[可能的重构 2](#_Toc356767867)

[1. LayerManager改成抽象类，有多个子类。 2](#_Toc356767868)

[问题说明 2](#_Toc356767869)

[未进行的重构 3](#_Toc356767870)

[1. 可考虑使用桥接模式，将层Layer的表现与逻辑分离，形成LayerShow类和LayerLogic类 3](#_Toc356767871)

[重构前的设计 3](#_Toc356767872)

[重构思路 3](#_Toc356767873)

[进一步思考 3](#_Toc356767874)

[不进行重构的原因 4](#_Toc356767875)

## 可能的重构

1. LayerManager改成抽象类，有多个子类。

#### 问题说明

因为EnemyLayer实现了ICheckPlayer接口，因此可增加CheckPlayerLayerManager子类，把实现了该接口的层加入到该子类中。

而MapLayer、PlayerLayer因为方法个数相同，行为类似，可以加入到另一个子类（如CommonLayerManager）中。

1. 现在移动状态子类只有WalkState族，如果需要飞行（在判断方向时，不考虑地形），则可以增加FlyState状态族？

## 未进行的重构

1. 可考虑使用桥接模式，将层Layer的表现与逻辑分离，形成LayerShow类和LayerLogic类

#### 重构前的设计

Layer类族有不同层级的类。Layer继承于Collection，Layer有子类MapLayer、CharacterLayer，CHaracterLayer有子类PlayerLayer、EnemyLayer。

#### 重构思路

LayerLogic <- Layer -> LayerShow

Layer、LayerLogic、LayerShow都继承于Collection集合类，从而获得集合操作。

LayerShow类：

与调用canvas显示相关的操作（如setCanvas、draw、clear、getContext等）。

LayerLogic类：

状态的相关操作（change等）

精灵的移动、碰撞检测等。

Layer类：

render

提供外界调用的接口，如change（具体工作委托给LayerLogic或LayerShow类）。

#### 进一步思考

将Layer类与LayerLogic合并为LayerLogic类（类名仍然为Layer，后面就将合并后的类成为Layer类）！因为通过继承Collection类，可以获得集合的相关操作，所以集合的相关操作不用提出来形成Layer类！并且Layer类中单纯的委托接口方法（如change）也没有存在的必要！

此处有个问题！因为Layer、LayerShow都是集合类，所以都需要继承Collection类！且两者维护不同的childs数组！所以需要在Layer类获得LayerShow的实例时，将LayerShow的childs数组指向Layer的childs数组，使他们维护同一个childs数组！

#### 不进行重构的原因

* LayerShow中的draw、clear等方法，实际上就是调用层内元素的draw、clear方法，应该也是属于逻辑部分（即也应该放到LayerLogic中）！
* MapLayer中的双缓冲按照上面重构的思路，也应该放到LayerShow中，但是双缓冲应该是属于优化的职责，并不是表现的职责！这样看来，又可以提炼出LayerOptimize类，将双缓冲移到该类中了！
* 如果将setCanvas移到LayerShow中，也不是很适合！因为不同的层级z-index不同，所以需要setCanvas方法的设置canvas的z-index属性。  
  所以这样看来还是应该保持重构前的设计（即Layer类族应该有不同层级的类，即MapLayer、CharacterLayer类等（MapLayer层级为0，在最下面；Character层级为1，显示的时候覆盖MapLayer层）。不同层级的类实现setCanvas方法）。  
  如果采用桥接模式重构，setCanvas该如何处理？canvas的z-index该在此处设置？