二、分模块设计说明

(3) GamePlayScene设计说明

以下将分模块说明GamePlayScene各个子模块的程序设计。

**1 NowCardSet设计说明**

**1.1程序描述**

此类为衔接各个界面之间传输当前卡组信息的类，用于保存/读取当前卡组信息。

**1.2功能**

存储当前正在使用的卡组，并且负责读取已有卡组或将当前卡组写入文件。

**1.3性能**

读写数据应当快速高效。

**1.4输入项**

只需要输入一个string类，用于说明读取/存储的文件名。

**1.5输出项**

当读取文件时，会将读出的数据存储于该类的缓存区。

**1.6接口**

void LoadFile(string filename);

void SaveFile(string filename);

int[] nowcard;

**1.7限制条件**

该类成员均为静态成员

**1.8尚未解决的问题**

暂无特殊问题

**1 Netlink设计说明**

**1.1程序描述**

此类为负责网络数据交流的类，用于与远程用户就游戏数据进行交流。

**1.2功能**

负责与远程主机建立网络连接，以及传输数据。

**1.3性能**

采用阻塞的方式收发数据。因为网络的性能取决与网络环境，所以没有对时延的要求。

**1.4输入项**

在建立连接时，输入对方的IP地址以及端口号。

在发送数据时，输入NetMessage类（会自动解包为string并发送，在远端自动封包回NetMessage类）

**1.5输出项**

在接受数据时，会返回对手发送的NetMessage类

**1.6流程逻辑**



**1.7接口**

void Host(int port);

void Client(string address,int port);

void SendMSG(NetMessage input);

NetMessage RecvMSG();

**1.8限制条件**

本模块均为静态函数。

**1.9尚未解决的问题**

因为在等待对手的操作过程中使用同步的方式进行数据交流会在对手没有操作时完全卡死不能进行操作，所以有可能的话希望能通过异步的方式进行网络数据传输。但是这么做又会打破程序结构，不能很好地走好游戏流程，所以导致了一些问题。

**1 GameCenter设计说明**

**1.1程序描述**

此类保存一些游戏基本数据，例如双方用户的牌库的牌，当前法力值等。并且也负责游戏初始化，回合转换，解析对手发送的网络信息的功能。

**1.2功能**

分为两大类：存储游戏基础数据，例如双方的用户的牌库，当前的法力值与最大法力值。

以及负责游戏流程的运行，例如游戏开始，回合转换。并且在对手的回合，负责解析对手发送的网络信息，并且表示出其动作与结果。

**1.3性能**

无特殊性能要求，但是不能导致程序的卡顿和破坏游戏流程。

**1.4输入项**

没有需要输入的。

**1.5输出项**

没有需要输出的。

**1.6流程逻辑**

见整体流程图。

**1.8限制条件**

为游戏空对象GameCenter的MonoBehavior。

**1.9尚未解决的问题**

暂无自身问题。但是每次更新游戏功能时都需要在该类上面反映出来，所以很容易导致疏漏。

**1 CommonRandom设计说明**

**1.1程序描述**

此类为本游戏公用的随机类。

**1.2功能**

负责同步游戏双方的随机数种子，保证相同的随机序列，并且严格依此将各随机数提供给需要随机数的模块。

**1.3性能**

立刻返回。

**1.4输入项**

同步随机数种子时，需要给定指定的随机数种子。

获取随机数时，需要给出所要的随机数的范围。

**1.5输出项**

获取随机数时，返回一个整数，为指定范围内的一个随机数。

**1.7接口**

void init();

void init(int seed);

int random(int min,int max);

**1.8限制条件**

本模块均采用静态变量。

**1.9尚未解决的问题**

暂无。不过Unity操作系统很可能有默认的随机数种子，在测试运行时多次运行总是给出一样的结果。

**1 CardFactory设计说明**

**1.1程序描述**

此类为生成游戏卡牌而使用的工厂类。

**1.2功能**

负责用于生成卡片Object。输入所需卡片对应的编号，就能获得对应的卡片。

**1.3性能**

所有函数应当快速返回，不能发生阻塞或者死循环。

**1.4输入项**

只输入所需的卡片的编号

**1.5输出项**

返回已经生成的卡牌的GameObject

**1.7接口**

GetCard(int id);

**1.8限制条件**

本类为空对象CardFactory的MonoBehavior

**1.9尚未解决的问题**

想要添加卡片必须手动一个一个添加case条件。不过因为没有办法描述卡牌所带有的效果的脚本化（从而可以被解释执行），所以也只能采用笨重的方法。

（4）卡片设计说明

因为附着在卡片上的script为三种功能完全不重合的脚本，故以下将分别说明

**1 CardInfo设计说明**

**1.1程序描述**

此类为卡片数据类

**1.2功能**

负责存储卡片的所有信息，并且将数据update到程序的视图中。

**1.3性能**

所有函数应当快速返回，不能发生阻塞或者死循环。

**1.4输入项**

无输入项

**1.5输出项**

无输出项

**1.7接口**

void update();

**1.8限制条件**

本类为Card的一个MonoBehavior

**1.9尚未解决的问题**

无

**1 CardMove设计说明**

**1.1程序描述**

此类为卡片动作类。

**1.2功能**

在游戏事件发生时，负责使卡片在屏幕上做相应的动画。

**1.3性能**

使用协程进行程序。不会发生卡顿。

**1.4输入项**

输入动作的类型编号。

**1.5输出项**

无。

**1.7接口**

void DoMove(int MoveType);

**1.8限制条件**

此类为Card的一个MonoBehavior

**1.9尚未解决的问题**

卡片动画只能通过Unity特有的“协程”来绘制。（不能使用多线程）而Unity没法获得协程结束的通知，不知道协程何时结束，所以因为结算顺序的问题，必须要将卡片效果结算显式置于卡片动画的结尾，才能顺利做到先执行动画，再变化卡片数据。

**1 Trigger设计说明**

**1.1程序描述**

此类为卡片结算类。

**1.2功能**

完成所有卡片效果的结算。

**1.3性能**

所有函数应当快速返回，不能发生阻塞或者死循环。

**1.4输入项**

输入两个GameObject类，分别为事件的使用者和被使用者。

**1.5输出项**

没有输出，但是卡片数据会发生变化。

**1.7接口**

virtual void Exec(GameObject user,GameObject target);//提示：通过继承实现扩展

**1.8限制条件**

此类为Card的一个MonoBehavior

**1.9尚未解决的问题**

见CardMove的“尚未解决的问题”章节。