<移动迷宫>

软件架构设计

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **修订说明** | **作者** |
| <2017-04-25> | <1.0> | <初稿修订> | <组员> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[1引言 1](#_Toc448817922)

[1.1编写目的 1](#_Toc448817923)

[1.2适用范围 1](#_Toc448817924)

[1.3定义 1](#_Toc448817925)

[1.4参考资料 1](#_Toc448817926)

[1.5概述 1](#_Toc448817927)

[2 目前软件系统体系架构 1](#_Toc448817928)

[3 软件系统架构设计目标 2](#_Toc448817929)

[4 建议的软件系统架构 2](#_Toc448817930)

[4.1 概述 2](#_Toc448817931)

[4.2 用例视图 2](#_Toc448817932)

[4.3 系统逻辑视图 2](#_Toc448817933)

[4.4系统运行视图 3](#_Toc448817934)

[4.5系统实现视图 3](#_Toc448817935)

[4.6系统物理视图 3](#_Toc448817936)

[4.7 边界条件设计 3](#_Toc448817937)

[4.8数据管理设计 3](#_Toc448817938)

[4.9 其他设计 3](#_Toc448817939)

软件需求规约

# 1引言

## 1.1编写目的

本软件架构文档的编写目的是对移动迷宫软件的系统结构进行描述和定义，在之前的需求规约、系统分析的基础上，利用各种模型，详细展示系统的结构，为后续的软件实现工作奠定基础。文档将定义系统架构的设计目标，描述系统的结构、子系统定义、软硬件部署、数据管理、软件控制、边界条件等内容。本文档用于开发团队明确系统的架构和设计，并以之为依据进行开发工作。

## 1.2适用范围

本文档适用于的软件：移动迷宫

与该软件相关的特性、子系统、模型等均符合本文档中的内容。

## 1.3定义

本文档中设计的术语定义在项目词汇表中给出。

## 1.4参考资料

《面向对象软件工程——使用UML、模式与JAVA》（第3版），清华大学出版社，2011.

## 1.5概述

本文档包括引言、目前系统架构、系统架构设计目的、建议的软件系统架构四部分。目前系统部分对当前潜行类游戏进行分析并指出其不足。系统架构设计目标部分结合软件需求，列举出系统设计的目标。建议的软件系统架构给出系统的架构和子系统的分解，并以文字表述和模型图相结合的方式展示系统的对象设计、软硬件部署、数据管理、软件控制、边界条件等设计。本文件的各部分内容联系紧密，互为补充和对照，共同呈现本系统的软件架构。

# 2 目前软件系统体系架构

目前存在的系统是潜行类游戏，即拥有固定视角范围的玩家共同进行游戏。目前存在的系统主要运行在大型平台上，同时对系统硬件配置有比较高的要求。

新的系统是手机端的APP，快捷简便且便于用户上手。

# 3 软件系统架构设计目标

系统架构的设计目标如下：

1. 高可用性：本软件作为一个手机游戏，如果在游玩的过程中发生系统错误，将会严重打击玩家的积极性。因此本系统需要保证较高的可用性。
2. 安全性：本系统计划开发网络联机游戏模式，因此必须保证在多人联机过程中，系统的安全性。
3. 高性能：本系统运行过程中实时响应，并且具有比较繁重的即时渲染工作，因此要求有较高的性能。
4. 可扩展性：本系统在初期规模较小时，不需要较为高规格的硬件。但应该考虑到在用户数量增加后，以及游戏内容扩展后对系统的扩展。

# 4 建议的软件系统架构

## 4.1 概述

本系统的架构及采用架构模式。

1. 三层体系结构

本系统用三个层次来组织子系统：第一层为用户界面层，提供与用于交互的借口，级App的UI。第二层为应用逻辑层。着一层主要负责控制并实现系统功能，包含MapCreator，SinglePlayer，MultiPlayer三个子系统，分别负责地图编辑、单人游戏和多人游戏。第三层为存储层。主要负责系统数据的存储、检索和查询。

三层体系结构比较成熟，将借口和应用逻辑分开，耦合度低，灵活性和扩展性强，方便开发过程的分工。因此本系统采用这个结构。

1. 模型视图控制器模式

MVC模式适用于交互系统。在上述的三层体系结构中已经将系统分为视图（第一层）、控制器（第二层）和模型（第三层）三部分。

本系统为独立开发，在支持双方交互通信功能的实现上，将使用第三方的类库。

本系统包含以下子系统：

1. UI：负责用户界面部分。
2. SinglePlayer：负责单人游戏部分。
3. Multiplayer：负责多人游戏部分。
4. MapEditor：负责地图编辑器部分。

## 4.2 用例视图

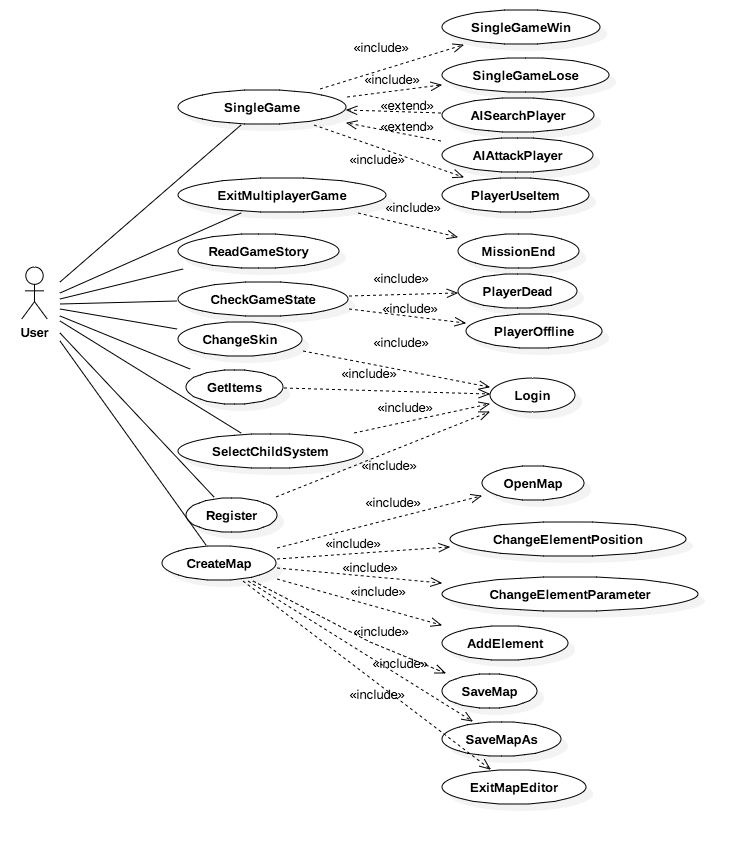


图4-1用例图

## 4.3 系统逻辑视图

1. 系统架构

本系统采用了三层体系结构和客户机／服务器模式。最顶层是UI层，是用户直接交互的客户机部分。中间层是应用逻辑层，包含实现系统功能的各项子系统。第三层为公共服务层，实现对数据的存储和管理、移动端与服务器端的通信功能。

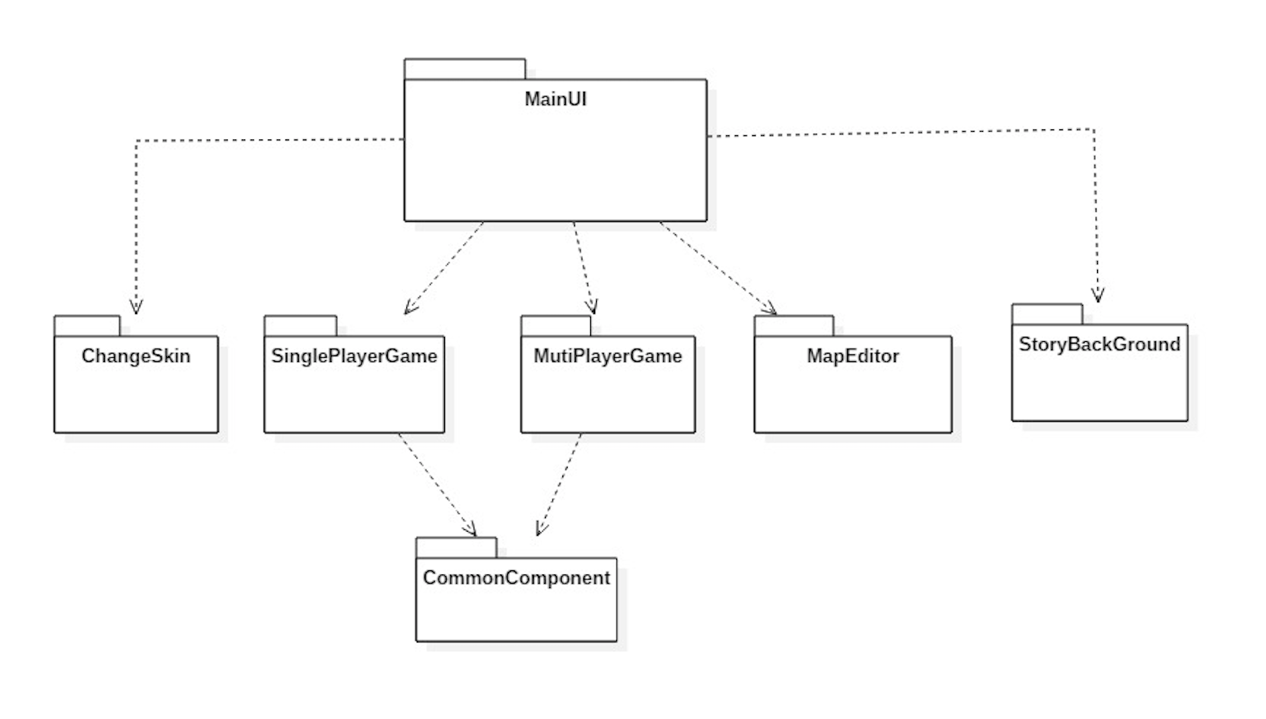


图4-2系统架构图

1. 子系统
2. UI子系统。

功能：负责与用户的交互

服务：UI层主要提供各种操作按钮、文本提示等进行信息的显示和与用户的交互的功能。

public String getName()

public String getPassword()

public Set getSkinSet()

public void createRegisterationPage()

public RegisterationPage destroyRegisterationPage()

public bool validatePersonInfo()

public UserInfoControl createUser()

public bool accessUser(UserInfoController ctrl)

public void giveNotice()

public DisplaySetOfSkin createSetOfSkin()

public void selectSkin()

public void destroySkin(DisplayOfSkin disOfSki)

public void giveNotice()

public StoryBox createStoryBox()

public void destroyStoryBox(StoryBox sb)

public bool validateClick()

public void enterSubsystem()

public void exit()

public ChangeSkinControl createChangeSkinControl()

public UserInfoControl submitInfo(String name, String data)

public MainPageControl createMainPageControl()

public void clickSubsystem()

public ReadStoryControl createReadStoryControl()

类图：

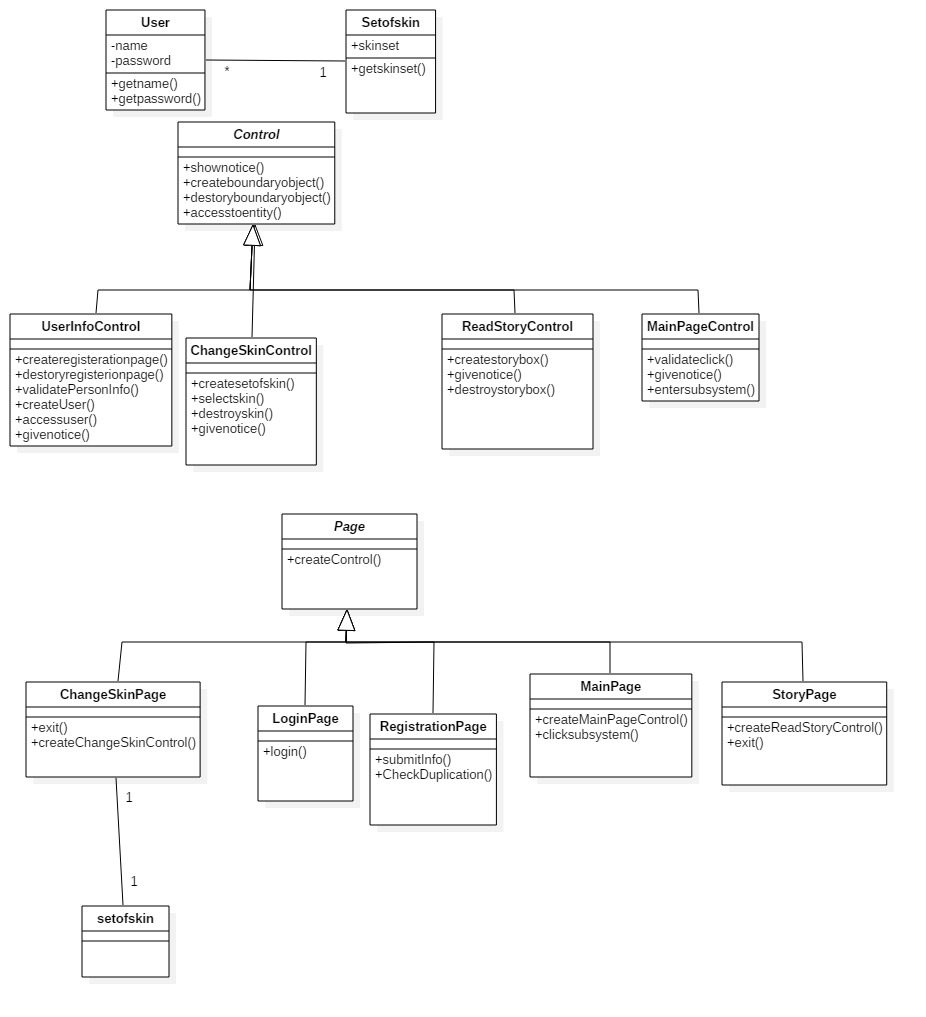


图4-3 UI子系统类图

组件图：

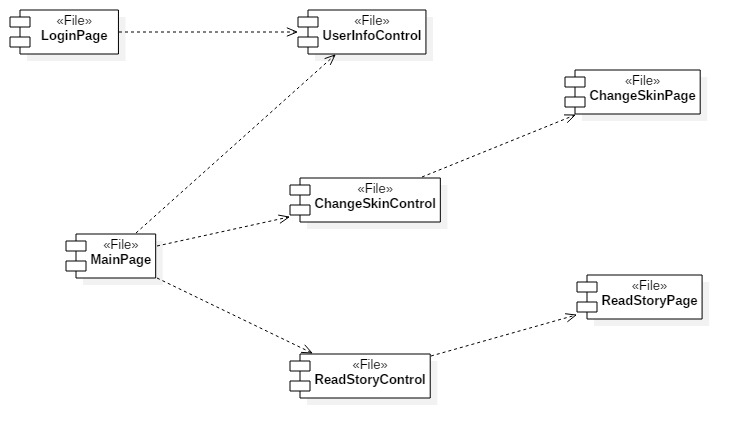


图4-4 UI子系统组件图

1. SingleGame子系统。

功能：负责单人游戏，控制单人游戏的开始、结束、及单人游戏的AI。

服务：

public Vector2 detectCollision(Vector2 v, double x)

public String getResult()

public CharacterType getType()

public bool testPickup(Vector2 v, double x)

public bool hasDirection()

public void userPowerUp()

public void switchScene(Scene scene)

类图：

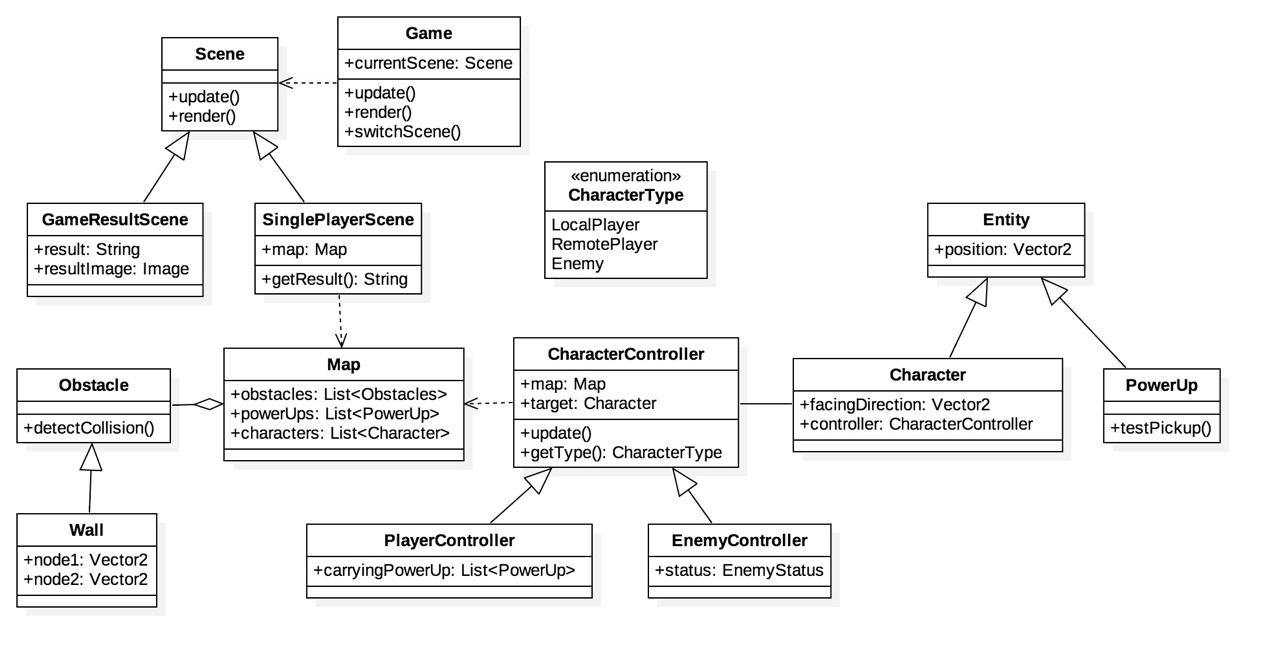


图4-5 SingleGame子系统类图

组件图：

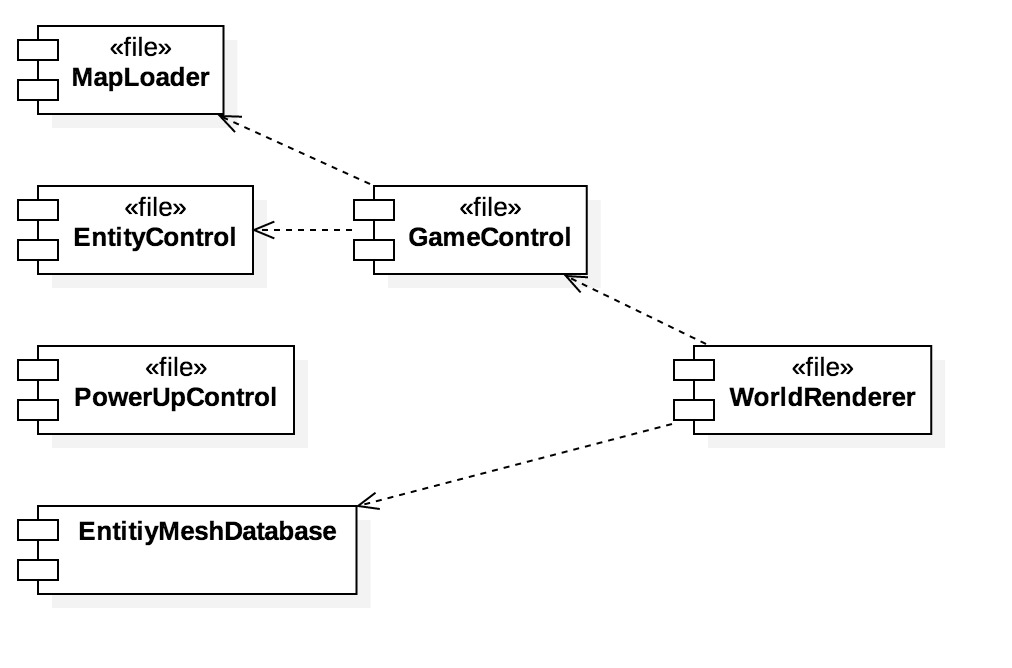


图4-6 SingleGame子系统组件图

1. MultiPlayer子系统

功能：负责多人游戏，控制多人游戏的开始、结束、及多人游戏的房间选择

服务：

public void subscribe(Observer ob)

public void unsubscribe(Observer ob)

public void update(Subject subj)

public void notifyAll()

public void makeChange()

public void addPlayer(Player p)

public void notifyPlayer()

public void setGroup(Group g)

public void getGroup()

public void setState()

public void getMove()

public void createMission()

public void updateMission()

public void createGroup()

public void createPlayer()

public void onSerializeNetwork()

public void updatePlayer(Player p)

public bool isOver(Mission m)

public void update()

public void getInfo()

类图：

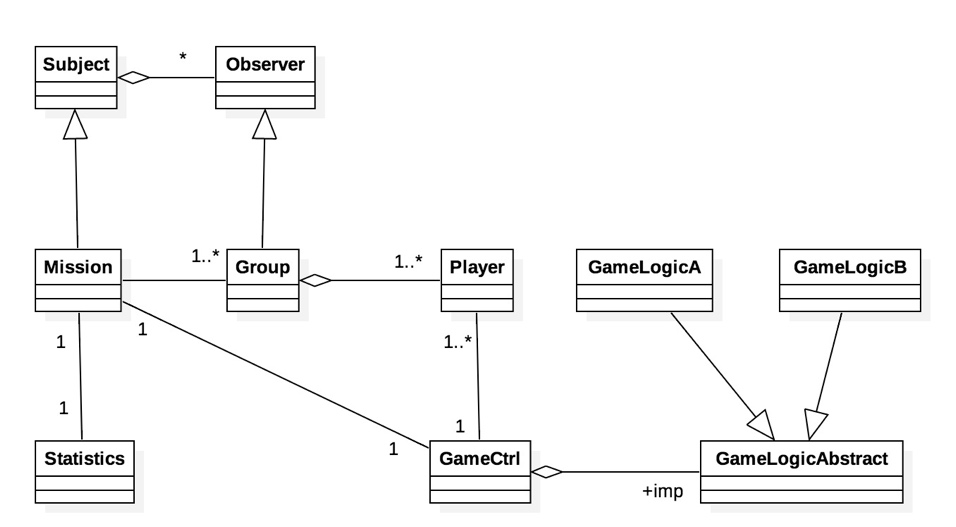


图4-7 MultiPlayer子系统类图

组件图：

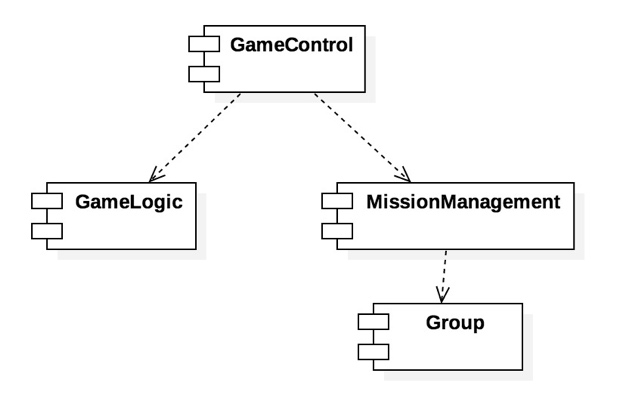


图4-8 MultiPlayer子系统组件图

1. MapEditor子系统

public MapFile createMap()

public void saveMap(MapFile)

public MapFile saveMap(Path path, String name)

public void openMap(MapFile map)

public void createElement(Element e)

public void changeElement(Element element, Attribute atti, Value val)

public void changePosition(Element element, Position pos)

public void delete(Element element)

public void undo()

public void showMap()

public void changeSight()

public void getUserOperation()

类图：

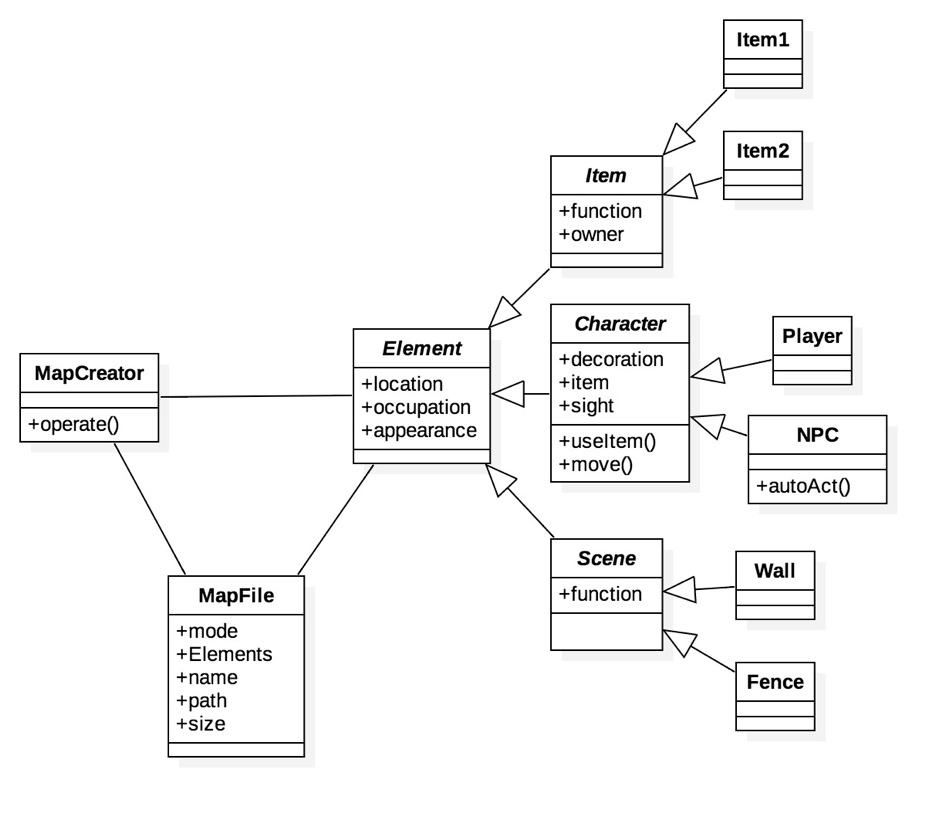


图4-9 MapEditor子系统类图

组件图：

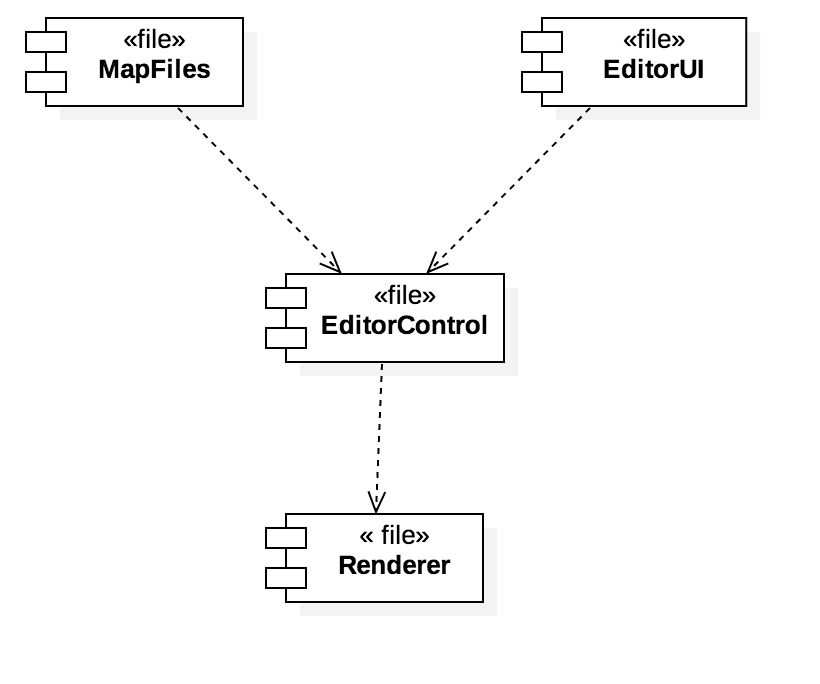


图4-10 MapEditor子系统组件图

1. 用例实现

本系统中的核心用例为MultiPlayer。

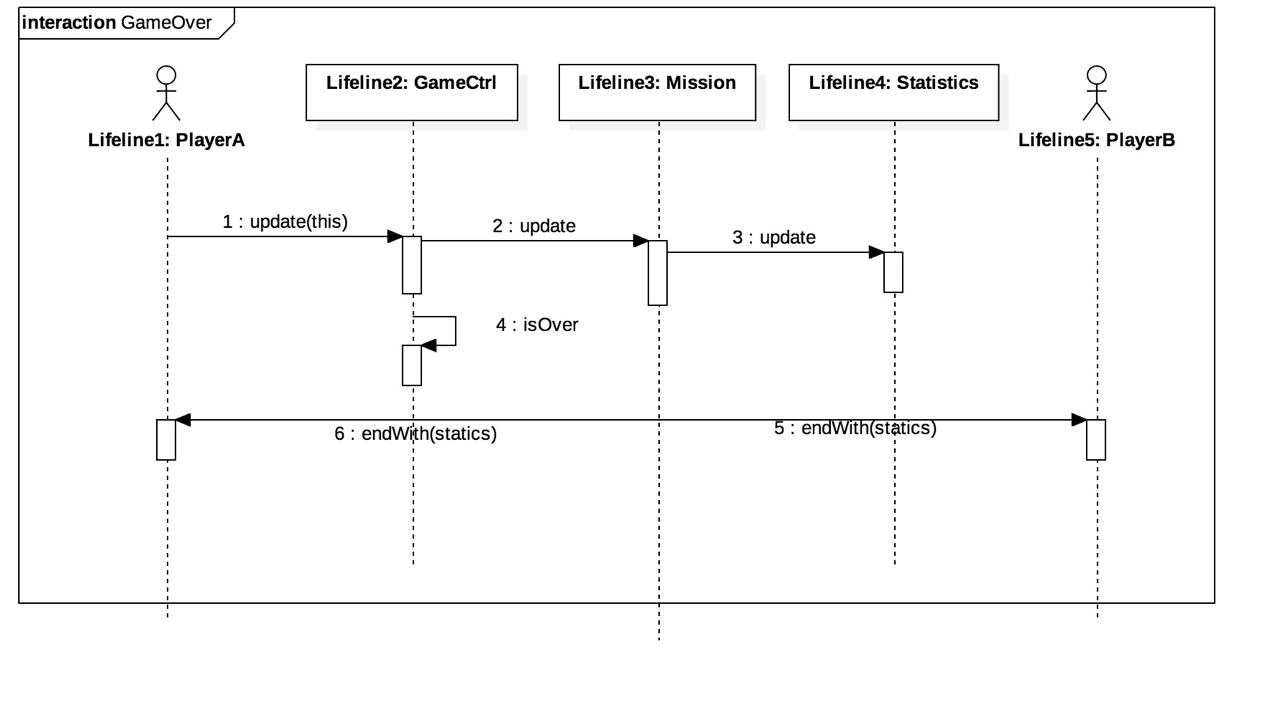


图4-11 Multiplayer核心用例图

1. 子系统协作

核心用例MultiPlayer的子系统协作图如下所示。

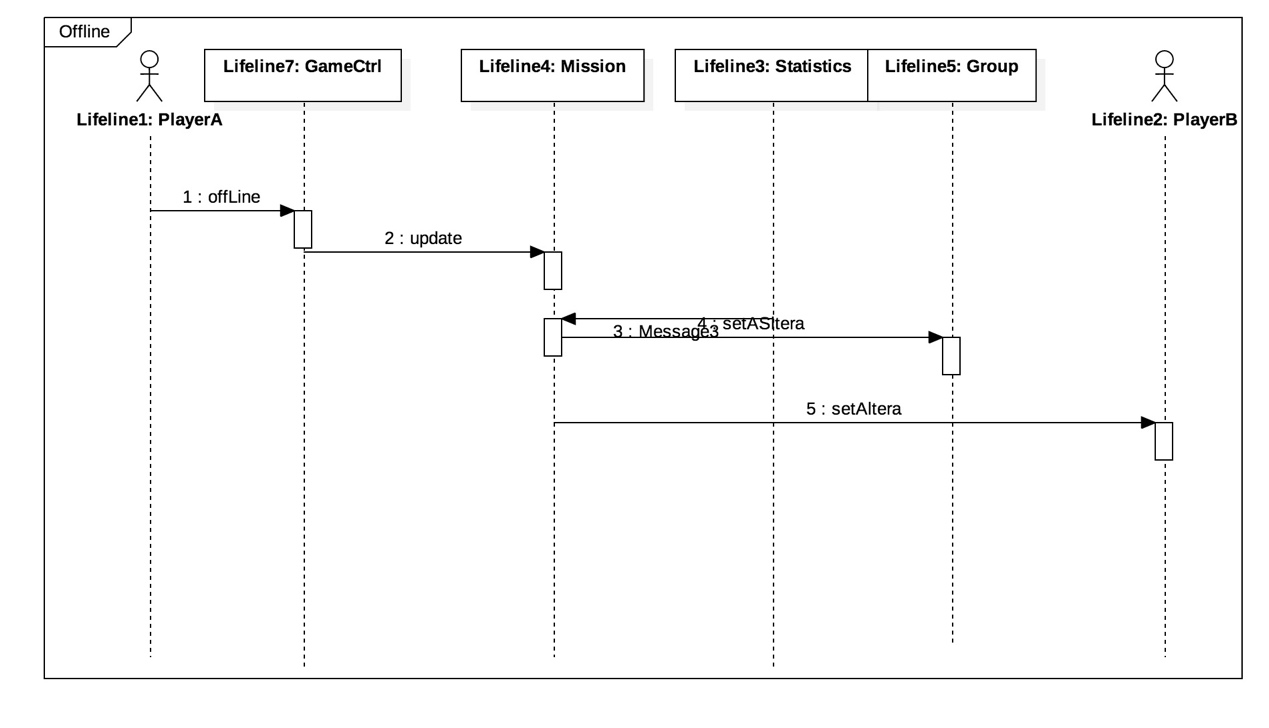


图4-12 子系统协作图

## 4.4系统运行视图

在客户端，为了保证用户体验，UI线程只负责边界对象与用户的交互。同时对每一个控制器单独启用一个线程。同时对于通信模块，也启动一个线程。这样的设计能够保证前台、功能实现、通信不会阻塞应用的运行。

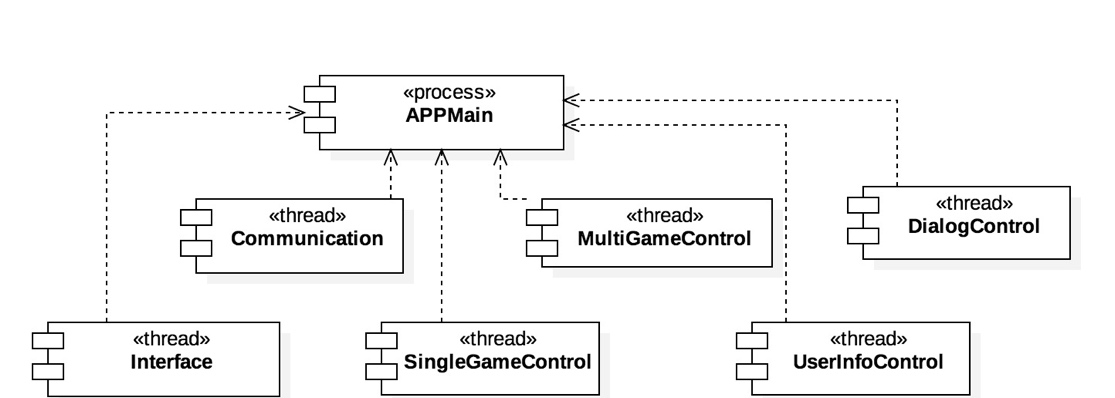


图4-13 子系统运行视图

## 4.5系统实现视图

1. 系统开发环境：

开发环境：Unity。

开发语言：c#

1. 系统开发模型

本系统中按照组件图进行了相应的软件文件定义，因此，软件开发模型与前节组件图一致。

## 4.6系统物理视图

硬件配置要求：

1. User machine：一台配有1GHz双核处理器及1GB内存（或以上）的手机。为保证较为良好的用户体验，此为流畅运行大多数安卓应用程序的最低配置。
2. 网络：30Mbps的互联网络。由于程序与服务器交互不频繁，因此选用适当的带宽。

## 4.7 边界条件设计

通过定义启动、关闭用例来考虑边界条件。

1. 启动App
2. 用户在客户端上点击应用程序图标

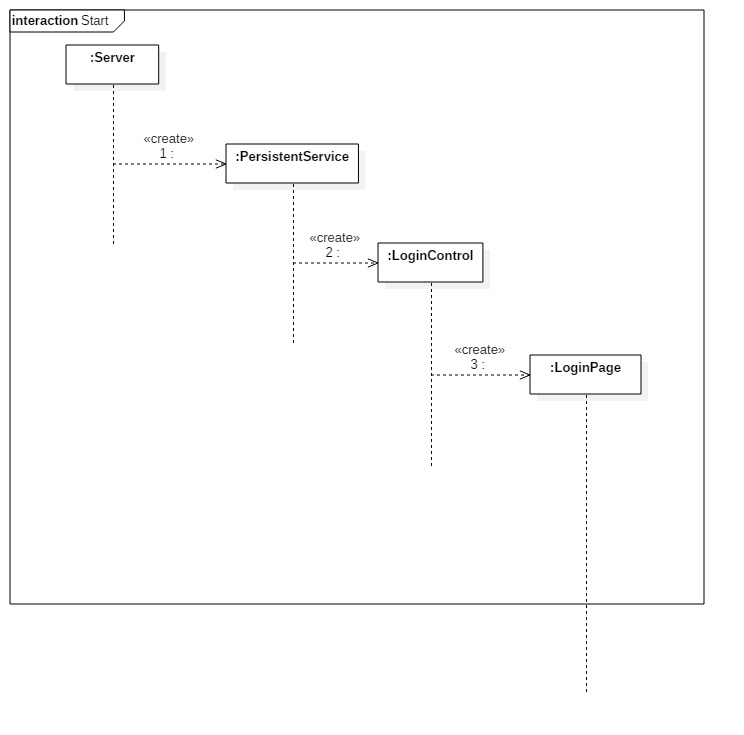


图4-14 系统启动流程图

1. 启动PersistentService对象
2. 启动LoginControl对象
3. 启动LoginPage对象
4. 关闭App

关闭部分与启动App部分相反。

## 4.8数据管理设计

由于本系统带有地图信息及玩家信息，基于文件的系统无法适应，因此我们选择使用关系数据库存储信息。

1. 地图元素表MapElement

表4-1地图元素表MapElement Form

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 说明 | 数据类型 | 允许为空 | 主键 | 备注 |
| 1 | elementID | 元素ID | long | N | Y |  |
| 2 | name | 元素名称 | string | N | N |  |
| 3 | model | 模型原语（primitive） | string | N | N |  |
| 4 | positionX | 位置x | float | N | N |  |
| 5 | positionY | 位置y | float | N | N |  |
| 6 | sizeX | x轴拉伸 | float | N | N |  |
| 7 | sizeY | y轴拉伸 | float | N | N |  |
| 8 | sizeZ | z轴拉伸 | float | N | N |  |

1. 玩家信息表Player

表4-2玩家信息表Player Form

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 说明 | 数据类型 | 允许为空 | 主键 | 备注 |
| 1 | playerID | 玩家ID | long | N | Y |  |
| 2 | name | 玩家姓名 | string | N | N |  |
| 3 | scoreLocal | 单击游戏时得分 | long | N | N |  |
| 4 | stage | 玩家游戏阶段 | long | N | N |  |
| 5 | scoreRemote | 多人游戏时得分 | long | N | N |  |
| 6 | model | 模型原语名称 | stirng | N | N |  |
| 7 | money | 游戏内部货币 | long | N | N |  |

1. 地图物品Item

表4-3地图物品Item Form

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 说明 | 数据类型 | 允许为空 | 主键 | 备注 |
| 1 | itemID | 物品ID | long | N | Y |  |
| 2 | itemName | 物品名称 | string | N | N |  |
| 3 | model | 模型原语名称 | string | N | N |  |

实体类设计类图如下图所示：

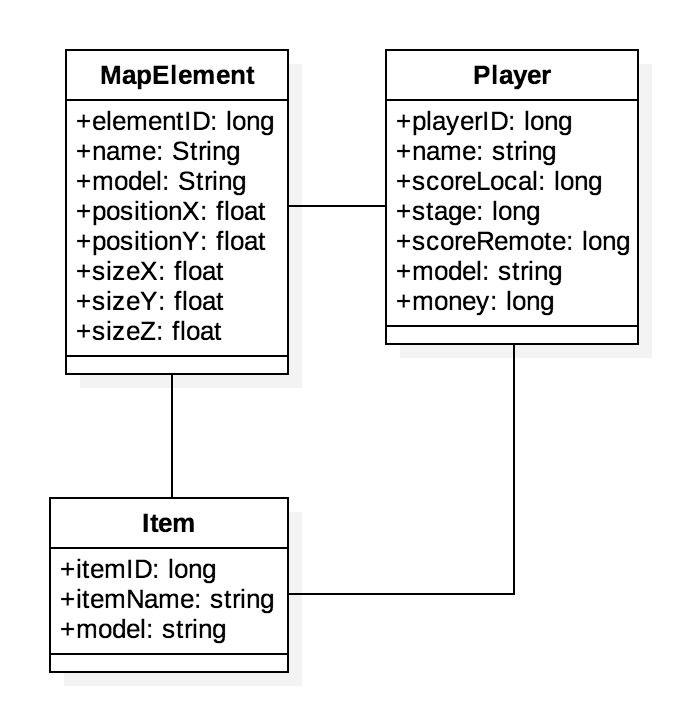


图4-15实体类设计类图

## 4.9 其他设计

（1）访问控制和安全设计

登录用户在使用他们的权限之前需要先提供用户名和登录密码，经验证是匹配的菜通过认证。

（2）可靠性设计

为了提高系统的可靠性，我们考虑采取冗余与备份的方案。设置备用服务器，在主服务器发生故障时将系统运行切换至备用服务器上；数据库冗余并备份，防止数据的丢失与意外更改。