首先是在上次的类关系图的基础上进行优化：

1. 与玩家交互的功能应当都集成在Board这个类中，或者说，让他来充当屏幕的作用，来判断当前应当显示什么，玩家的操作输入应当如何解释、处理。其他所有类不应当能直接和真实玩家产生交互关系，减少复杂性。
2. 实体类有显示、产生动作、被动作影响的需求，一律交给Board类，一个动作应当是基础的、可以通过手数出来的，附加的参数不超过1个，其余更加多的效果通过Effect实体实现，专门负责效果处理。实体在定义时是在屏幕上有交互实现或者有随时间变化的活动的，应当统一提供显示、产生动作方法供Board调用，而各自的交互方法通过接口实现（可能无法交互，单纯产生动作，这样可以获得如门等机关陷阱的实现以及路障的实现）。
3. Map（地图）类和Term词条类实际在生成、使用的时候没有必要设计的如此复杂，他们都是数据，应当设计各项简单条目序列化后持久在文件系统中，需要时调出到内存中即可，无需复杂的在类内的生成、设计。
4. 增加两个接口：交互和展示。考虑到在菜单中仍然需要显示词条、菜单也需要可以交互（用户提供一个键鼠指令，程序提供一个反馈），而交互并不是实体所必须，可以抽象出一个接口。
5. 并行（并发）化：事实上图形的刷新、触发事件，尤其是NPC的行为设计，他们的行为不顺序依赖于任何其他事件，在直觉上应当是同步在运行的。这样，通过Board这个类维护一个消息（行为）队列传递他们之间的行为互动（Action数组），使其如同玩家一般，可以异步的向游戏桌面发送操作信号或者查看信号。于是设计一个触发器类来辅助其异步、延时或条件传递行为，在Board中维护这些类的实例队列。可以看到，事件与AI部分是一个类生产者-消费者的模型，玩家、AI生产事件，作为各种行为的发起方，同时他们又是消费者，消费产生的部分合法行为。相比于串行，这样必定会造成一定的损耗，例如AI进行了一次即刻的攻击，但可能由于队列问题并无法即刻被消费掉，但正常游戏应当有一秒钟至少20-30帧的刷新频率，表现为一秒钟进行相应次数的生产-批量消费，即使跨帧消费仍然是一个相对很短的时间所以可行。

得到进一步的类图（简化掉了部分简单依赖的关系防止混乱）和时序图

其中在核心流程上，实际在用例中已经基本说明了，只需对应到相应的类上即可。简单的选择了玩家按下攻击按键时的流程（略去简单的技能检索）、玩家在物品栏菜单中选择武器升级时的流程。