卡牌游戏

# 思路

* 卡牌不能复制，但是可以传承，它们具有生命力，生活在自己的世界
* 玩家和它们是伙伴关系，会亲密也会疏离
* 类似于一个宠物战斗游戏，但是制作成本很低
* 用类似于种子链接/比特币的方式共享这些卡牌信息；也可以自定义服务器，每个服务器就是一个独特的世界，有各自的世界观
* 卡牌自动化：设计一场事件“陷阱”，触发的时候，将随机的符合条件的卡牌人物拉进来，可以借此赋予新的关键词给他们；新世界有大量的事件等待触发，并且在不断的设计新的事件，设计新的古老剧情，将卡牌逐渐完善起来，将世界观逐渐丰富起来，每张卡牌都有其特殊的“基因”，以此作为触发媒介
* NPC和事件必须要加入游戏推进，并且给卡片添加属性，而不是由玩家添加
* 相对于3A游戏，卡牌的属性添加更加方便灵活，其故事不是固定的，并且具有生命力
* 故事编写员：基于遭遇编写（自动？）
* 基本玩法必须要有趣，不然后续展不开

# 命令行

# 架构与实现

# 炉石传说机制分析

两种实现方法，一种是在各个阶段，轮询所有卡牌，执行相关操作，类似于Win32编程，消息将发放给窗口消息处理函数。

WinProc()

case:

CustomProc();

break;

default:

DefWindowProc ();

另一种是注册机制，卡牌将自己的回调注册到对应的属性中，属性发生改变时执行。

前者是轮询机制，后者是中断机制。

我觉得第二种更符合设定，对属性敏感，并且理解起来更加直观。

注册的时候有优先级概念。

## 法术反制

法术施放包含以下步骤：

1. 法术进场，即位置发生改变，这是一条属性变化（法术反制在此阶段被检测并取消后续所有属性变化）
2. 法术生效

所以需要细化到每个属性发生改变。

## 博尔夫碎盾

每当你的英雄受到伤害。

**博尔夫碎盾**会修改**事件**内容，在匹配完成后，被触发。所以，匹配完成后，会有一次调整事件的机会。

如果匹配到是英雄，并且是血量属性减少，这个陷阱才会触发。

所以，细分到执行前，执行中和执行后。

## 舰长

下一怪，对所有其他随从造成一点伤害。

此事件在下场的随从执行完所有指令后触发的。

## 市长

所有选择的目标随机。

注意，是已经确定了的目标会发生改变。

## 疯狂的奴隶主

每受到一点伤害，召唤一个奴隶主。

注意，如果召唤数量超过限制，不再召唤。所以，召唤属于系统行为。召唤应该产生事件，让系统去做。**卡牌内部的消息处理仅仅用来发送新的消息。这个消息要放在开头，不能放在末尾**。