为了人类的和平，卡牌过程的逻辑设计会很重要。

现在我先将游戏简化到最简单。

假设卡牌只有两种：

具有a/b/c属性的白板随从

a费造成b点伤害的法术

那么，作为最终管理的黑箱

应当存储这些必要信息：

双方玩家有哪些手牌

双方玩家的最大水晶/当前水晶数

双方牌库有哪些牌

双方场面上有哪些随从，随从的属性如何。

这是第一步。

第二步则是

加入特效随从：战吼随从，亡语随从和光环随从。

第三步是加入普通法术卡（多种类型）

第四步加入奥秘

第五步加入英雄技能

第六步加入其它新机制…

好的，现在我们先来进行第一步的设计。

一件很重要的事情：

随从卡和随从是不一样的。嗯，这是句废话，但是很重要。

所以召唤随从的过程应该是这样的：

使用一张卡片->召唤一个随从，数据与卡片描述一致。

嗯，这样一来就可以解决闷棍的问题：闷一个随从回去等于使一个随从消失并且对方手牌重新获得该随从卡。

呃，还不一定有闷棍这类的卡呢。

随从可能的动作只有一个：攻击

一旦攻击，按照攻击计算规则进行结算即可。

关于这里，请注意，可以加入触发器（以应对各种触发式随从效果以及奥秘）

所以简单的trigger可以包括这么几类：

attack:攻击（在这里还有反向的被攻击）

summon：召唤

usecrad：使用卡片（使用随从/法术卡可以分类处理）

drawcard：抽取卡片

getcard：获得卡片（注意这两者的区别，某翻倍抽卡的龙有话说）

dead：随从死亡（以及反向的杀死随从）

turnstart：回合开始

turnend：回合结束

trigger是为了以后考虑的命名法。

现在的话请简单地当作trigger说的是啥就执行什么操作执行即可。

使用这种方式管理的话。

那么所有的玩家操作就会被归结为调用trigger，如果没有触发器的话就直接执行trigger事件，否则转入trigger指定的事件。

于是每个回合就会自动进入这样的循环：

turnstart->drawcard->getcard->（用户操作）->turnend

用户操作包括usecard,attack。

除此之外每个操作都是自动进行的。

用这种方法，只要trigger内部机制正确，第一步所请求的那种只有标准随从和标准法术的程序就完成了。

针对game类的设计：

成员变量：

Card playercard1[],playercard2[];//玩家手牌

Unit playerunit1[],playerunit2[];//玩家随从（提示：随从群中第0位是英雄）

Card playercardcollect1[],playercollect2[];//玩家牌库

成员函数：

构造函数与析构函数。

game类不会管理计算过程（因为game类严格来说是只具有trigger功能的）

Trigger的使用方式是

Trigger\_turnstart(game,\*\*\*)（其它变量，暂时未定）

在UI中被调用。

（例如，在UI\_gamemain中，玩家动了某个卡牌，就会执行Trigger\_usecard命令；按了回合结束按钮，就会执行Trigger\_turnend命令）

关于UI程序部分与draw部分的连结。

程序的其它部分采用默认的方法就好，添加纹理本身是有内置操作的….吧，应该。