
 - 17.3.2 正确有效地获取UI的游戏对象 266
 - 17.3.3 游戏用户界面的实现 267
 - 17.3.4 兵营界面的实现 269
 - 17.4 结论 274
- 第18章 兵营系统及兵营信息显示 276
 - 18.1 兵营系统 276
 - 18.2 兵营系统的组成 277
 - 18.3 初始兵营系统 281
 - 18.4 兵营信息的显示流程 287
- 第19章 兵营训练单位——命令模式 (Command) 288
 - 19.1 兵营界面上的命令 288
 - 19.2 命令模式 (Command) 291
 - 19.2.1 命令模式 (Command) 的定义 291
 - 19.2.2 命令模式 (Command) 的说明 294
 - 19.2.3 命令模式 (Command) 的实现范例 294

- 19.3 使用命令模式 (Command) 实现兵营训练角色 297
 - 19.3.1 训练命令的实现 297
 - 19.3.2 实现说明 298
 - 19.3.3 执行流程 302
 - 19.3.4 实现命令模式 (Command) 时的注意事项 303
- 19.4 命令模式 (Command) 面对变化时 305
- 19.5 结论 306
- 第20章 关卡设计——责任链模式 (Chain of Responsibility) 307
 - 20.1 关卡设计 307
 - 20.2 责任链模式 (Chain of Responsibility) 312
 - 20.2.1 责任链模式 (Chain of Responsibility) 的定义 312
 - 20.2.2 责任链模式 (Chain of Responsibility) 的说明 314
 - 20.2.3 责任链模式 (Chain of Responsibility) 的实现范例 314
 - 20.3 使用责任链模式 (Chain of Responsibility) 实现关卡系统 317
 - 20.3.1 关卡系统的设计 317
 - 20.3.2 实现说明 318
 - 20.3.3 使用责任链模式 (Chain of Responsibility) 的优点 329
 - 20.3.4 实现责任链模式 (Chain of Responsibility) 时的注意事项 329
 - 20.4 责任链模式 (Chain of Responsibility) 面对变化时 330
 - 20.5 结论 332
- 第6篇 辅助系统
 - 第21章 成就系统——观察者模式 (Observer) 334
 - 21.1 成就系统 334
 - 21.2 观察者模式 (Observer) 338
 - 21.2.1 观察者模式 (Observer) 的定义 338
 - 21.2.2 观察者模式 (Observer) 的说明 340
 - 21.2.3 观察者模式 (Observer) 的实现范例 341
 - 21.3 使用观察者模式 (Observer) 实现成就系统 344
 - 21.3.1 成就系统的新架构 344
 - 21.3.2 实现说明 346
 - 21.3.3 使用观察者模式 (Observer) 的优点 358
 - 21.3.4 实现观察者模式 (Observer) 时的注意事项 358
 - 21.4 观察者模式 (Observer) 面对变化时 359
 - 21.5 结论 361
 - 第22章 存盘功能——备忘录模式 (Memento) 362
 - 22.1 存储成就记录 362
 - 22.2 备忘录模式 (Memento) 366
 - 22.2.1 备忘录模式 (Memento) 的定义 366
 - 22.2.2 备忘录模式 (Memento) 的说明 367
 - 22.2.3 备忘录模式 (Memento) 的实现范例 367
 - 22.3 使用备忘录模式 (Memento) 实现成就记录的保存 371
 - 22.3.1 成就记录保存的功能设计 371
 - 22.3.2 实现说明 371
 - 22.3.3 使用备忘录模式 (Memento) 的优点 374
 - 22.3.4 实现备忘录模式 (Memento) 的注意事项 374
 - 22.4 备忘录模式 (Memento) 面对变化时 374
 - 22.5 结论 375
 - 第23章 角色信息查询——访问者模式 (Visitor) 376
 - 23.1 角色信息的提供 376

- 23.2 访问者模式 (Visitor) 385
 - 23.2.1 访问者模式 (Visitor) 的定义 386
 - 23.2.2 访问者模式 (Visitor) 的说明 390
 - 23.2.3 访问者模式 (Visitor) 的实现范例 392
 - 23.3 使用访问者模式 (Visitor) 实现角色信息查询 397
 - 23.3.1 角色信息查询的实现设计 397
 - 23.3.2 实现说明 398
 - 23.3.3 使用访问者模式 (Visitor) 的优点 405
 - 23.3.4 实现访问者模式 (Visitor) 时的注意事项 405
 - 23.4 访问者模式 (Visitor) 面对变化时 405
 - 23.5 结论 408
- 第7篇 调整与优化
- 第24章 前缀后缀—装饰模式 (Decorator) 410
 - 24.1 前缀后缀系统 410
 - 24.2 装饰模式 (Decorator) 415
 - 24.2.1 装饰模式 (Decorator) 的定义 415
 - 24.2.2 装饰模式 (Decorator) 的说明 418
 - 24.2.3 装饰模式 (Decorator) 的实现范例 419
 - 24.3 使用装饰模式 (Decorator) 实现前缀后缀的功能 422
 - 24.3.1 前缀后缀功能的架构设计 423
 - 24.3.2 实现说明 423
 - 24.3.3 使用装饰模式 (Decorator) 的优点 433
 - 24.3.4 实现装饰模式 (Decorator) 时的注意事项 433
 - 24.4 装饰模式 (Decorator) 面对变化时 434
 - 24.5 结论 435
 - 第25章 俘兵—适配器模式 (Adapter) 436
 - 25.1 游戏的宠物系统 436
 - 25.2 适配器模式 (Adapter) 440
 - 25.2.1 适配器模式 (Adapter) 的定义 440
 - 25.2.2 适配器模式 (Adapter) 的说明 441
 - 25.2.3 适配器模式 (Adapter) 的实现范例 441
 - 25.3 使用适配器模式 (Adapter) 实现俘兵系统 443
 - 25.3.1 俘兵系统的架构设计 443
 - 25.3.2 实现说明 443
 - 25.3.3 与俘兵相关的新增部分 445
 - 25.3.4 使用适配器模式 (Adapter) 的优点 450
 - 25.4 适配器模式 (Adapter) 面对变化时 450
 - 25.5 结论 451
 - 第26章 加载速度的优化—代理模式 (Proxy) 453
 - 26.1 最后的系统优化 453
 - 26.2 代理模式 (Proxy) 457
 - 26.2.1 代理模式 (Proxy) 的定义 458
 - 26.2.2 代理模式 (Proxy) 的说明 458
 - 26.2.3 代理模式 (Proxy) 的实现范例 459
 - 26.3 使用代理模式 (Proxy) 测试和优化加载速度 460
 - 26.3.1 优化加载速度的架构设计 460
 - 26.3.2 实现说明 461
 - 26.3.3 使用代理模式 (Proxy) 的优点 464
 - 26.3.4 实现代理模式 (Proxy) 时的注意事项 464

26.4 代理模式 (Proxy) 面对变化时 466

26.5 结论 466

第8篇 未明确使用的模式

第27章 迭代器模式 (Iterator)、原型模式 (Prototype) 和解释器模式 (Interpreter) 468

27.1 迭代器模式 (Iterator) 468

27.2 原型模式 (Prototype) 469

27.3 解释器模式 (Interpreter) 471

第28章 抽象工厂模式 (Abstract Factory) 472

28.1 抽象工厂模式 (Abstract Factory) 的定义 472

28.2 抽象工厂模式 (Abstract Factory) 的实现 473

28.3 可应用抽象工厂模式的场合 476

参考文献 477

《设计模式与游戏完美开发》

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu111.com