1. 项目总体架构

我们将本项目分为前端实现和后端实现。前端实现主要负责加载各种图像，后端实现则负责读取、修改游戏配置文件以及玩家存档中的内容。

前端实现

前端实现又分为地图图层和UI图层，而两部分又使用一个单例类（SceneManager类）统一加载、管理。本项目中，每张瓦片地图都被视作一个独立的场景，而UI图层被添加至一个永久节点中，每次切换场景时，都会组装整张地图，并将地图图层和永久节点添加到场景中，假如要展示UI场景，可以通过调整UI图层的上下顺序实现。

后端实现

后端实现又分为时间管理器，主角类和各种地图对象。时间管理器是一个单例类，在加载存档时实例化，在卸载存档时析构。负责管理游戏场景内的所有时间，在进入下一天时调用结算函数，进行清算并存档。主角类是一个单例类，同样在加载存档时实例化，在卸载存档时析构。负责管理游戏中主角的所有信息的修改和存档的写入，包括：持有物品，精力，金钱等。各种地图对象都继承自一个MapObject类，该类是一个抽象接口类，该类中预留了各种接口函数，可以由地图图层统一管理、调用。

1. 游戏场景实现细节
2. 从文件创建游戏场景

我们将场景的具体信息写入文件（存档、配置），每个地图图层都对应一个独立的场景文件，该场景文件中包含了该地图图层的名字、瓦片地图、背景音乐、背景颜色等各项信息。

该文件还包含一个在该图层上的常驻对象列表；在存档中，还有一个可修改对象列表。对象列表中储存了所有对象的必要信息，包括位置、对象的种类以及每个对象的独立信息。

1. 地图逻辑常驻加载

在本项目中，地图图层的数据逻辑是常驻加载的，而图像只有在主角在对应场景时才会加载。每当存档加载时，SceneManager会创建所有地图图层（一个MapLayer类实例），在创建地图图层时，不会加载任何一个精灵，也不会读取任何一个图片，MapLayer只会从场景文件读取所有的逻辑信息并加载。这样的设计可以方便网络层修改数据，为后续开发预留了接口。

1. 抽象接口类与工厂函数方法创建

地图图层上的所有地图对象都继承自一个抽象接口类：MapObject，该类继承自cocos2d-x中的Ref类，方便使用自动释放池管理对象的生命周期。

当MapLayer创建时，都读取两个场景文件中的对象列表。

1. MapObject抽象类
2. UI实现细节
3. 后台数据实现细节
4. 存档管理
5. 文件系统

文件统一使用json作为储存格式

1. 项目分工
2. 地图前端
3. UI前端