G16

详细设计



**小组名称： 明德1-518小组**

**项目名称： 模拟三国**

**组长： 王华怿**

**组员： 王仕杰，吴帅毅**

**参考标准： 国标.计算机软件文档编制规范**

**标准号： GB/T 8567-2006 GB/T 8567-88**

<https://blog.csdn.net/vop444/article/details/53183449>

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 代码版本 | 更新内容 | 更新日期 |
| 1 | 无 | 编写 初始详细设计 | 2019.4.30 |
| 2 | 无 | 增加补充内容 | 2019.5.5 |
| 3 | V0.0.1 | 修改 根据评审 | 2019.5.18 |
| 4 | V0.1.0 | 增加 代码清单 | 2019.5.27 |
| 4 | V0.4.0 | 修改 根据实际进度 | 2019.6.7 |

注：红色为当前版本。

目录

[一 引言 1](#_Toc9927400)

[1 目的 1](#_Toc9927401)

[2 背景 1](#_Toc9927402)

[3 定义 2](#_Toc9927403)

[4 文档格式 2](#_Toc9927404)

[5 预期读者和阅读建议 2](#_Toc9927405)

[6 参考文献 3](#_Toc9927406)

[二 需求概述 4](#_Toc9927407)

[1 产品功能 4](#_Toc9927408)

[2 运行环境 4](#_Toc9927409)

[3 性能需求 4](#_Toc9927410)

[4 设计思路 5](#_Toc9927411)

[流程图 5](#_Toc9927412)

[Jackson图 5](#_Toc9927413)

[HIPO图 8](#_Toc9927414)

[5 设计实现限制 9](#_Toc9927415)

[6 假设和依赖 10](#_Toc9927416)

[三 外部接口需求 11](#_Toc9927417)

[1 用户接口 11](#_Toc9927418)

[2 硬件接口 15](#_Toc9927419)

[3 软件接口 15](#_Toc9927420)

[4 外部接口 15](#_Toc9927421)

[5 内部接口 15](#_Toc9927422)

[四 模块设计 16](#_Toc9927423)

[1 各模块概述 16](#_Toc9927424)

[2 各模块接口 17](#_Toc9927425)

[3 运行模块的组成 17](#_Toc9927426)

[五 数据设计 17](#_Toc9927427)

[1 数据字典 17](#_Toc9927428)

[2 ER图 19](#_Toc9927429)

[3 数据流图 19](#_Toc9927430)

[4 PDL设计 20](#_Toc9927431)

[六 代码清单 23](#_Toc9927432)

[1 前端部分： 23](#_Toc9927433)

[2 后端部分： 35](#_Toc9927434)

[七 验收说明 47](#_Toc9927435)

# 引言

## 目的

本阶段要在系统需求的基础上，对项目做进一步的总体说明，它主要解决一下三方面的问题：

A．确定软件的结构---有哪几个模块组成,调用关系，各模块间的接口。

B．硬件端口分配设计---确定IO端口、和外设之间的资源分配。

C．文档的书写

## 背景

软件名称：模拟三国

任务提出者：杨枨

开发者：G16，明德1-518小组

组长王华怿：项目经理、项目技术负责人

组员吴帅毅：开发经理、实施和测试工程师

组员王仕杰：开发工程师、实施和测试工程师

用户：杨枨老师，游戏主要面向战略策略游戏爱好者和三国迷

软件平台：该游戏以微信小程序的方式发布

## 定义

三国题材，SLG ([Sim](https://www.baidu.com/s?wd=Sim&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)ulation Game,模拟游戏)，策略，模拟，经营，战略，回合制，微信小程序，跨平台，简单易上手。

## 文档格式

1. 文档编辑顺序遵循IEEE 830相关标准。
2. 标题使用黑体，初号，加粗。
3. 一级标题使用宋体，二号，加粗。
4. 二级标题使用等线（中文标题），三号，加粗。
5. 正文使用等线（中文正文），小四，首行缩进2字符。
6. 目录和附录同以上格式。

## 预期读者和阅读建议

1. 审查人员（老师、助教、组长）：查阅此文档。
2. 项目经理（组长）：根据此文档进行项目的设计和项目管理。
3. 程序员（全体组员）：按照此文档的要求编写功能。
4. 测试员（全体组员）：根据本文档用例对产品进行评测。
5. 用户：了解此产品的功能和性能，并配合分析人员讨论。

## 参考文献

[1]聂明.游戏开发导论[M].西安电子科技大学出版社,2009.

[2]张海藩.软件工程导论[M].清华大学出版社,1996:1-73.

[3] GB/T 8567-2006,GB/T 8567-88. 计算机软件文档编制规范[S].

# 需求概述

## 产品功能

本产品为模拟策略类游戏，可以快速简单地进行游戏，其主要功能有：游戏存档并加载、自己国家的资源管理与升级、与其他国家的对战、好友排行榜功能

## 运行环境

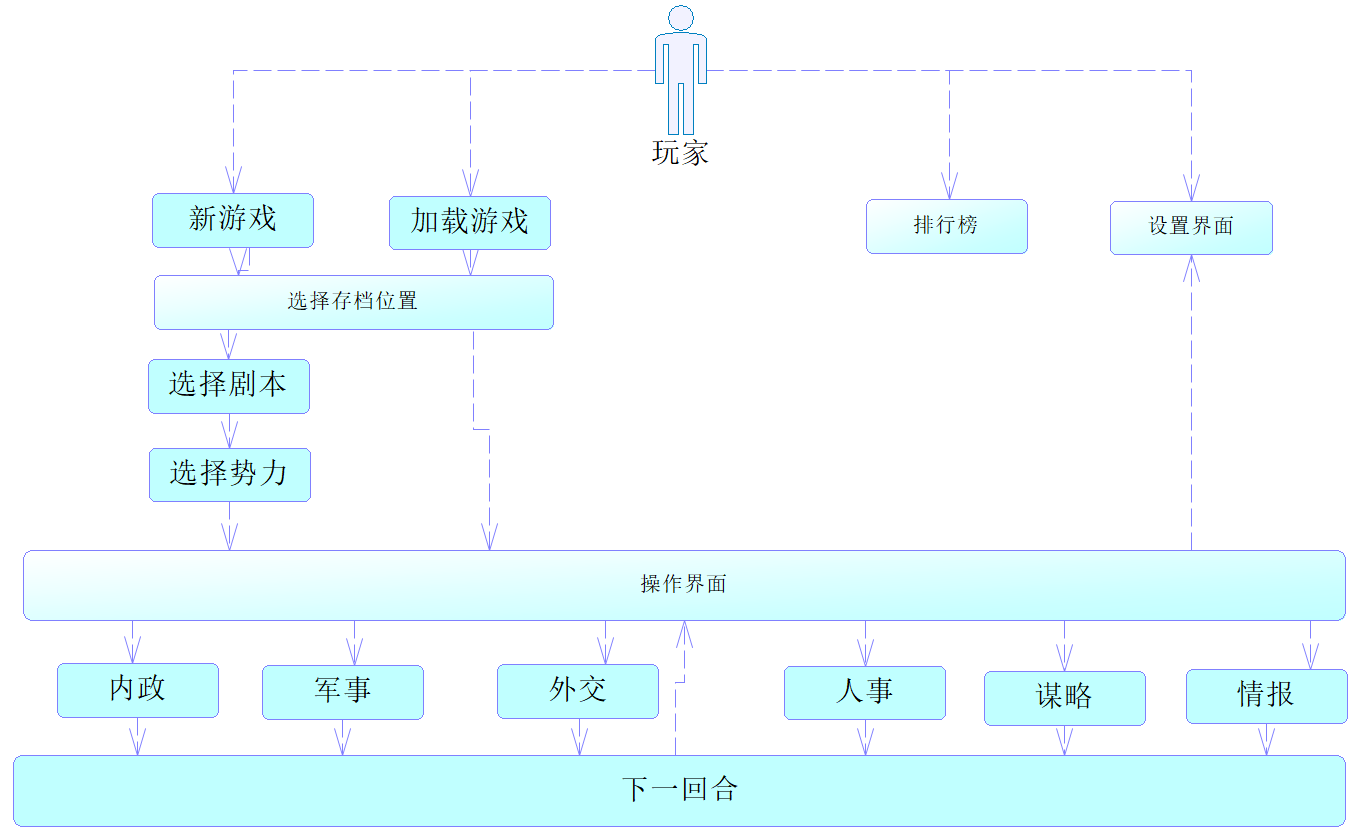
1. 客户端：微信较高版本
2. 服务器端：微信小程序自带服务器
3. 客户机端：智能手机

## 性能需求

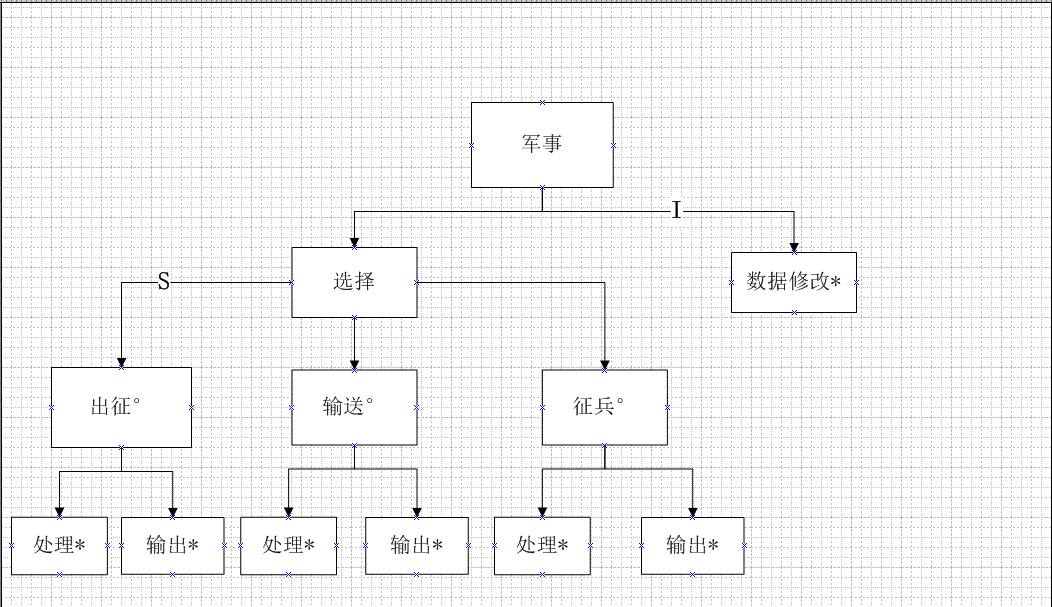
快速的运行游戏不卡顿，其性能主要根据游戏引擎和客户端硬件水平。

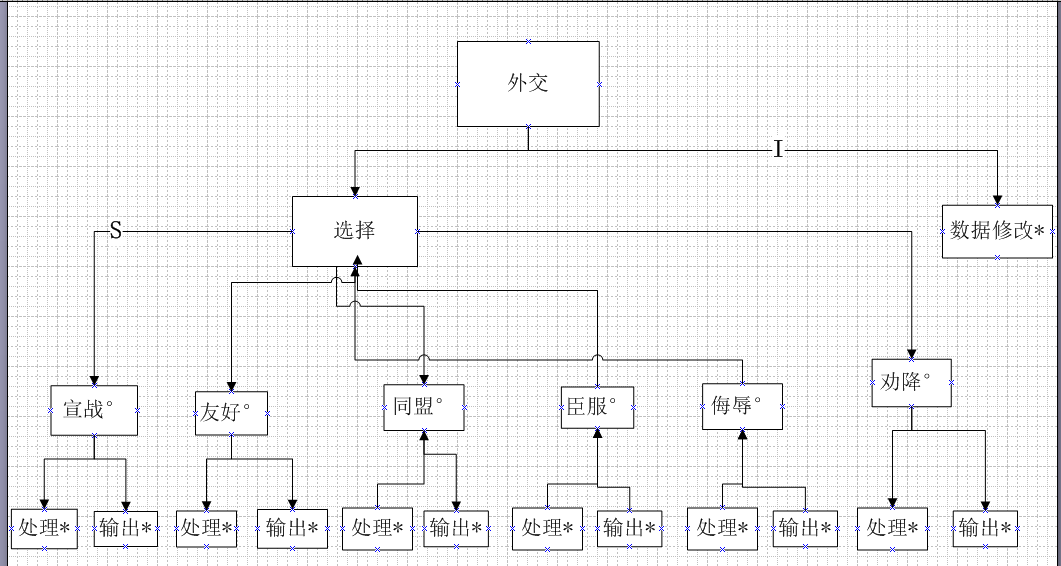
## 设计思路

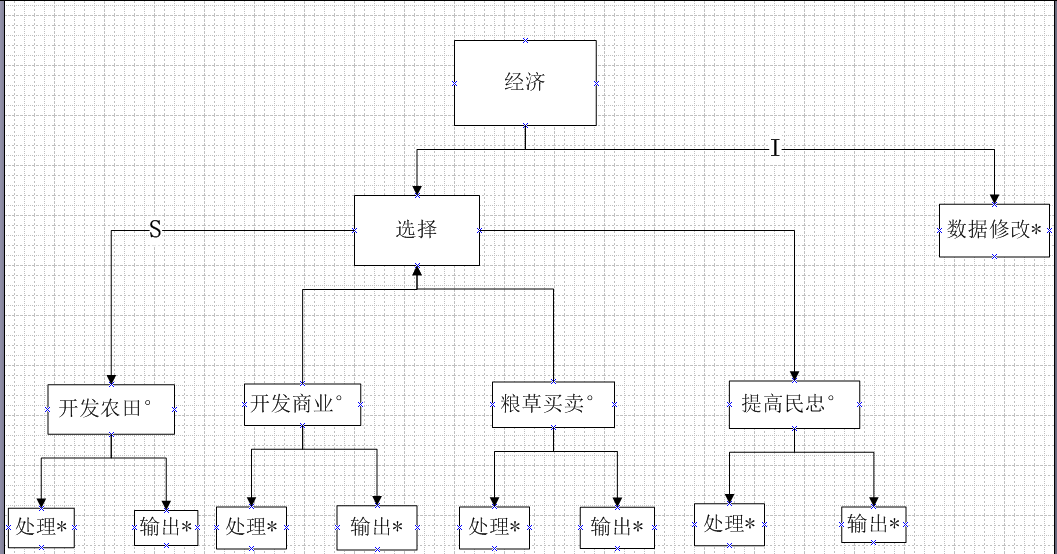
### 流程图

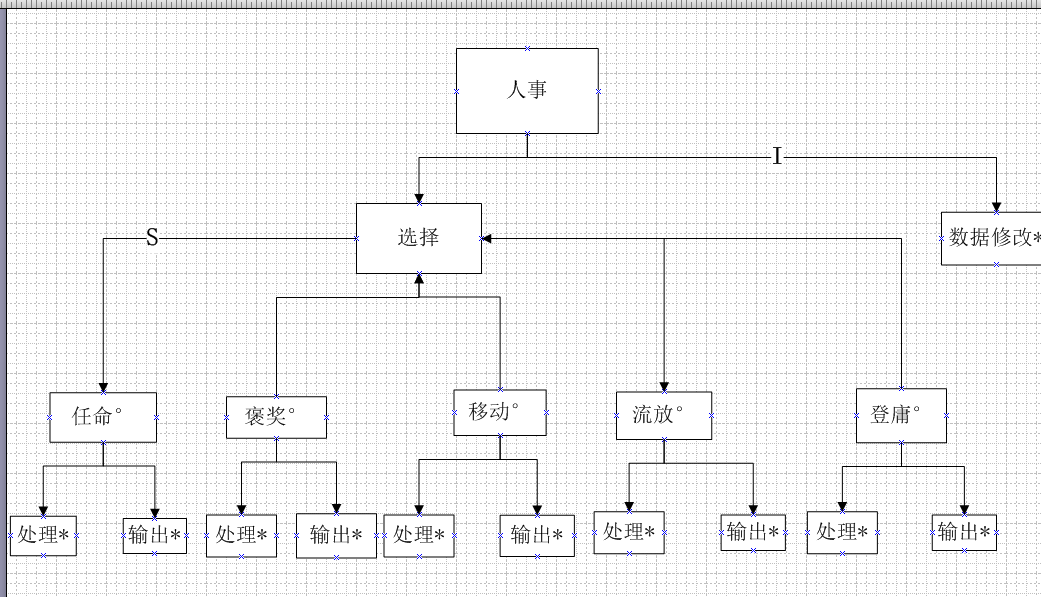


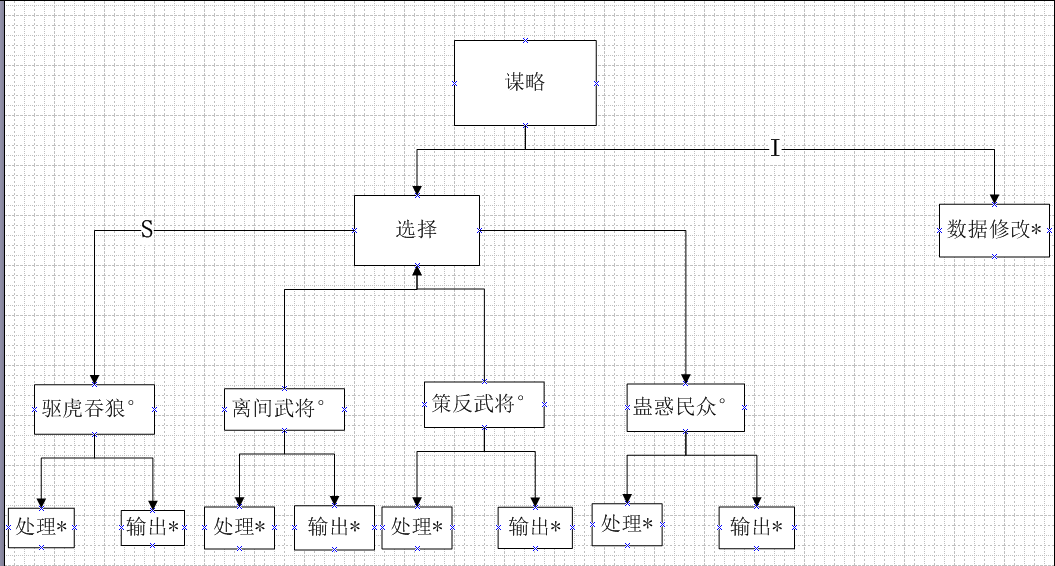
### Jackson图

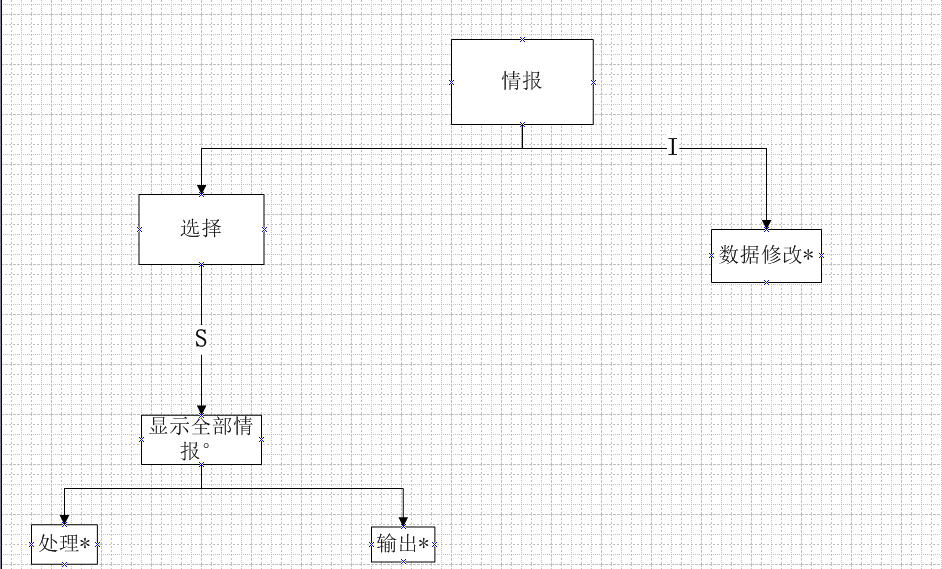




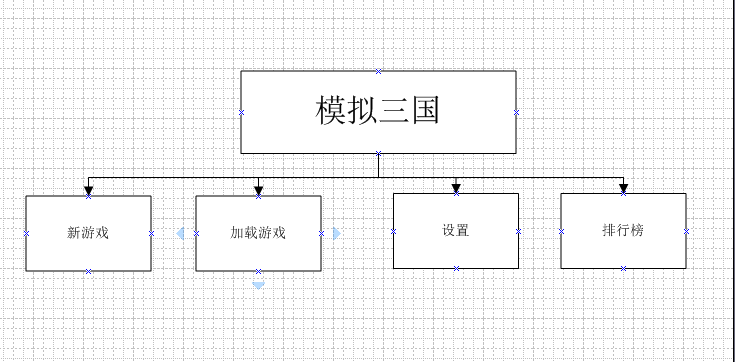


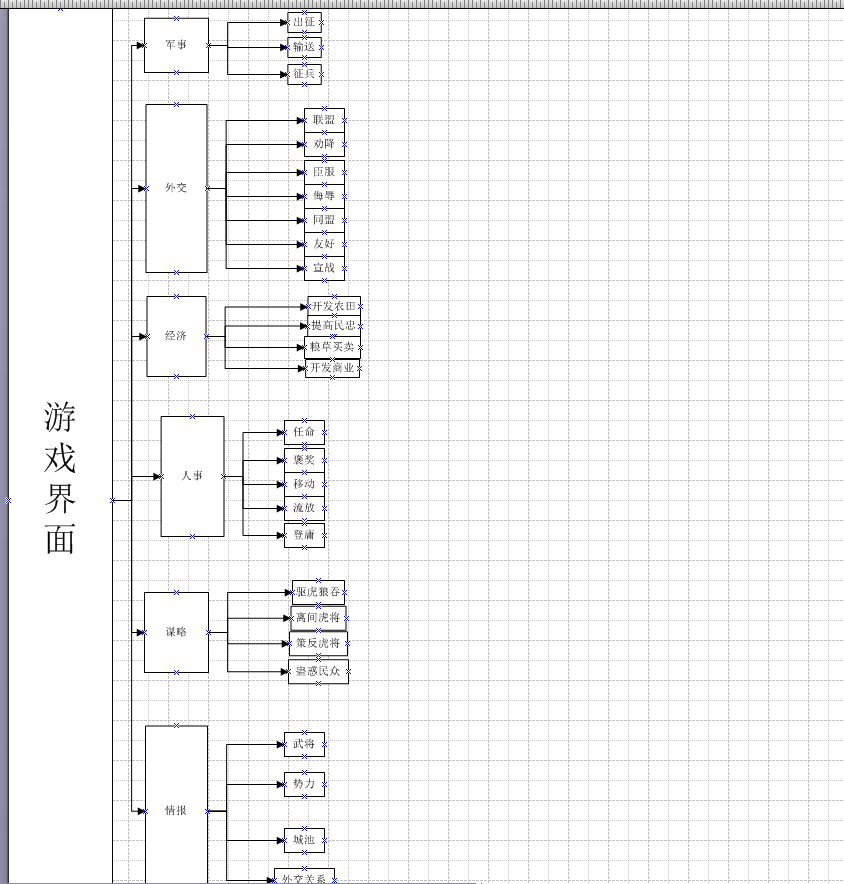






### HIPO图





## 设计实现限制

A. 游戏有一定数量的玩家，可以在好友排行榜上进行比较

B. 游戏可以按照既定的规则和机制进行

C. 游戏存档与加载需同步

D. 不同的历史人物需有不同的属性，各个历史人物同各个不同的势力需匹配好

## 假设和依赖

建议开发和运行软件的寿命最短为2年，经费来源为小组G16，使用限制为手机微信小程序，符合法律和政策反面所有条件，运行环境与之前的“运行环境”相同，开发环境由开发方提供，可利用的信息来自问卷星调查数据与互联网上的信息。

1. 只有玩家进行存档，下次加载必须在存档处继续游戏
2. 玩家成功结算后，总分记入排行榜，其好友可以同时看见
3. 玩家在获得更好的分数后，排行榜上的数据应该及时跟新

# 外部接口需求

## 用户接口

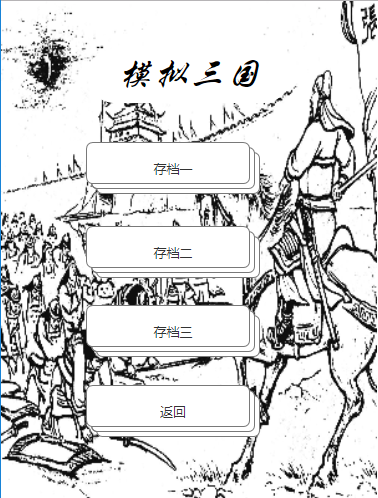
|  |  |  |
| --- | --- | --- |
| 命令 | 语法 | 信息回复 |
| 开始游戏 | Click事件，点击 | 进入游戏 |
| 查询排行榜 | Click事件，点击 | 出现排行榜内容 |
| 返回 | Click事件，点击 | 返回上级菜单或主界面 |
| 设置 | Click事件，点击 | 对音量等进行设置 |
| 游戏内各类操作命令 | Click事件，点击 | 游戏内功能选择 |

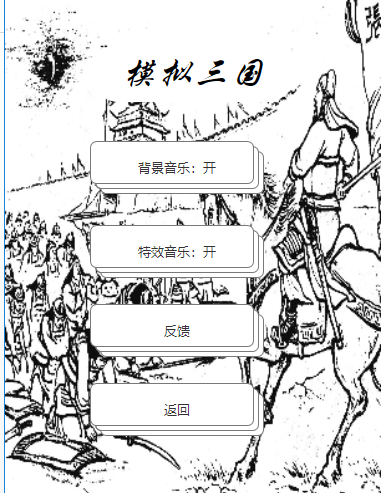


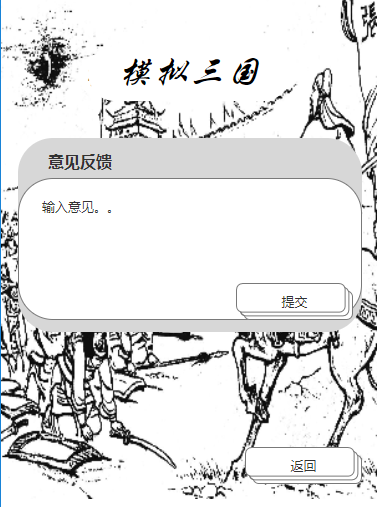












## 硬件接口

Linux操作系统

## 软件接口

微信小程序

## 外部接口

本软件不需要特定的外部接口进行支撑，只需要可以运行微信的手机便可以运行这个游戏。

## 内部接口

系统内部与数据库接口为SQL链接

客户端通过配置数据源与服务器建立连接

# 模块设计

## 各模块概述

|  |  |  |  |
| --- | --- | --- | --- |
| **一级模块** | **二级模块** | **三级模块** | **模块描述** |
| 新游戏 | 存档选择 | 空白新增 | 在未有存档的地方新增存档 |
| 覆盖原有 | 在已有存档的地方覆盖存档 |
| 剧本选择 | | 选择游玩的剧本 |
| 势力选择 | | 选择游玩的势力 |
| 加载游戏 | 存档选择 | 存档读取 | 读取选择的存档 |
| 存档删除 | 删除选择的存档 |
| 游戏界面 | 军事 | | 出征，输送，征兵 |
| 经济 | | 开发农田，开发商业，粮草买卖，提高民忠,提高城防 |
| 人事 | | 褒奖，移动，流放，登庸 |
| 谋略 | | 驱虎吞狼，离间武将，降低商业，降低农业 |
| 情报 | | 显示总体的情报，例如武将、城池、势力等 |
| 设置 | | 同下【设置】模块 |
| 下一回合 | | 进行下一回合的推演 |
| 排行榜 | | | 查看本人在好友和全局的排名和分数 |
| 设置 | | | 对音量等进行设置 |

## 各模块接口

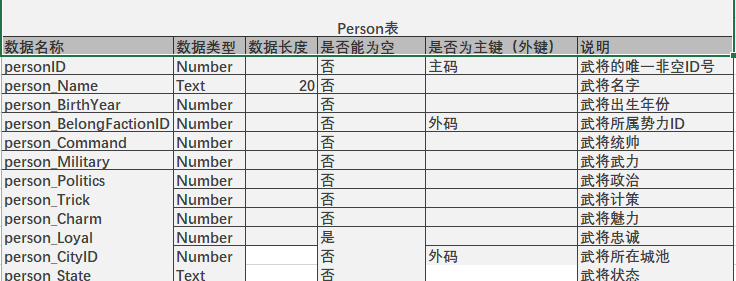
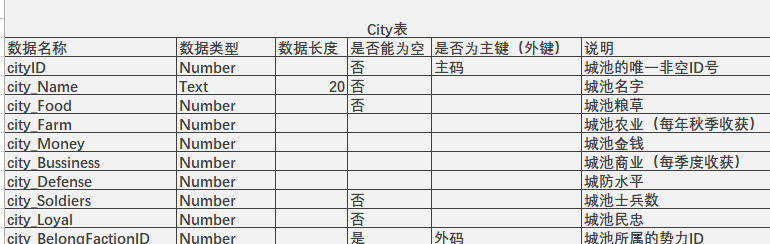
未定。

## 运行模块的组成

1. 系统分成几个相对独立的模块，但这些模块都进行集中式管理。
2. 分层的模块划程序设计思想，整个系统采用模块化结构设计，作为应用程序有较强的课操作性和扩展性。
3. 合理的数据流设计，在应用系统设计中，相对独立的模块间以数据流相互连接，使各模块间的耦合性较低，方便系统运行，提高系统的安全性。

# 数据设计

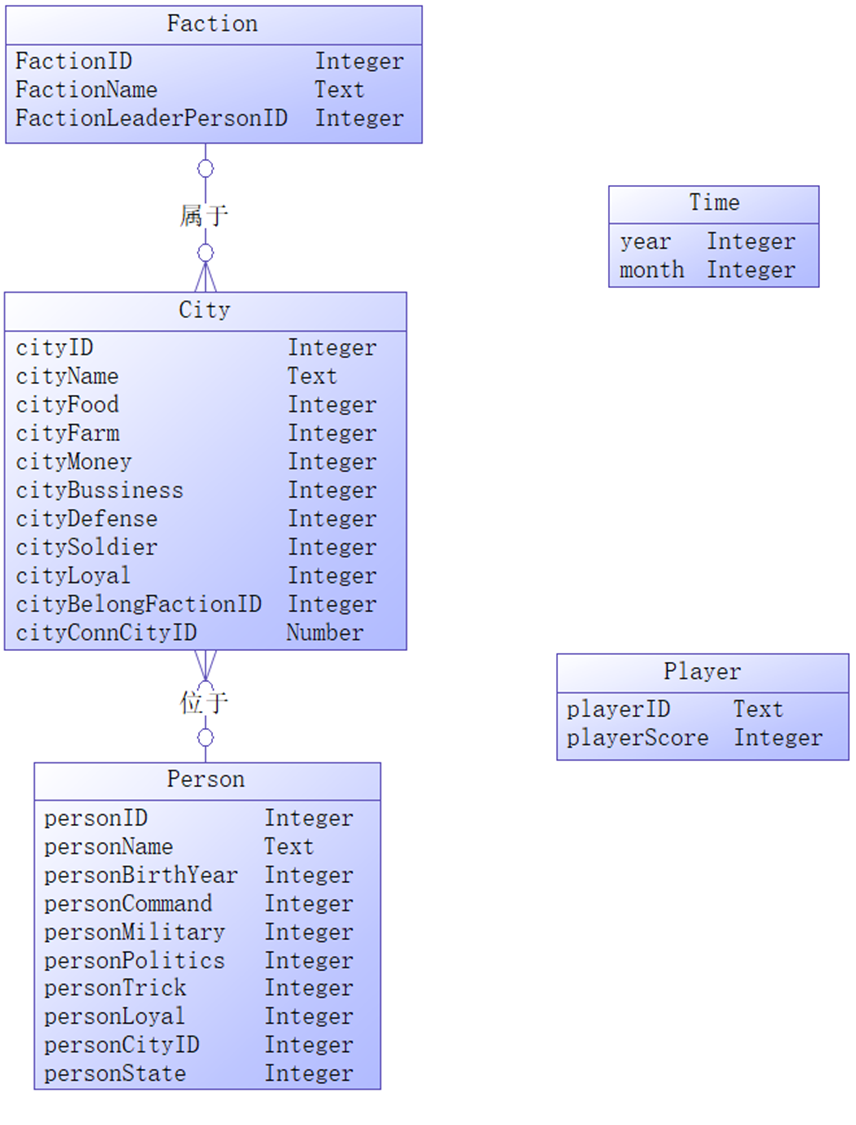
## 数据字典







## ER图



## 数据流图



## PDL设计

**主界面**

全局变量：用户ID;

Main(){

While（1）{

选择{

1.新游戏()

2.读取存档()

3.排行榜()

4.设置()

}

}

退出

}

**新游戏**

New(){

剧本号=剧本选择（）

势力号=势力选择（）

游戏数据库操作（剧本号，势力号）

进入游戏（）

}

**读取游戏**

Load(){

读取数据从服务器到本地（用户id）//包括所有存档数据

存档号=选择存档（）

加载存档到游戏数据库（存档号）

进入游戏()

}

**排行榜**

Phb(){

读取数据从服务器（）

列出玩家所在（用户ID）

}

**设置**

Setting(){

While(1)

选择{

1.

2.

3.

0.返回Main()

}

}

**游戏**

Game(){

While(1){

for(ID=1;ID<=MAXID;ID++){

if(AI)

AIAUTO();

Else if(Player)

Player();

}

Save();

}

}

# 代码清单

## 前端部分：

游戏开始按钮控制脚本

export default class Game\_con extends Laya.Script {

constructor() {

super();

/\*\* @prop {name:money, tips:"金钱", type:Int, default:1000}\*/

let money = 1000;

/\*\* @prop {name:numType, tips:"数字类型示例", type:Number, default:1000}\*/

let numType = 1000;

/\*\* @prop {name:strType, tips:"字符串类型示例", type:String, default:"hello laya"}\*/

let strType = "hello laya";

/\*\* @prop {name:boolType, tips:"布尔类型示例", type:Bool, default:true}\*/

let boolType = true;

}

onClick(){

Laya.Scene.open("Game/gameChose.scene")

}

}

游戏选择剧本控制按钮脚本

export default class Gamechose extends Laya.Script {

constructor() {

super();

/\*\* @prop {name:intType, tips:"整数类型示例", type:Int, default:1000}\*/

let intType = 1000;

/\*\* @prop {name:numType, tips:"数字类型示例", type:Number, default:1000}\*/

let numType = 1000;

/\*\* @prop {name:strType, tips:"字符串类型示例", type:String, default:"hello laya"}\*/

let strType = "hello laya";

/\*\* @prop {name:boolType, tips:"布尔类型示例", type:Bool, default:true}\*/

let boolType = true;

}

onClick(){

Laya.Scene.open("Game/gameStartchose.scene")

}

}

游戏加载按钮控制脚本

export default class gameLoad extends Laya.Script {

constructor() {

super();

/\*\* @prop {name:intType, tips:"整数类型示例", type:Int, default:1000}\*/

let intType = 1000;

/\*\* @prop {name:numType, tips:"数字类型示例", type:Number, default:1000}\*/

let numType = 1000;

/\*\* @prop {name:strType, tips:"字符串类型示例", type:String, default:"hello laya"}\*/

let strType = "hello laya";

/\*\* @prop {name:boolType, tips:"布尔类型示例", type:Bool, default:true}\*/

let boolType = true;

}

onClick(){

Laya.Scene.open("Game/gameStart1.scene")

}

}

游戏势力选择进入游戏控制脚本

export default class gameStartchose extends Laya.Script {

constructor() {

super();

/\*\* @prop {name:intType, tips:"整数类型示例", type:Int, default:1000}\*/

let intType = 1000;

/\*\* @prop {name:numType, tips:"数字类型示例", type:Number, default:1000}\*/

let numType = 1000;

/\*\* @prop {name:strType, tips:"字符串类型示例", type:String, default:"hello laya"}\*/

let strType = "hello laya";

/\*\* @prop {name:boolType, tips:"布尔类型示例", type:Bool, default:true}\*/

let boolType = true;

}

onClick(){}//触发点击转换界面

loadScene(){}//进入加载界面

}

返回按钮控制脚本

export default class gameReturn extends Laya.Script {

constructor() {

super();

/\*\* @prop {name:intType, tips:"整数类型示例", type:Int, default:1000}\*/

let intType = 1000;

/\*\* @prop {name:numType, tips:"数字类型示例", type:Number, default:1000}\*/

let numType = 1000;

/\*\* @prop {name:strType, tips:"字符串类型示例", type:String, default:"hello laya"}\*/

let strType = "hello laya";

/\*\* @prop {name:boolType, tips:"布尔类型示例", type:Bool, default:true}\*/

let boolType = true;

}

onClick(){

Laya.Scene.open("Game/gameStart.scene")

}

}

游戏主界面逻辑代码

export default class gameMainUi extends Laya.Scene {

constructor() {

super();

/\*\* @prop {name:intType, tips:"整数类型示例", type:Int, default:1000}\*/

let intType = 1000;

/\*\* @prop {name:numType, tips:"数字类型示例", type:Number, default:1000}\*/

let numType = 1000;

/\*\* @prop {name:strType, tips:"字符串类型示例", type:String, default:"hello laya"}\*/

let strType = "hello laya";

/\*\* @prop {name:boolType, tips:"布尔类型示例", type:Bool, default:true}\*/

let boolType = true;

}

openMessage(){}//打开详细信息界面

openSetting(){}//打开设置界面

openMilitary(){}//打开军事系统

openInterior(){}//打开内政系统

openTactics(){}//打开策略系统

openPersonnal(){}//打开人事系统

openNext(){}//下一回合按钮控制

setCityMessage(){}//城池信息设置

setMessage(){}//详细信息界面控制

setMap(){}//放入地图并控制地图

}

游戏界面加载并转换游戏界面

export default class gameLoading extends Laya.Scene{

constructor() {

super();

/\*\* @prop {name:intType, tips:"整数类型示例", type:Int, default:1000}\*/

let intType = 1000;

/\*\* @prop {name:numType, tips:"数字类型示例", type:Number, default:1000}\*/

let numType = 1000;

/\*\* @prop {name:strType, tips:"字符串类型示例", type:String, default:"hello laya"}\*/

let strType = "hello laya";

/\*\* @prop {name:boolType, tips:"布尔类型示例", type:Bool, default:true}\*/

let boolType = true;

}

loadScene(){}//加载进入场景

open(){}//打开场景

close(){}//关闭完成后，调用此方法（如果有关闭动画，则在动画完成后执行）

}

排行榜设置脚本

export default class setRanking extends Laya.Script{

constructor() {

super();

/\*\* @prop {name:intType, tips:"整数类型示例", type:Int, default:1000}\*/

let intType = 1000;

/\*\* @prop {name:numType, tips:"数字类型示例", type:Number, default:1000}\*/

let numType = 1000;

/\*\* @prop {name:strType, tips:"字符串类型示例", type:String, default:"hello laya"}\*/

let strType = "hello laya";

/\*\* @prop {name:boolType, tips:"布尔类型示例", type:Bool, default:true}\*/

let boolType = true;

}

onAwake(){

setImage()//在组件加入舞台后进行赋值操作

}

}

## 后端部分：

1. 类常量设计及实现：

军队类属性的定义及实现

class Army{

constructor(armyID,armyMainID,armyAssisiant1ID,armyAssisiant2ID,armySoldier,armyFood,armyMoney){

this.armyID=armyID;

this.armyMainID=armyMainID;

this.armyAssisiant1ID=armyAssisiant1ID;

this.armyAssisiant2ID=armyAssisiant2ID;

this.armySoldier=armySoldier;

this.armyFood=armyFood;

this.armyMoney=armyMoney;

}

} //军队类属性的定义

set ArmyID(armyID)

get ArmyID() //军队ID的实现

set ArmyMainID( armyMainID)

get ArmyMainID() //军队主将ID的实现

set ArmyAssisiant1ID( armyAssisiant1ID)

get ArmyAssisiant1ID() //军队副将1ID的实现

set ArmyAssisiant2ID( armyAssisiant2ID)

get ArmyAssisiant2ID() //军队副将1ID的实现

set ArmySoldier( armySoldier)

get ArmySoldier() //军队士兵的实现

set ArmyFood( armyFood)

get ArmyFood() //军队兵粮的实现

set ArmyMoney( armyMoney)

get ArmyMoney() //军队财富的实现

城池类属性的定义及实现

class City{

constructor(cityID,cityName,cityFood,cityFarm,cityMoney,cityBussiness,cityDefense,citySoldier,cityLoyal,cityBelongFactionID){

this.cityID=cityID;

this.cityName=cityName;

this.cityFood=cityFood;

this.cityFarm=cityFarm;

this.cityMoney=cityMoney;

this.cityBussiness=cityBussiness;

this.cityDefense=cityDefense;

this.citySoldier=citySoldier;

this.cityLoyal=cityLoyal;

this.cityBelongFactionID=cityBelongFactionID;

} //城池类属性的定义

set CityID(cityID)

get CityID() //城池ID的实现

set CityName(cityName)

get CityName() //城池姓名的实现

set CityFood(cityFood)

get CityFood() //城池粮草的实现

set CityFarm( cityFarm)

get CityFarm() //城池农业的实现

set CityMoney( cityMoney)

get CityMoney() //城池财富的实现

set CityBussiness(cityBussiness)

get CityBussiness() //城池商业的实现

set CityDefense(cityDefense)

get CityDefense() //城池防御的实现

set CitySoldier( citySoldier)

get CitySoldier() //城池士兵的实现

set CityLoyal( cityLoyal)

get CityLoyal() //城池民心的实现

set CityBelongFactionID( cityBelongFactionID)

get CityBelongFactionID() //城池所属势力ID的实现

势力类属性的定义及实现

class Faction{

constructor(FactionID,FactionName,FactionLeaderPersonID,FactionCapitalID){

this.FactionID=FactionID;

this.FactionName=FactionName; this.FactionLeaderPersonID=FactionLeaderPersonID;

this.FactionCapitalID=FactionCapitalID;

}

}//势力类属性的定义

set FactionID( FactionID)

get FactionID() //势力ID的实现

set FactionName(FactionName)

get FactionName() //势力名字的实现

setFactionLeaderPersonID( FactionLeaderPersonID)

get FactionLeaderPersonID() 势力领导者ID的实现

set FactionCapitalID( FactionCapitalID)

get FactionCapitalID()//所属势力首都ID的实现

武将类属性的定义及实现

class Persons{

constructor(personID,personName,personBirthYear,personCommand,personMilitary,personPolitics,personTrick,personLoyal,personCityID,personState){

this.personID=personID;

this.personName=personName;

this.personBirthYear=personBirthYear;

this.personCommand=personCommand;

this.personMilitary=personMilitary;

this.personPolitics=personPolitics;

this.personTrick=personTrick;

this.personLoyal=personLoyal;

this.personCityID=personCityID;

this.personState=personState;

}

}

set PersonID( personID)

get PersonID() //武将ID的实现

set PersonName( personName)

this.personName = personName;

get PersonName() //武将名字的实现

set PersonBirthYear( personBirthYear)

get PersonBirthYear() //武将出生年份的实现

set PersonCommand( personCommand)

get PersonCommand() //武将命令的实现

set PersonMilitary( personMilitary)

get PersonMilitary() //武将命令的实现

set PersonPolitics( personPolitics)

get PersonPolitics() //武将政治的实现

set PersonTrick( personTrick)

get PersonTrick() //武将智力的实现

set PersonLoyal( personLoyal)

get PersonLoyal() //武将忠诚的实现

set PersonCityID( personCityID)

get PersonCityID() //武将城池ID的实现

set PersonState( personState)

get PersonState() //武将状态的实现

对战类属性的定义及实现

class Battle{

export default class Battle

constructor(BattleID,BattleName,BattleAttackArmyID,BattleAttackCityName,BattleDefendArmyID,BattleFieldCityID,BattleResult){

this.BattleID=BattleID;

this.BattleName=BattleName;

this.BattleAttackArmyID=BattleAttackArmyID;

this.BattleDefendArmyID=BattleDefendArmyID;

this.BattleFieldCityID=BattleFieldCityID;

this.BattleResult=BattleResult; this.BattleAttackCityName=BattleAttackCityName;

}

}//对战类实现

set BattleID( BattleID)

get BattleID()//对战ID的实现

set BattleName( BattleName)

get BattleName()//对战名字的实现

set BattleAttackArmyID( BattleAttackArmyID)

get BattleAttackArmyID()//对战进攻方ID的实现

set BattleDefendArmyID( BattleDefendArmyID)

get BattleDefendArmyID()//对战防守方ID的实现

set BattleFieldCityID( BattleFieldCityID)

get BattleFieldCityID()//对战胜利方ID的实现

set BattleResult( BattleResult)

get BattleResult()//对战结果的实现

set BattleAttackCityName(BattleAttackCityName)

get BattleAttackCityName(BattleAttackCityName)

//对战进攻方城市ID的实现

时间类属性的定义及实现

class Time {

constructor(year,season){

this.year=year;

this.season=season;

}

}

set Year( year)

get Year() //年份的实现

set Season( season)

get Season()//季节的实现

玩家类属性的定义及实现

class Citys {

constructor(playerID,playerScore){

this.playerID=playerID;

this.playerScore=playerScore;

}

}

set PlayerID( playerID)

get PlayerID()//玩家ID的实现

set PlayerScore( playerScore)

get PlayerScore()//玩家成绩的实现

2类函数实现

军队类函数的实现

armyChangeMainID( number)//主将修改

armyChangeAssisiant1ID( number)//副将1修改

armyChangeAssisiant2ID( number)//副将2修改

armyChangeSoldier( number)//士兵修改

armyChangeFood( number)//粮草修改

armyChangeMoney( number)//金钱修改

城池类函数的实现

cityChangeFood( number)//粮草改变

cityChangeFarm( number)//农业改变

cityChangeMoney(number)//金钱改变

cityChangeBusiness( number)//商业改变

cityChangeDefense( number)//城防改变

cityChangeSoldier( number)//士兵数改变

cityChangeLoyal( number)//民忠改变

cityChangeBelongFactionID( number)//所属势力改变

势力类函数的实现

FactionChangeLeaderPersonID( number)势力主公更换

FactionChangeCapitalID( number)//势力首都城池更换

武将类函数的实现

personChangeLoyal( number)//武将忠诚改变

PersonChangeCityID( cityID)//所在城池改变

personChangeState(state)//状态改变

对战类函数的实现

BattleNameGeneration(BattleAttackCityName)//战役名称生成

BattleResultGeneration(BattleAttackArmyID,BattleDefendArmyID, BattleFieldCityID)//战役结果生成

时间类函数的实现

timeSeasonChange( season)//季节变化

timeYearChange( year)//新年快乐

玩家类函数的实现

PlayerChangeMaxScore( MaxScore)//玩家最高成绩改变

# 验收说明

完成所有基础目标。老师、评审通过最终审核。