**1. 类设计与职责分工**

**1.1 SceneManager**

**职责**

* **管理地图场景的加载与初始化。**
* **读取玩家存档，并动态添加玩家角色、动态物体（NPC、作物等）到场景中。**
* **包含 CollideManager，初始化地图碰撞检测的相关逻辑。**
* **创建并返回 Tiled 地图场景的 Layer。**

**接口**

1. **createLayer()**
   * **创建地图场景，加载 Tiled 地图和动态物体，返回场景的 Layer。**
2. **initializeScene()**
   * **加载玩家存档，动态初始化玩家角色和场景物体。**
3. **getCollideManager()**
   * **返回 CollideManager 的实例，供其他模块调用。**

**1.2 UICreator → UIManager**

**职责**

* **管理所有 UI 界面，包括设置界面、背包界面、对话框、主菜单等。**
* **创建和管理 UI 的事件监听器，处理用户交互。**
* **能在事件发生后正确切换到其他场景或界面（如加载存档后进入游戏场景）。**

**接口**

1. **createLayer(UIType type)**
   * **根据 UIType 创建并返回对应的 UI 界面（如设置界面、背包界面）。**
2. **activateUI()**
   * **激活当前 UI 界面，吞没玩家输入事件。**
3. **deactivateUI()**
   * **关闭 UI 界面，恢复玩家输入事件。**
4. **onEvent(EventType event)**
   * **根据用户输入或游戏逻辑触发 UI 相关事件（如切换场景、加载存档）。**

**1.3 GridComputer → GridManager**

**职责**

* **提供地图坐标与网格坐标的双向转换。**
* **管理网格化地图的基本信息（网格大小、地图范围等）。**
* **验证坐标范围，避免非法操作。**

**接口**

1. **initialize(float gridWidth, float gridHeight, Vec2 mapSize)**
   * **初始化网格大小和地图尺寸。**
2. **pixelToGrid(Vec2 pixelCoord)**
   * **将像素坐标转换为网格坐标。**
3. **gridToPixel(Vec2 gridCoord)**
   * **将网格坐标转换为像素坐标。**
4. **isWithinBounds(Vec2 coord, bool isGrid)**
   * **验证像素或网格坐标是否在地图范围内。**

**1.4 ArchiveManager**

**职责**

* **管理玩家存档，包括存储与加载玩家状态。**
* **负责动态物体（如作物）的状态更新。**
* **写入游戏的设定信息（如玩家配置、游戏时间等）。**

**接口**

1. **loadSaveData()**
   * **加载玩家的存档信息，包括作物状态、玩家位置等。**
2. **saveGameState()**
   * **保存当前游戏状态到存档文件。**
3. **updateDynamicObjects()**
   * **更新动态物体的状态（如每天结束时作物的生长状态）。**

**1.5 PlayerController → PlayerManager**

**职责**

* **管理玩家的操作逻辑，包括移动、交互等。**
* **处理玩家输入事件（如键盘或触摸输入）。**
* **当 UI 界面激活时禁用玩家的输入。**

**接口**

1. **initialize(Sprite\* player)**
   * **初始化玩家角色，设置初始位置和状态。**
2. **move(Direction dir)**
   * **根据玩家输入方向移动角色，并调用碰撞检测。**
3. **interact()**
   * **玩家与场景物体的交互逻辑。**
4. **disableInput()**
   * **禁用玩家输入（如 UI 激活时）。**
5. **enableInput()**
   * **恢复玩家输入（如 UI 关闭时）。**

**1.6 CollideManager**

**职责**

* **管理地图的碰撞逻辑。**
* **存储地图的碰撞位图，提供碰撞检测接口。**
* **动态添加或移除碰撞体。**

**接口**

1. **initializeWithMap(TMXTiledMap\* map)**
   * **根据 Tiled 地图生成碰撞位图。**
2. **checkCollision(Rect objBounds)**
   * **检测物体是否发生碰撞。**
3. **getCollisionAt(Vec2 gridCoord)**
   * **返回某个网格是否有碰撞体。**
4. **addDynamicCollider(Collider\* collider)**
   * **动态添加碰撞体。**
5. **removeDynamicCollider(Collider\* collider)**
   * **动态移除碰撞体。**

MapCreator（读取tmx场景，读取玩家存档并添加场景节点，加载玩家人物形象，包含一个CollideManager（管理碰撞体积，存储一个位图，位图的一点是某个网格是否有碰撞体积），创建tiled场景），UICreator（创建设置UI、背包UI、开始界面、对话界面，并且能在事件发生后正确调用其他场景（比如开始界面加载存档后加载开始界面）），GridComputer（单例模式，网格化地图，实现将tiled的xy坐标转化为网格坐标），SaveManager（管理玩家存档，正确添加作物的位置、状态，并在每天结束时正确更新作物的状态信息，写游戏的设定信息），PlayerController（有UI在前台时禁用，同时要处理玩家操作的所有逻辑（统一吞没所有事件，并调用正确的场景、UI）），这样的分工是否合理，是否有不平衡的任务。GridComputer和ArchiveManager是同一个完成的。取名是否合理，假如不合理请给出一个新名称。（目前有一个DocumentManager，用于管理所有的json文档）在这个对话先不要给出代码，只整合并总结所有的想法。