游戏编程模式 阅读记录

目录

[游戏编程模式 阅读记录 1](#_Toc502828014)

[设计模式 2](#_Toc502828015)

[命令模式 2](#_Toc502828016)

[享元模式 2](#_Toc502828017)

[观察者模式 2](#_Toc502828018)

[原型模式 2](#_Toc502828019)

[单例模式 2](#_Toc502828020)

[状态模式 2](#_Toc502828021)

[序列模式 2](#_Toc502828022)

[双缓冲 2](#_Toc502828023)

[游戏循环 2](#_Toc502828024)

[更新方法 2](#_Toc502828025)

[行为模式 2](#_Toc502828026)

[字节码 2](#_Toc502828027)

[子类沙盒 2](#_Toc502828028)

[对象模型 2](#_Toc502828029)

[解耦模式 3](#_Toc502828030)

[组件 3](#_Toc502828031)

[事件队列 3](#_Toc502828032)

[服务定位器 3](#_Toc502828033)

[优化模式 3](#_Toc502828034)

[数据 3](#_Toc502828035)

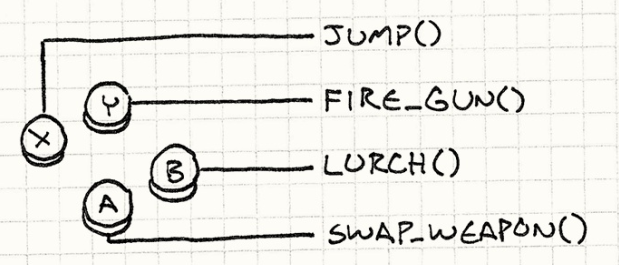
[脏标记 3](#_Toc502828036)

[对象池 3](#_Toc502828037)

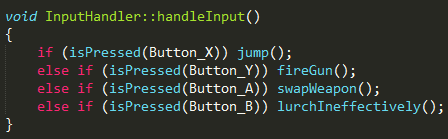
[空间划分 3](#_Toc502828038)

# 设计模式

## 命令模式



译为编码：

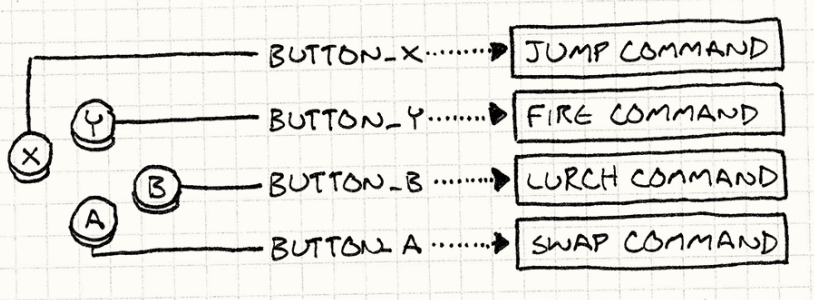


如果说玩家可以自定义配置，按键对应的功能，那么上面的构造显然不能够满足我们的要求。

反复思考

通过命令模式控制如何呢？

如下图：





## 享元模式

## 观察者模式

## 原型模式

## 单例模式

## 状态模式

# 序列模式

## 双缓冲

## 游戏循环

## 更新方法

# 行为模式

## 字节码

## 子类沙盒

## 对象模型

# 解耦模式

## 组件

## 事件队列

## 服务定位器

# 优化模式

## 数据

## 脏标记

## 对象池

## 空间划分