设计理念

一、场景

场景统一管理：前台设计

由SceneManager统一加载、管理所有场景，将场景的具体配置写入文件，通过文件中的内容加载场景

二、物品

**1.继承关系**

引擎定义：Ref（基）

Note

Sprite

程序实现：每一个物品如耕地

有继承自Ref的Object

Object中有一个\*sprite

**2.调用时机+主要功能**

1. 启动游戏： 遍历数据库，Object对象实例化
2. 走入当前地图 Sprite对象实例化，渲染画面，即切换场景
3. 交互
4. 一天结束
5. 关闭游戏

**3.Object.h文件，对Object**

static Object\* create(rapidjson::Value& val, MapLayer\* parents);

1. 作用：由数据库创建Object对象，\*Object插入交互地图，\*sprite=nullptr
2. 被调用时机：启动游戏，也就是“加载存档”
3. 被调用方法：前端调用此函数，工厂模式根据json调用对应类（如Land类create方法，创建Land类对象）
4. 返回值：在交互地图对应坐标放上\*Object
5. 形参：

"Info": {

"CropName": "Potato",

"LiveDay": "1",

"MaturationDay": "5",

"Water": true,

"Fertilizer": 1

}

和MapLayer\* parent

virtual void init() = 0;

1. 作用：调用Land类的init（）函数，创建Sprite对象，假如有动作，还需要创建所有的动作序列。
2. 被调用时机：人物走进当前地图，前端需要sprite指针来渲染图片
3. 被调用方法：由director每帧对所有子节点自动调用
4. 实现：调用前台给出的接口（或自行加载）图片、精灵帧（plist和集合图等）

virtual void interact() = 0;

1. 作用：改变渲染；根据目前的人物状态（工具等级、所持物品类型），确定调用哪个状态改变函数，子类中的状态改变函数会改变图像状态把路径找出（写进当前状态json？把json当作类的数据成员），调用哪个函数渲染？、展示动作序列、平移等，并同时改变游戏的后台数据。
2. 被调用时机：交互时
3. 返回值：返回一个结构体，占据的坐标和图片，精灵指针

virtual void settle() = 0;

1. 被调用时机：游戏内一天结束时

virtual void clear() = 0;

1. 作用：delete对象
2. 被调用时机：关闭游戏，即存档结束

三、时间管理器

与ui logic/player对接

游戏中所有场景共用一个时间

时间管理器的实现，要求每天结束时自动调用所有物品的settle函数，时间流逝有自动计数，时间包括游戏天数和当前时间需要写进json文档以便下次打开游戏仍然是这个时间。

睡觉时机：由主角在任何时间与床交互触发；时间在游戏内两点时强制玩家睡觉（不实现）

睡觉功能：把当前数据保存至json，玩家体力值恢复为100%，时间变成第二天早上六点，随后睡醒

成员组成：

前端（渲染画面）

逻辑（更改Object对象，返回Sprite对象用于渲染，更改数据库）

数据库

前后端交互接口：工厂函数

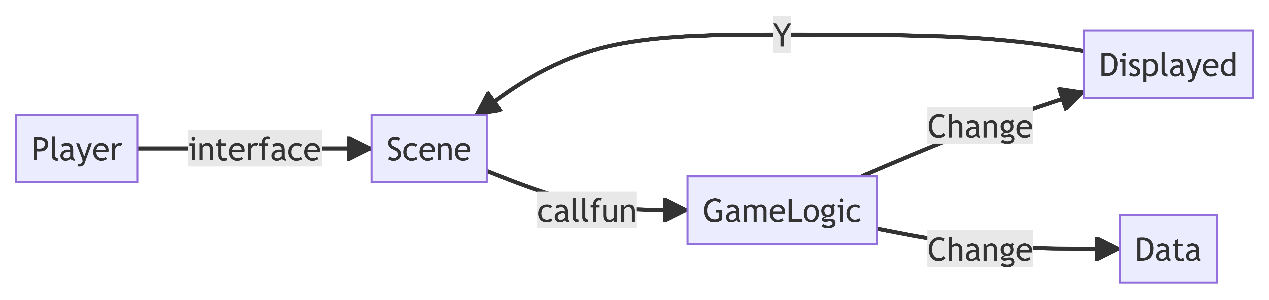
网格化地图：

因为是瓦片地图，所以可以将地图上的所有位置均除以网格大小（16pixel），然后每个格储存对应的网格信息，因而可以实现如下功能：

CollideMap：碰撞地图，通过位图形式储存，记录所有有碰撞体积的网格

Interfacemap：交互地图，储存一个地图上物品的抽象类的指针，用户与地图交互时，自动调用相应对象的interface函数，做出图像以及后台数据的改变。

程序调用方向：



分工：

王相：

玩家可以耕种、种植并收获多种作物，作物的成熟时间根据种类和季节变化。

农场操作包括浇⽔、施肥，以及管理⼲旱和病害威胁，未及时处理会导致作物死亡。

管理主角物品，如工具、种⼦和肥料存储。

挖掘矿物，自动刷新

工具

时间管理器和睡觉。

传送门代码逻辑）包括床和钓鱼台（这个还有吗）

大部分内容能存档

体力消耗和恢复，判断当前体力能否完成当前动作