《亚士尼》面向对象设计报告

负责人：沈溯，黄歆

审核人：曾柏铭，李沛珍

日　期：2021.5.18

目录

1 引言

1.1 项目名称

本项目的名称为《亚士尼》。

1.2 编写目的

本设计说明书是开发游戏《亚士尼》的主要依据，为开发人员提供工作基准文件，并对后续阶段的工作起指导作用。

预期读者为软件开发人员（课程项目小组成员：曾柏铭，李沛珍，沈溯，黄歆）、软件评审人员（课程教师、助教）以及其他相关人员。

1.3 项目背景

本项目是课程《软件工程》的期末 PJ，开发者为曾柏铭，李沛珍，沈溯，黄歆，用户为课程老师及其他将使用本软件的用户，该软件的运行环境为 PC 平台 Windows 系统，根据后续开发更新可能实现手机平台 IOS 及安卓系统的移植。

1.4 游戏引言

亚士尼是一个 15 岁的少女，生活在一个小镇上，和她的爸爸妈妈住在一起。这个小镇的生活平静单纯，亚士尼一家和小镇上的其他家庭一样，日复一日的过着简单温馨的生活。

在亚士尼这 15 年的记忆中，她的生活除了年龄的增长并没有什么区别，每日和爸妈一起吃过早餐就独自走到离家不远的学校，上了一天一点都不有趣的课后再独自一人走回家。不上学的日子里生活显得更单调了，没有哪些无趣的课程填补大量的空闲时间，亚士尼的日子显得很空旷。她不喜欢和小镇的其他孩子一起玩，她不能理解他们的游戏到底有什么好玩的。

但有一块记忆和这些截然不同。亚士尼记得小镇的边界处有一栋紫色的房子，在那所房子里有着和小镇截然不同的东西。但是亚士尼的记忆中并没有告诉她那是什么东西，这是一个神秘的存在。

在亚士尼的平静生活中，这栋神秘的房子总是时不时出现在她的脑海中，但是她没有去看过，也不知道它的具体位置。这栋房子和她的生活仿佛毫无交集，直到有一天……

1.5 参考资料

[1]软件工程 钱乐秋 赵文耘 牛军钰编著 清华大学出版社 2016 年 9 月

2 任务概述

2.1 项目总体目标

本项目预期实现一个名为《亚士尼》的第三人称恐怖向冒险解谜游戏的 demo 版本，玩家可通过与游戏场景中的物品、人物等互动，完成游戏设定的任务，解开谜题，逃出游戏设定的场景。

预期将实现游戏 UI，人物场景建模，游戏架构，基本操作流程和初始 1~2 章剧情，后续将采用增量模型继续完成整个游戏剧情。

2.2 运行环境

运行环境：Windows 系统。

2.3 开发环境

开发环境：Windows 系统，Unity 平台，使用 C# 语言。

3 系统设计

3.1 子系统划分

在本游戏中，整个系统可以划分成如下几个系统

- **UI 子系统**

  负责整个 UI 显示，以及游戏中各类交互消息的显示

  UI 子系统可以继续划分为以下两个子系统

• **主界面 UI**

主界面 UI 较为静态，分为几个子页面。几个子页面通过有限状态机连接起来

以下为主界面转换流程:

子页面 1 应包含游戏 title, 登录按钮，注册按钮和退出游戏按钮

单击注册按钮进入页面 2

• 页面 2 包含：账号输入框，密码输入框，重复密码输入框，点击注册按钮，返回按钮

• 点击注册确认无误后回到页面 1

单击登陆按钮进入页面 3

• 页面 3 包含：账号输入框，密码输入框，点击登陆按钮，返回按钮。

• 点击登陆后进行账号密码校验，失败则清空输入框，成功则进入页面 4

• 页面 4 包含：存档点展示

• 点击任意存档点进入游戏

单击退出游戏按钮则退出游戏

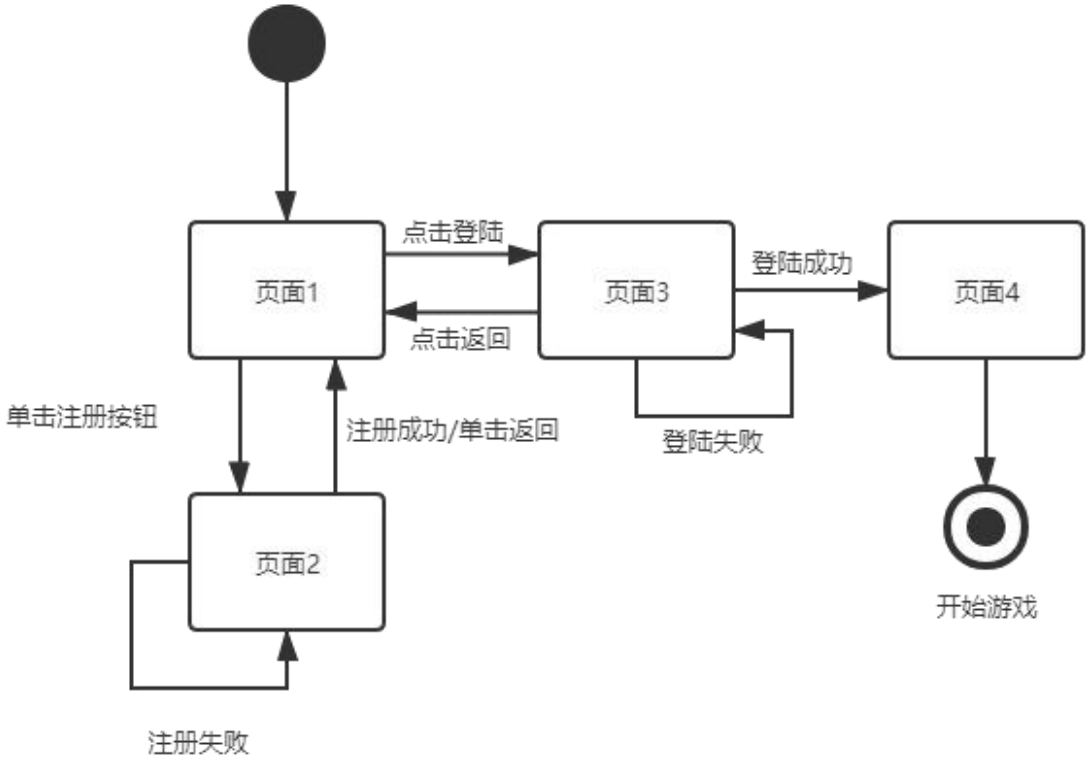

图 0 主界面 UI 状态机图

• **交互面板 UI**

交互面板 UI 用于展现事件子系统的变化，提供给玩家必要的信息。其消息展示凌驾于游戏场景之上，消息的展示和变化由子系统间通信决定。

• **用户管理子系统**

用户管理子系统用于管理所有注册的玩家信息，同时维护每个用户的所有存档。

• **事件子系统**

是一个有限状态机，由玩家的各类动作触发状态转化，完成对游戏过程中各状态的具象化。

• **物品子系统**

包含游戏中各类物品的类设计。物品的具体实例化参见具体对象。

3.2 子系统关联

物品子系统是一个相对静态的子系统，其可以视为若干具体对象的集合。事件子系统则是以物品子系统中的对象产生的交互行为为条件，发生状态转换的一个状态机设计集合。UI 子系统则凌驾于上述两个子系统之上，既有较为静态的 UI 部分，也有动态的 UI 部分。

子系统间通过发送消息进行通信。所有的通信可以归纳为一个抽象合约类。针对不同的子系统间通信，合约类又可以泛化成不同的具体合约类。合约类的设计会在抽象对象中详细描述。

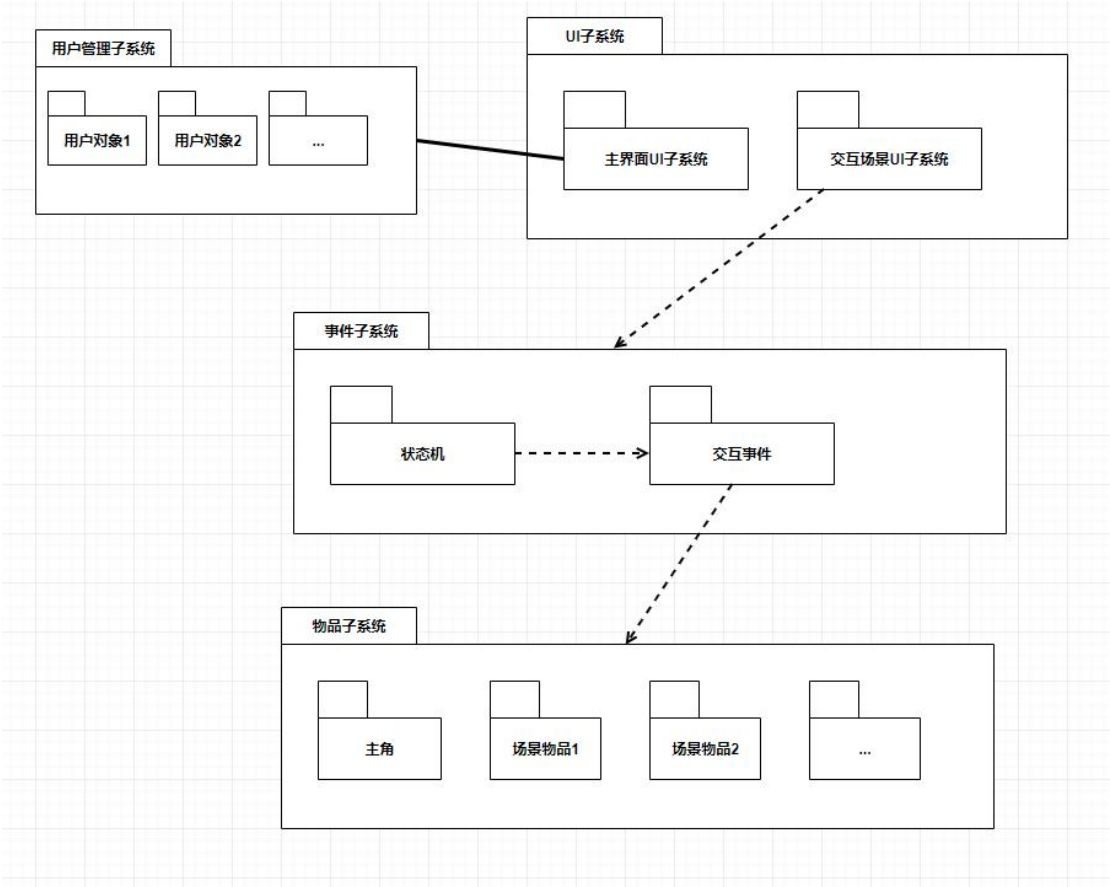

图 1 游戏系统的包图

3.3 任务管理设计

本游戏中主要任务有两种类型：

（任务均不可并行）

1. 事件触发任务

事件触发任务可以划分为三类：

1）时间类触发：优先级 1

当触发时间类事件，如炸弹爆炸，则任务开始计时，事件被加入事件触发器中等待。计时结束时无论处在什么情况下均触发事件如爆炸。

2）地点类触发：优先级 2

使用事件触发器检测特定触发地点，当玩家移动到该地点，事件触发器检测玩家当前属性是否满足触发事件要求，若满足情况则触发事件。

3）操作类触发：优先级 3

玩家对场景使用特定道具或选择特定选项则触发事件。需与物品操作任务相结合。事件触发器检测到玩家使用特定道具触发场景，检测该道具是否满足触发条件，若满足则触发事件。

事件触发任务流程图：

1. 玩家操作触碰设置在事件触发处的事件触发器。
2. 事件触发器将玩家目前状态发送给事件数据库进行比对，查询是否可以触发事件。
3. 若比对成功，则事件数据库将相应事件对象传送给事件触发器，事件触发器执行事件方法内容，修改游戏数据，并将事件触发结果反馈给玩家。
4. 若比对失败，则还不能触发事件，事件数据库反馈事件触发失败信息，什么都没发生。

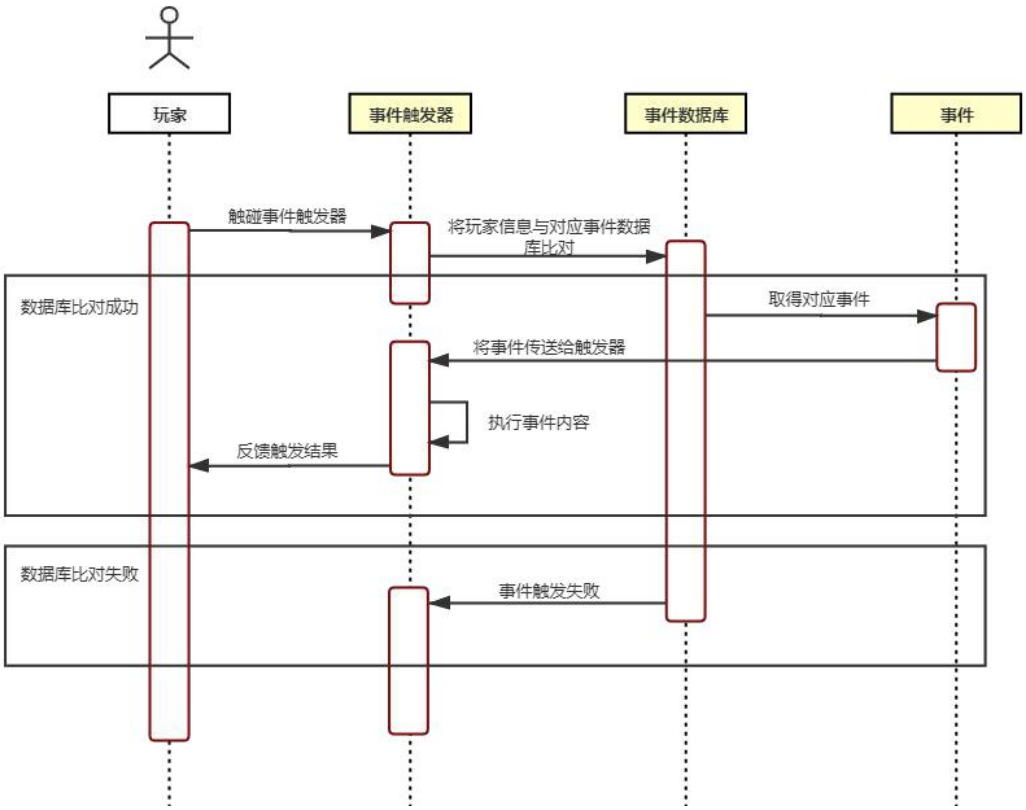

图 2 事件触发流程图

2. 物品操作任务

物品操作任务可分为以下两类：

物品使用类：优先级 4

当玩家处在无事件触发状态下，打开背包选择物品，点击使用，则使得工具台向物品数据库发送查询，可以使用则工具台进行相应操作。

2）物品合成类：优先级 5

玩家在无事件触发状态下，打开背包选中物品，点击合成，则工具台将相应物品数据加入工具台中，等待另外被合成物品。当玩家点击工具台中的合成键则向物品数据库发送查询，可以合成则工具台进行相应操作。

物品使用流程图：

1. 玩家打开背包，弹出背包界面
2. 玩家选择要使用的物品，弹出“使用”与“合成”按钮。
3. 玩家选择“使用”，则向工具台提交使用请求。
4. 工具台将物品信息发送给物品数据库，查询该物品是否能够使用。
5. 若物品可以使用，则物品数据库修改该物品信息，并将物品修改情况反馈给工具台，工具台改变自身状态并且将更新反馈给背包，背包修改自身状态。
6. 若物品不可使用，则物品数据库将物品使用失败情况反馈给工具台，工具台将使用失败反馈给背包。
7. 背包将物品使用情况反馈给玩家

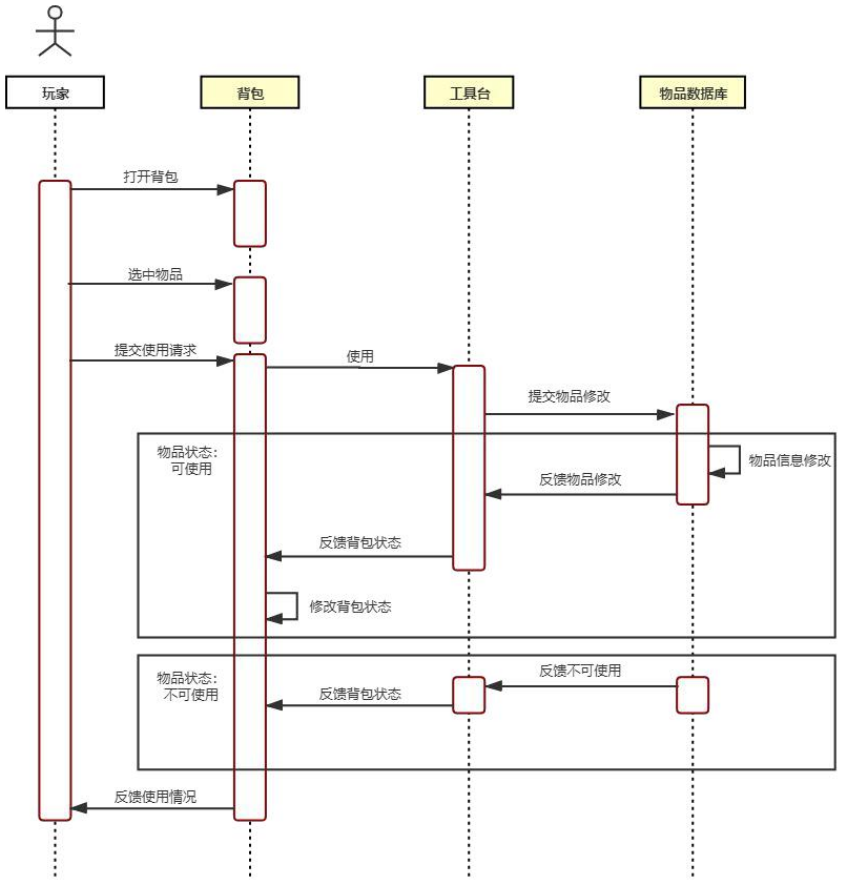

图 3 物品使用流程图

物品合成流程图：

1. 玩家打开背包，弹出背包界面
2. 玩家选择要使用的物品，弹出“使用”与“合成”按钮。
3. 玩家选择“合成”，则向工具台提交合成请求。
4. 工具台弹出合成界面，等待用户选择另外的被合成物品。
5. 用户选择所有被合成物品后点击合成，工具台将所有被合成物品信息发送到物品数据库中查询。
6. 若物品可以合成，则物品数据库设置所有被合成物品信息为已销毁，并新增合成物品。并将物品修改情况反馈给工具台，工具台改变自身状态并且将更新反馈给背包，背包修改自身状态。
7. 若物品不可合成，则物品数据库将物品合成失败情况反馈给工具台，工具台将合成失败反馈给背包。
8. 背包将物品合成情况反馈给玩家。

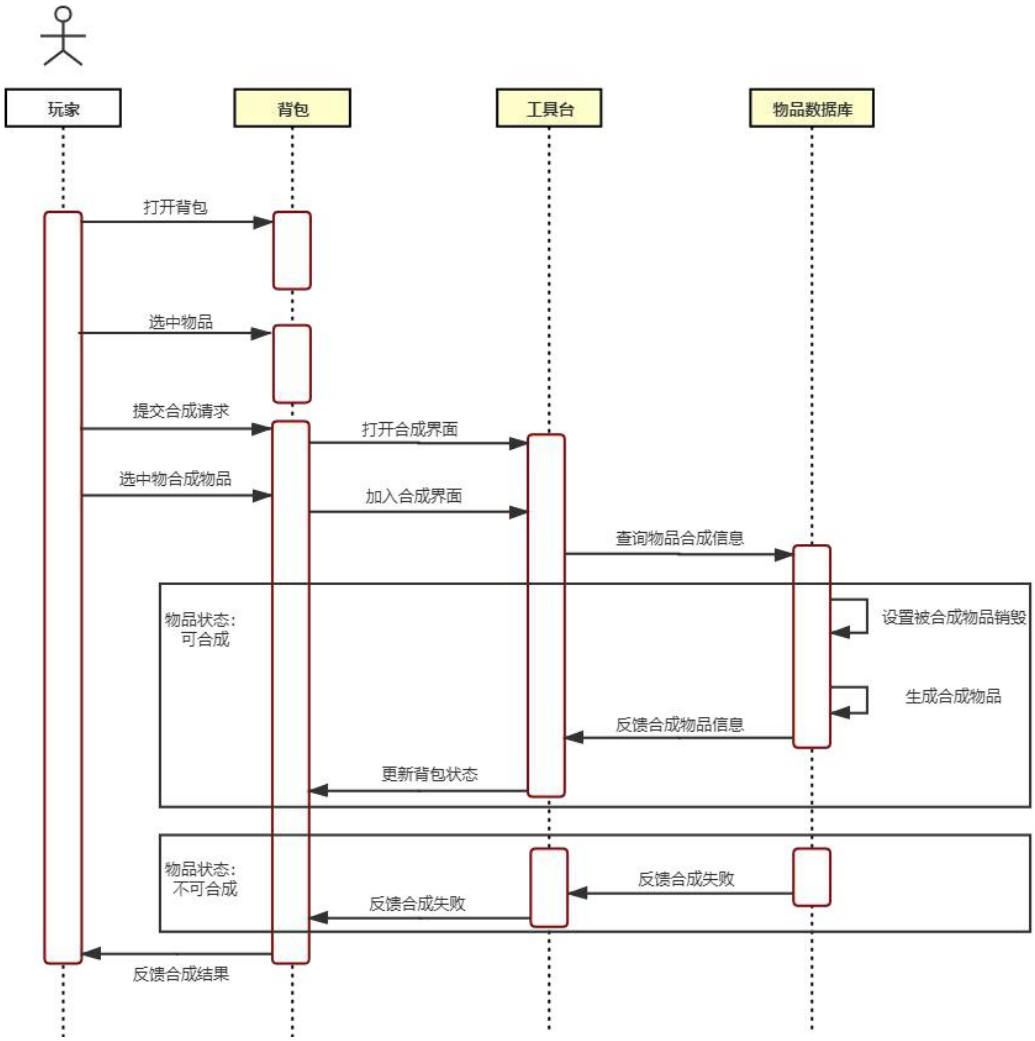

图 4 物品合成流程图

3.4 数据管理设计

本游戏中的怪物类，人物类，场景类均为实时变化数据，不需要额外数据管理，则数据管理主要为两个部分：

1. 物品数据库

该数据库划分为三个部分：**已创建/未创建/已销毁**

存在部分又划分成两个部分：**背包内/场景内**

用户仅可访问背包内的物品数据。

数据库内存储物品类实例，拥有增加，删除，修改，查询四个操作。

增加：在合成道具时使用，根据工具台发送的合成道具信息，在“未创建”类中找到相

应物品将其移动到“已创建”部分。

删除：在使用/合成道具时使用，根据工具台发送的使用/合成道具信息，在“已创建”

类中找到相应物品将其移动到“已销毁”部分。

查询：由工具台通过背包存储的索引检测数据库，数据库返回相应的物品数据进行比对，

比对成功则将该物品实例返回工具台，被工具台调用。

2. 事件数据库

该数据库划分为两个部分：**已触发/未触发**

数据库内存储事件实例，用有查询和打包发送两个操作。

所有事件又按照前述类别不同有不同存储方式。

时间类触发事件：优先级最高，触发条件最为严格，根据剧情固定触发，不能在事件触发器中被查询，单独存储。

地点类触发事件：通过地点索引存储，使用观察者模式，当观察者观察到人物靠近触发地点，就向事件触发器发送人物信息和触发申请，事件触发器通过观察者所在地点根据索引查询数据库，比对触发条件，若比对成功则将事件发送给触发器执行。

操作类触发事件：与场景类绑定。当人物点击可交互场景，即可选择使用物品栏物品，此时事件触发器检测该物品能否触发场景事件，可以则向时间数据库发送请求。

4 对象设计

对象设计为每个类的属性和操作做出详细的设计，并设计连接类与它的写作者之间的消息规约

4.1 具体对象

物品子系统中的对象具有具象化程度较高的特点，我们将其称为具体对象。对分析阶段已经列出了如下抽象类。设计阶段针对某些类的操作，在实现上进行了不同程度的修改。

类名	属性	操作
用户	用户名，密码，存档点[]	读档点，存档，删除存档，匹配密码
角色	*状态，控制参数*	*控制*
主角: 角色	血量，主角属性	
怪物: 角色	怪物属性	
物品	*名称，功能描述，使用状态*	*读取| 修改物品信息*
工作台	物品[]	放入物品，使用物品，合成物品
背包	物品[]	取出物品，放入物品
事件	名称，事件描述	触发事件

表 1 抽象类

(斜体表示某些抽象的父类，圆括号中的属性表示从父类继承而来的类)

受制于篇幅，本节选取代表性较强、内部较为复杂的几个类进行说明。

- **用户类**

用户类从属于用户管理子系统，用于玩家登录登出以及存档读档。用户类具有一个字符串型的用户名 `username` 以及一个字符串型的密码 `password`. 同时，每个用户类持有一个存放存档的存档类数组 `archive[]`. 当玩家不持有相关用户名的密码时，显然不能访问其存档。所以 `password` 和 `archive[]` 属性都是 `private` 的。

每个用户类需要维护一个密码匹配 `matchPassword` 接口，以便登陆时的密码匹配；以及三个对存档的操作接口。这些接口都是对外可见的。

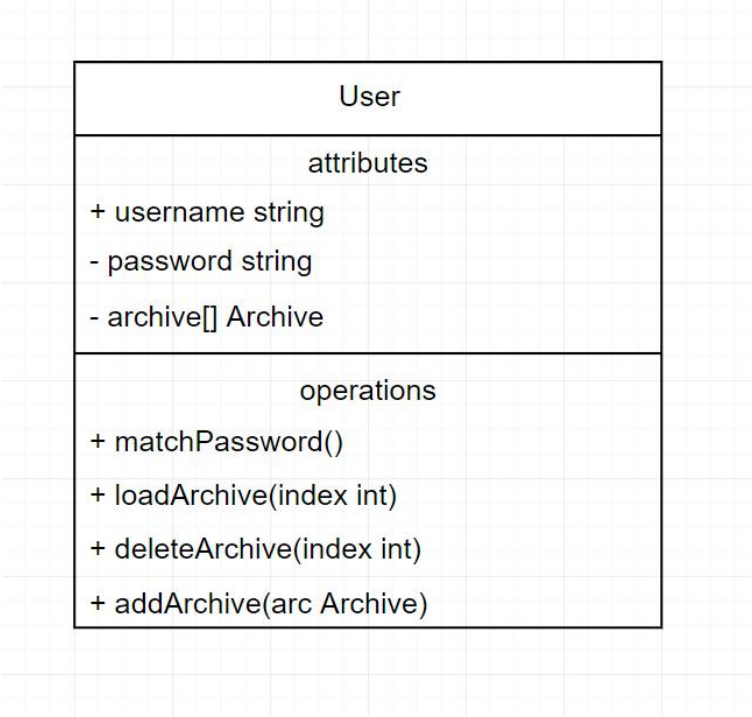

图 5 玩家类图

- **角色、 主角与怪物**

主角和怪物都是角色类的子类。主角和怪物都有一个用 int 标识的自己的状态，所以这个属性属于角色类。同时，二者都有自己的控制逻辑，但是共享一套类似的控制参数。所以角色类应该还有控制参数的属性，以及一个控制的操作。从实现上讲，控制操作是角色类的一个虚操作，主角类和怪物类都应该重写这个操作。主角的事件互动和怪物的攻击操作也都可以归纳进控制操作中。

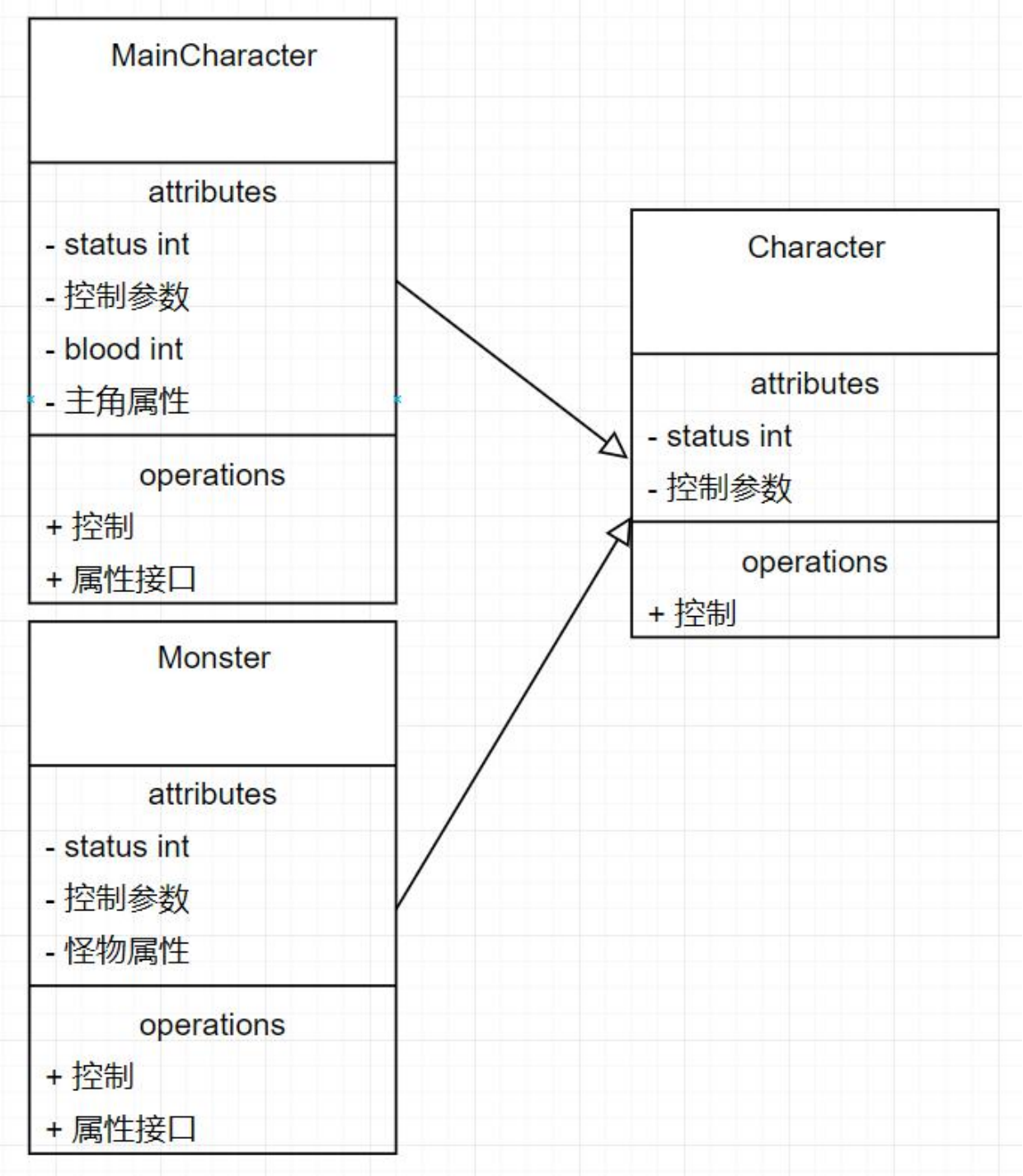

图 6 角色类和他的泛化

状态机图：

主角

主角主要有三个状态：

1. 正常：初始玩家角色创建时状态，当受到触发陷阱或不良影响事件时仅产生血条上的变

化，但并不影响主角移动速度等状态。

2. 中毒：当受到中毒事件影响进入中毒状态，即导致移动减缓，视线模糊，血量缓慢减少。

使用解毒药可回到正常状态。

3. 死亡：当遭遇怪物攻击或血量清零即触发死亡，游戏结束。

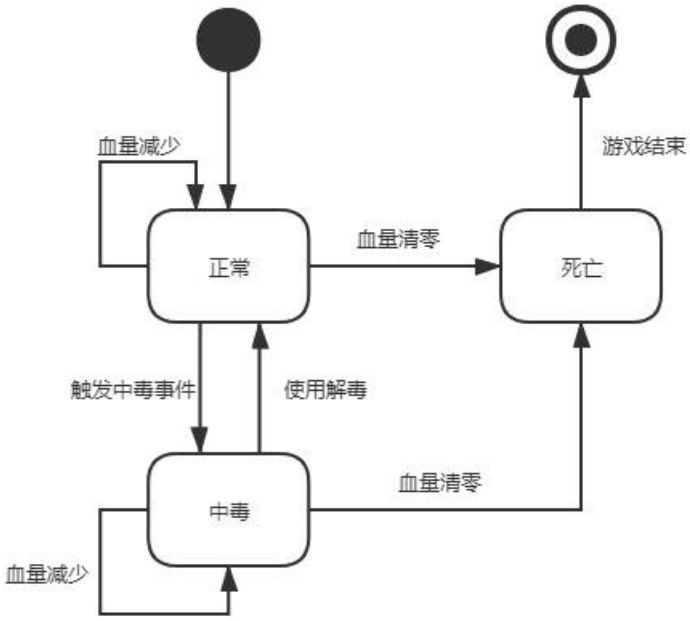

图 7 角色类状态机图

怪物类：

怪物类主要有三个状态：

1. 巡逻状态：怪物在创建之初即沿着固定路线巡逻。

2. 寻路状态：当主角进入怪物视线范围内（距离属性 sight），即改变怪物移动方式，为向主角所在目标寻路。若主角离开视线范围，则返回原路径，回到状态 1。

3. 攻击状态：当主角进入怪物攻击范围（距离属性 attack），即进入攻击状态，向主角发动攻击。若主角再次离开攻击范围则返回状态 2。

4. 由于怪物只能通过剧情杀死，因此不需要设置死亡状态。

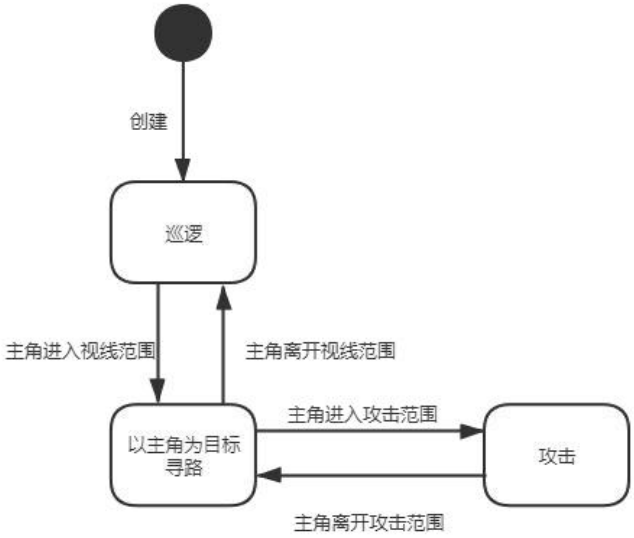

图 8 怪物类状态机图

• 物品、 工作台与背包

物品也是一个抽象类，场景中的若干可交互和不可交互的元素都是其子类的实例。

工作台和背包各维护一个物品的数组。所有涉及 UI 变化的操作就被划分进了交互场景 UI 子系统来提供显示。

图 9 物品类

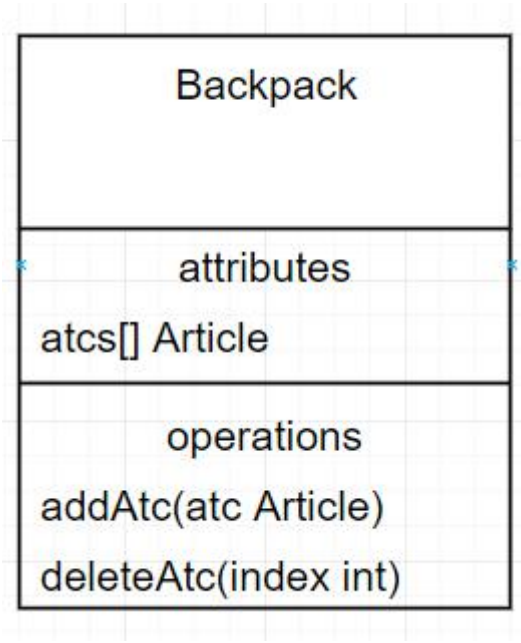

图 10 背包类

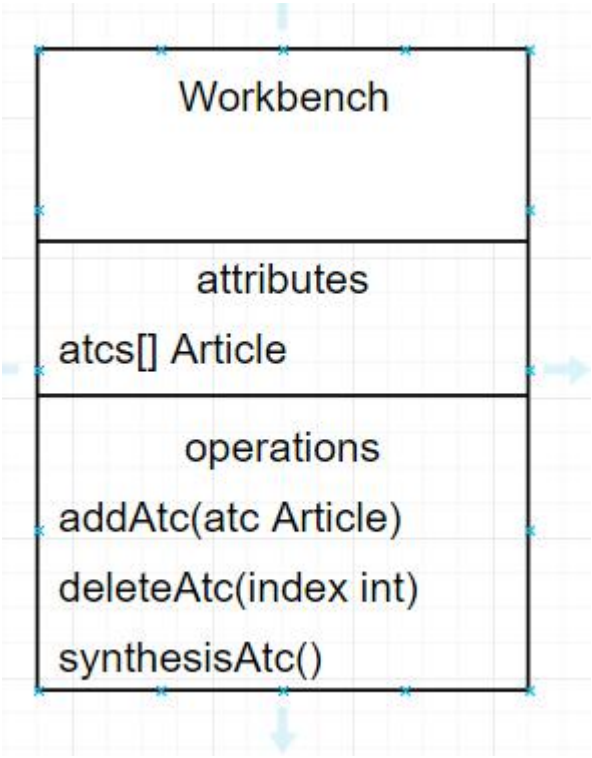

图 11 工作台类

状态机图：

物品类：

物品类有以下状态：

1. 当物品处于场景中，则处于物品数据库中“未加入背包”状态，不能对物品进行任何操作。

2. 当物品在场景中被选中，则被加入到背包中，设置为“在背包中”状态。

3. 当物品在背包中被选中，则弹出“合成”与“使用”选项：

选择“合成”使得物品进入“待合成”状态。

若合成成功，向玩家反馈合成成功信息，物品被设置为“已销毁”状态。

若合成失败，向玩家反馈合成失败信息，并回到“在背包中”状态。

选择“使用”，若使用成功，向玩家反馈使用成功信息，物品被设置为“已销毁”状态。

若使用失败，向玩家反馈使用失败信息，并回到“在背包中”状态。

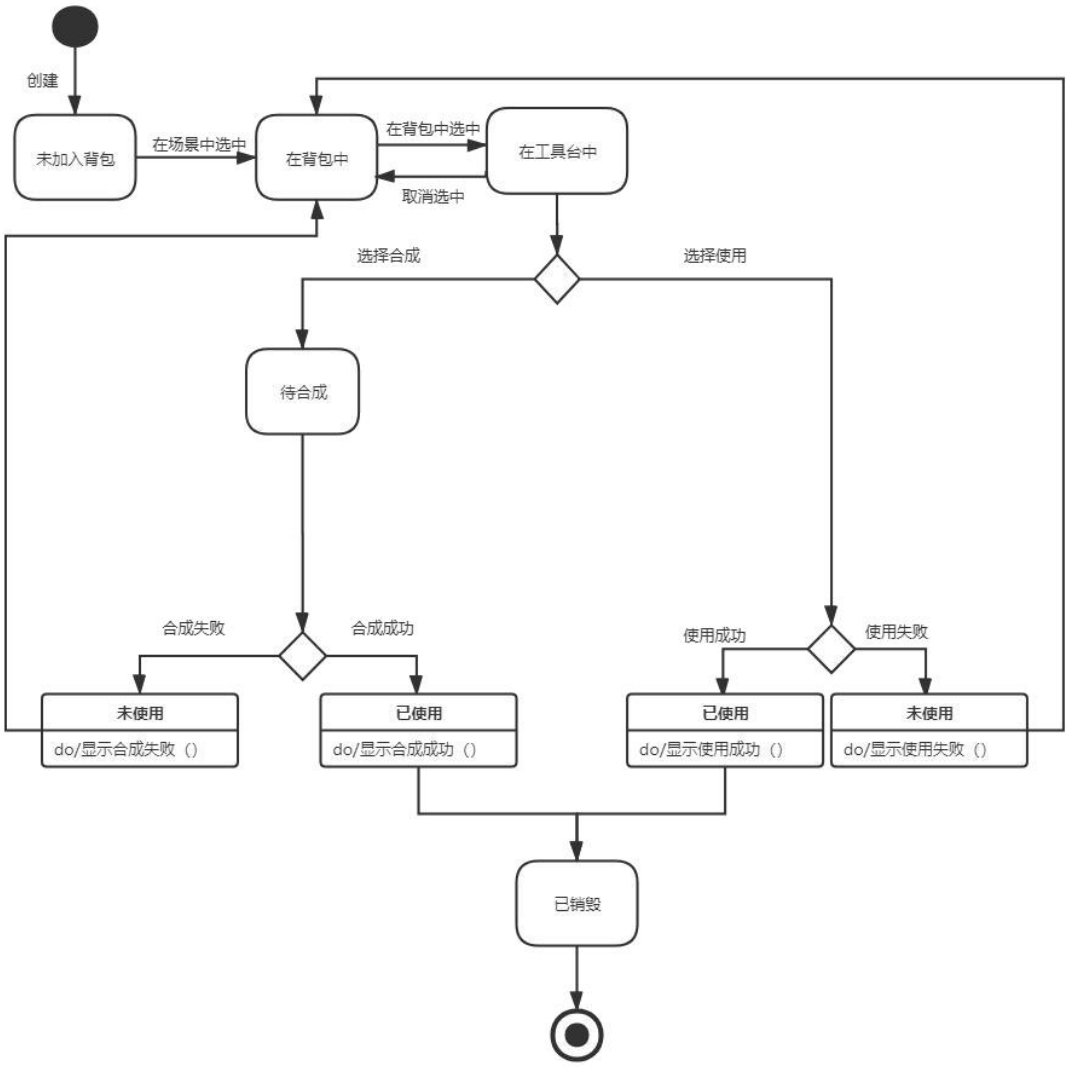

图 12 物品类状态机图

4.2 抽象对象

抽象对象包括子系统对象和子系统间通信使用的合约类。

- **子系统对象**

每一个子系统都应该有一个实例化的对象。每个子系统对象会向其他子系统对象发送信息， 或是接收其他子系统对象的信息，并处理该信息或是将该信息交付子系统内部类来处理。

系统类是对所有子系统类的一个抽象，其中定义了系统应该有的操作，即发送消息和接收消息。所有子系统类都是系统类的一个子类，并且应该根据自己的通信合约重写这两个方法。

在本小节中，笔者将讨论所有子系统对象的功能（责任）、交互对象（协作者）。

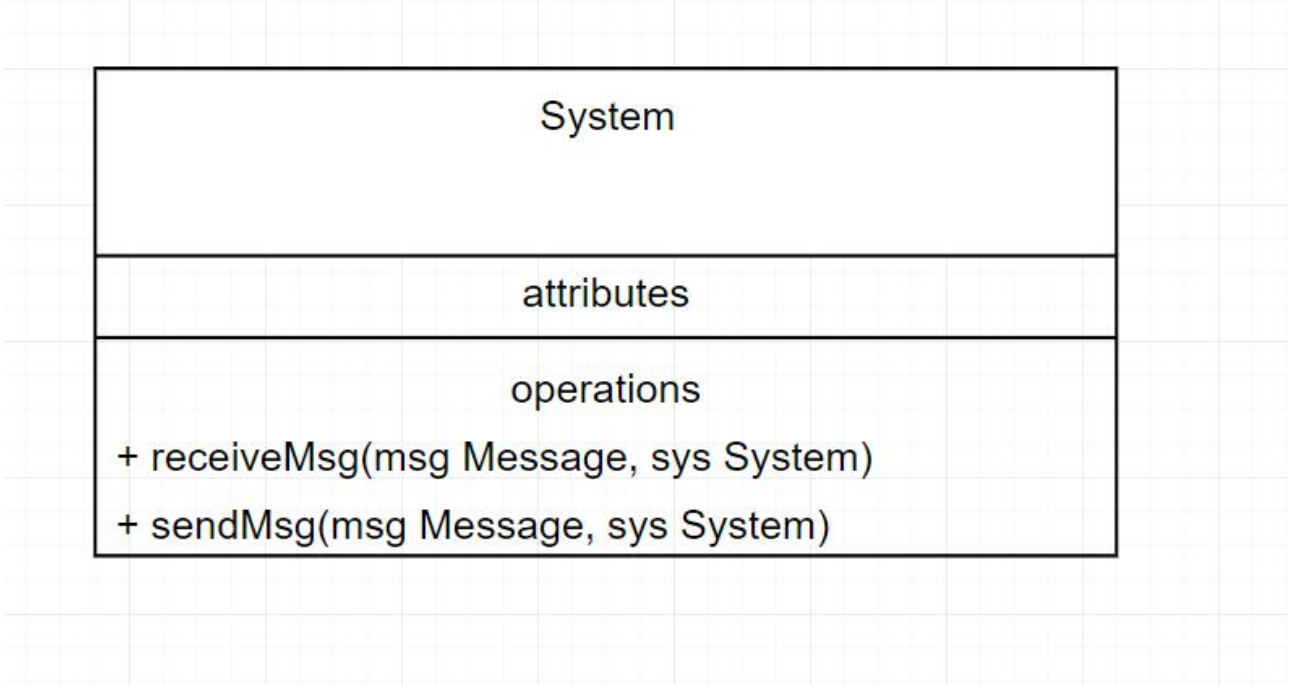

图 13 系统类

- **用户管理子系统**

用户管理子系统(User Manager System, UMS)负责存放用户信息。UMS 需要和主界面 UI 子系统进行通信。

对于接收到的注册、登录、登出以及删除用户的请求，UMS 需要返回是否成功的回应。

此外，登录成功后，UMS 需要将对应当前用户的所有的存档发送给主界面 UI 子系统用于显　示，以供用户挑选。

用户管理子系统不会主动向主界面 UI 子系统主动发送信息。

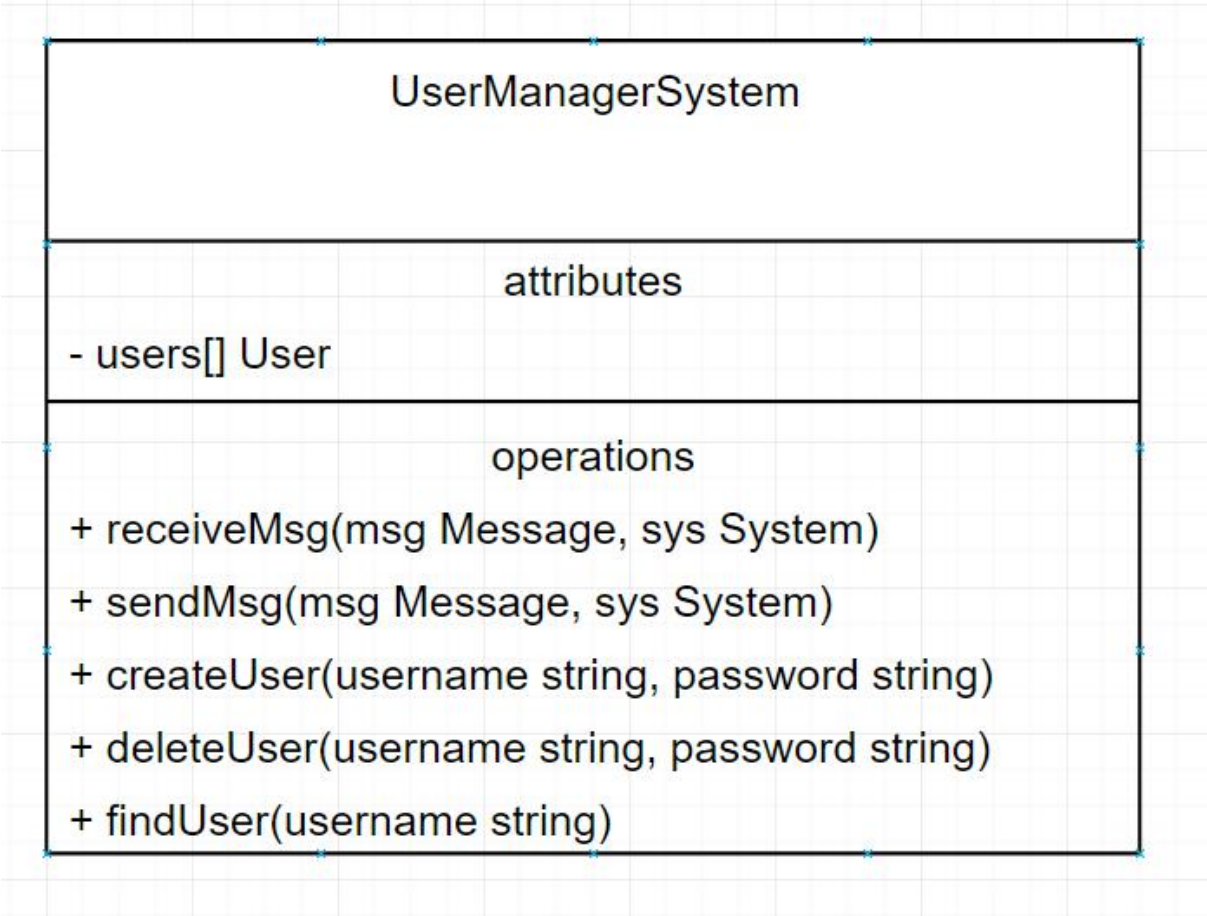

图 14 用户管理子系统类（该类只有一个对象）

- **主界面 UI 子系统**

主界面 UI 子系统包含一系列的页面。事实上，每个页面都是一个类，出于篇幅限制这里不予展开。

用户在与主界面交互的过程中会发生注册、登录、登出、读档、存档、删档等操作。这部分交互由用户通过鼠标操作激活。

在登录成功后，主界面 UI 子系统需要读取用户存档并予以展示。

需要注意的是，档案中存储了存档时所有的子系统的状态，当读档的时候，档案中的状态会覆盖当前系统中所有子系统的状态，这并不属于子系统通信。

• **交互场景 UI 子系统**

交互场景 UI 子系统负责对产生交互行为时需要出现在 UI 中的内容进行展示。交互场景 UI 子　系统对象包含了若干可显示元素。一个交互场景 UI 子系统显示的例子如下。

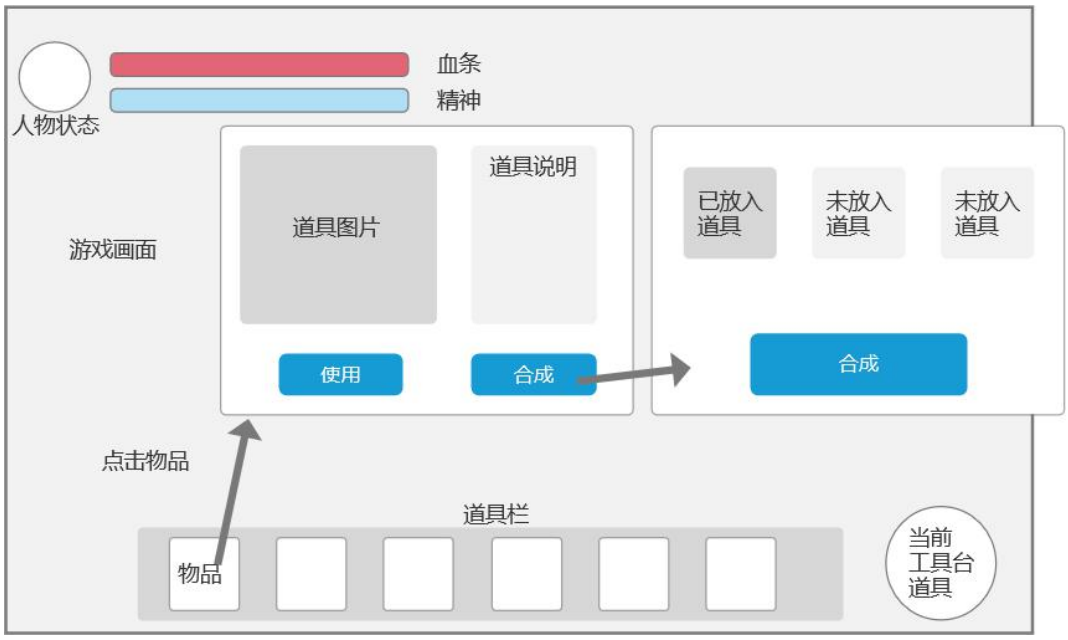

图 11 交互场景 UI 子系统显示的例子

交互场景 UI 子系统的协作者是事件子系统。

交互场景 UI 子系统维护一个二进制状态码，用于标识组成交互场景 UI 子系统的各个元素的可见性。每当事件子系统捕获到一个交互行为，事件子系统向交互子系统发送的通信中包含这个二进制状态码，用于标识交互事件发生时游戏期望的 UI 界面。交互场景 UI 子系统捕获到这个信息，并更新相应的元素的可见性，最后得到期望的游戏 UI 界面。

除了对事件子系统捕捉到的交互行为负责显示，交互场景 UI 子系统还可以自主展示显示内容。例如，再单击物品时会出现物品的详细信息。这种 UI 的变化也是通过调用 `changeState` 方法来修改状态码来实现的。

由于 UI 的显示应当由交互场景 UI 子系统来决定，所以 `changeState` 应该是一个 private 方法。

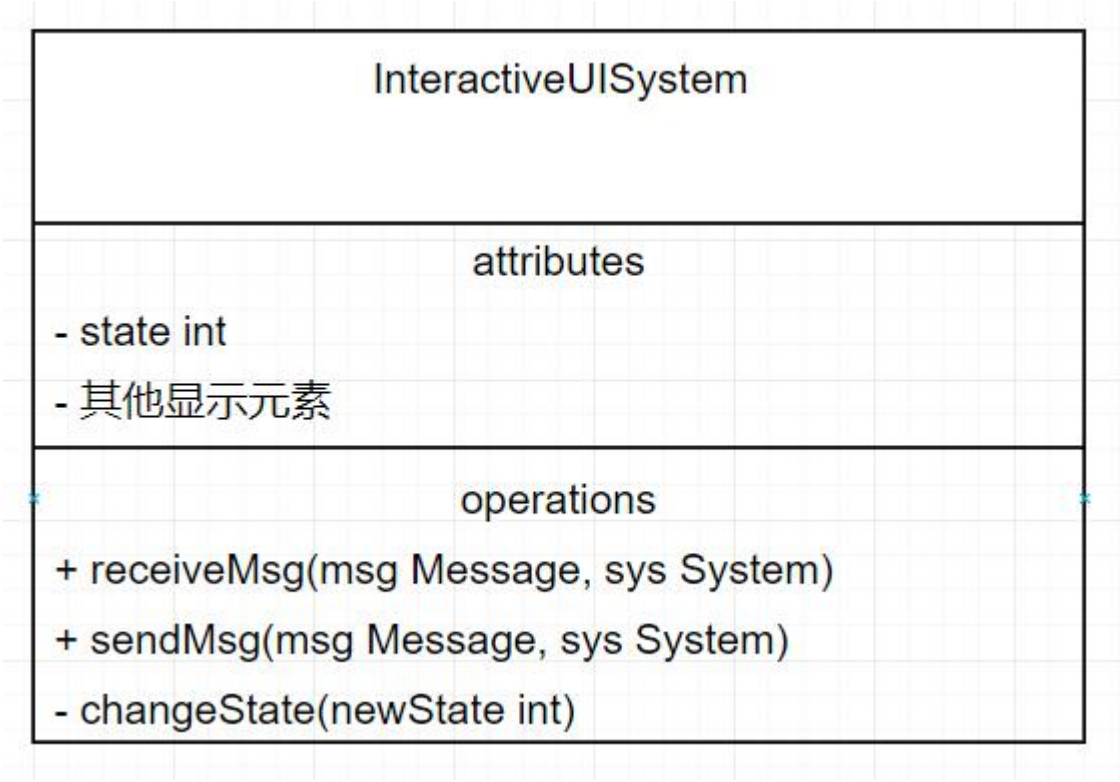

图 12 交互场景 UI 子系统

- **事件子系统**

事件子系统负责在监测到各类触发器触发时，向交互场景 UI 子系统发送 UI 变化请求，并处理触发对应事件，包括修改对象属性等，事件子系统通过内部方法直接修改物品信息，不需经过调用物品子系统。

- 物品子系统

物品子系统维护一个具体对象的集合。

- 合约对象

合约对象是子系统在交互过程中发送的消息抽象化成的对象。

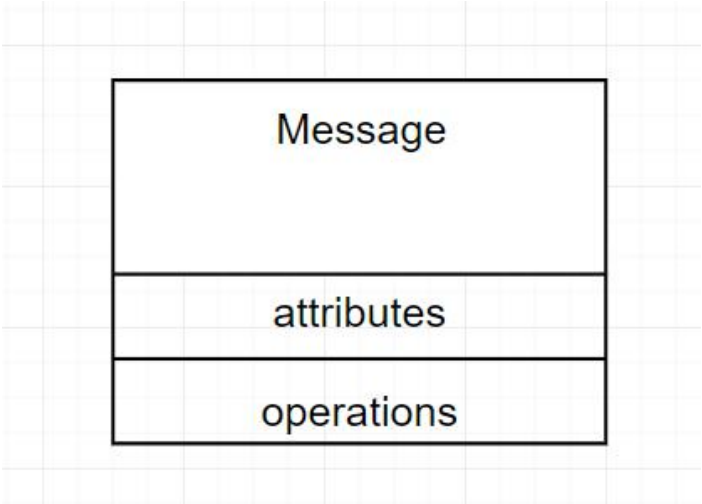

图 13 Message 抽象类

合约的规定较为复杂，这里仅选取用户管理子系统和主界面 UI 子系统的交互作为一个例子，来展现合约制定的过程。

考虑用户管理子系统和主界面 UI 子系统之间的交互行为

1. UMS 需要发送的内容，即主界面 UI 子系统需要接收的内容

   当接收到注册请求、登录、登出以及删除用户的请求，UMS 需要返回是否成功的 回应。

   当登录成功后，UMS 需要将对应当前用户的所有的存档发送给主界面UI子系统用 于显示，以供用户挑选。

   所以 UMS 发送给主界面 UI 子系统的 Message 应包含一个布尔变量，来标识请求的是否成功。同时包含一个存档的数组。

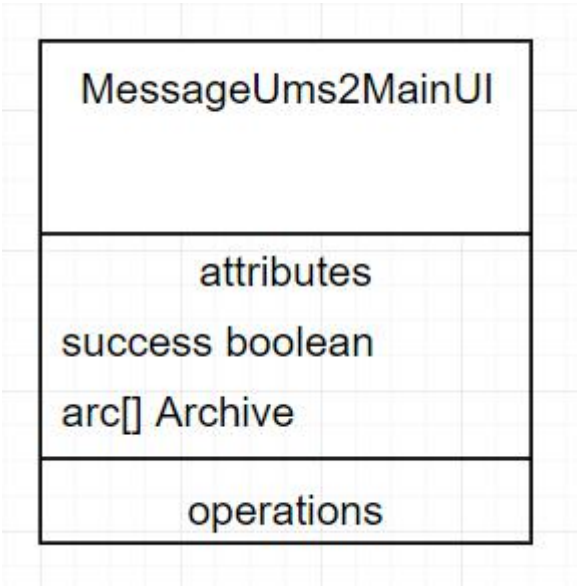

图 14 UMS 向主界面 UI 子系统通信的合约类

2. UMS 需要接收的内容，即主界面 UI 子系统发送的内容

主界面 UI 子系统会发送注册、登录、登出以及删除用户的请求。

我们可以用一个状态码表示请求类型，然后记录每次请求的 username 和 password.

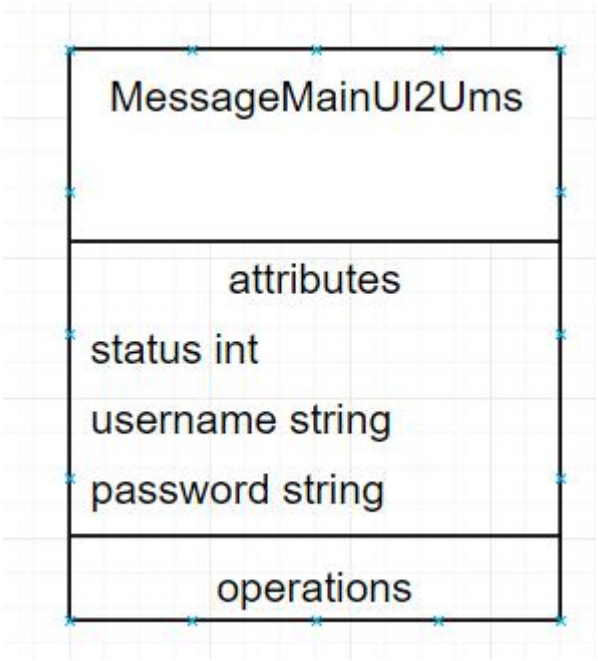

图 主界面 UI 系统向 UMS 通信的合约类

5 结尾

以上为本小组软件工程课程项目游戏《亚士尼》的面向对象设计报告，非常感谢您的阅读，如有任何建议望联系设计小组成员提出您的宝贵意见，再次感谢。