

**G16组号：G16**

**代 码 清 单**



课 程 名 称 软件工程基础

项 目 名 称 模拟三国

小 组 名 称 明德1-518

指 导 老 师 杨枨



|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 代码版本号 | 更新内容 | 更新日期 |
| 1 | V0.0.0 | 最初的代码框架 | 2019.5.14 |
| 2 | V0.0.1 | 框架细化 | 2019.5.24 |
| 3 | V0.0.2 | 初版代码清单编写完成 | 2019.5.26 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

注：红色标注为当前文档版本。

目录

[1 引言 2](#_Toc9774440)

[1.1 编写目的 2](#_Toc9774441)

[1.2 阅读对象 2](#_Toc9774442)

[1.3 注意事项 2](#_Toc9774443)

[1.4 参考模板 2](#_Toc9774444)

[2 系统简述 3](#_Toc9774445)

[2.1 系统流程 3](#_Toc9774446)

[2.2 功能模块 4](#_Toc9774447)

[2.3 代码分工 5](#_Toc9774448)

[2.4 功能关系 6](#_Toc9774449)

[3 详细代码 7](#_Toc9774450)

[3.1 前端部分： 7](#_Toc9774451)

[3.2 后端部分： 17](#_Toc9774452)

# 引言

## 编写目的

1. 开发时，帮助内部人员（指设计人员和实现人员）理清代码层次和关系，确认自身的工作完成情况，使工作按照设计完成。
2. 测试时，帮助测试人员确认实际代码是否按照设计来编写。
3. 评审时，为评审人员提供评审对象。

## 阅读对象

设计人员，实现人员，测试人员和评审人员。

## 注意事项

注意1：本文档仅供内部人员查阅！！如给他人借阅，请通报全组成员！

注意2：每次更新时需要在文档第2页更新文档版本，记得改变红色字体颜色。

注意3：当出现代码清单和实际设计矛盾时，应先组内通报并讨论，再决定修改实际代码还是代码清单。

## 参考模板

<https://wenku.baidu.com/view/853c9869453610661ed9f463.html>

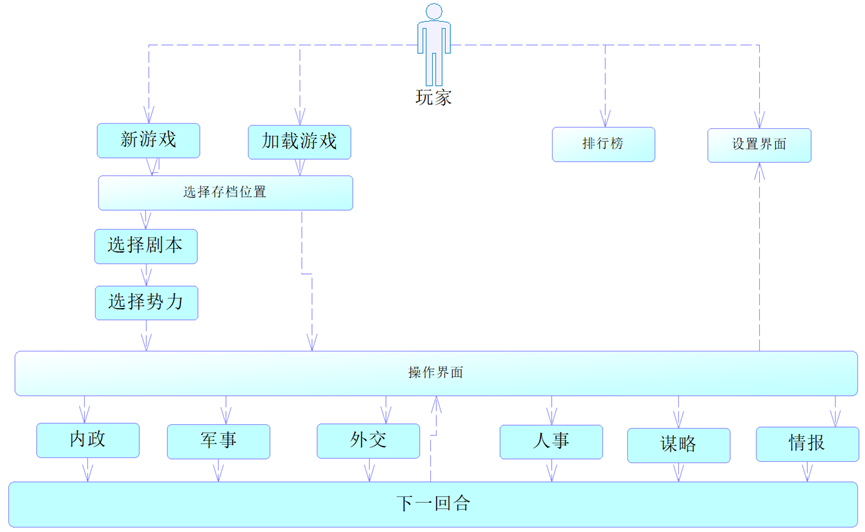
百度文库.源代码清单模板.2018-07-01

<https://wenku.baidu.com/view/fd41a46cf5335a8102d2207d.html>

百度文库.源代码清单模板.2018-06-30

# 系统简述

## 系统流程



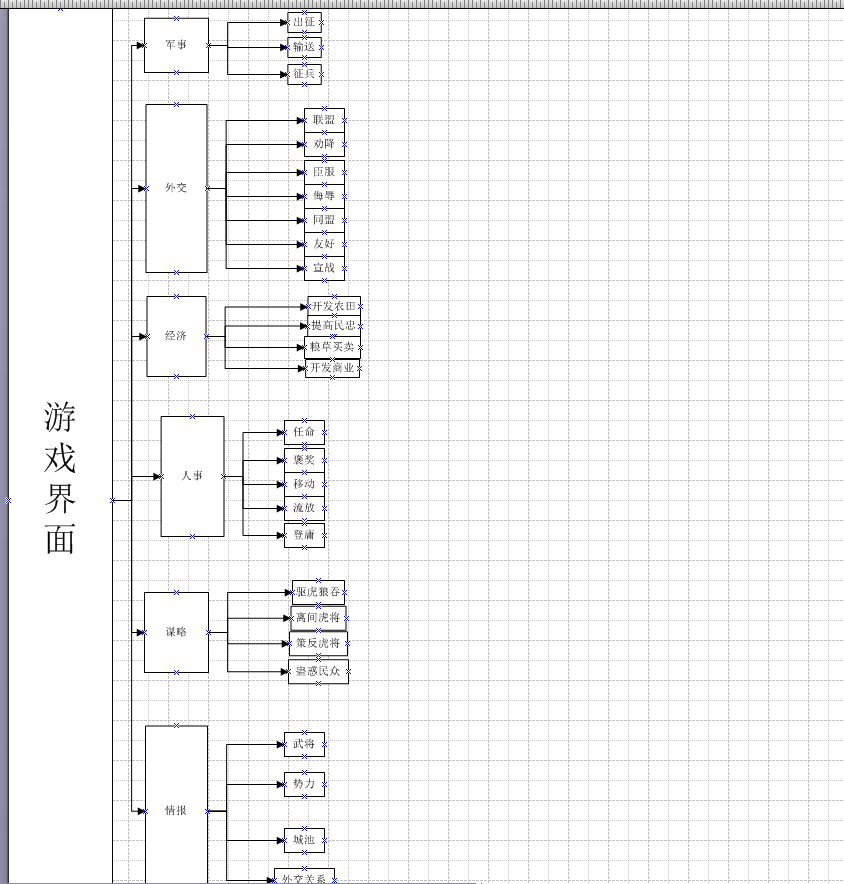
## 功能模块

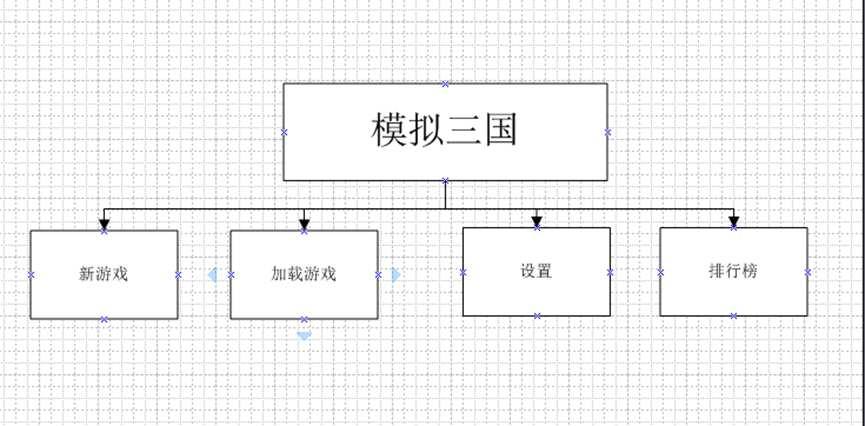
|  |  |  |  |
| --- | --- | --- | --- |
| **一级模块** | **二级模块** | **三级模块** | **模块描述** |
| 新游戏 | 存档选择 | 空白新增 | 在未有存档的地方新增存档 |
| 覆盖原有 | 在已有存档的地方覆盖存档 |
| 剧本选择 | | 选择游玩的剧本 |
| 势力选择 | | 选择游玩的势力 |
| 加载游戏 | 存档选择 | 存档读取 | 读取选择的存档 |
| 存档删除 | 删除选择的存档 |
| 游戏界面 | 军事 | | 出征，输送，征兵 |
| 经济 | | 开发农田，开发商业，粮草买卖，提高民忠 |
| 人事 | | 任命，褒奖，移动，流放，登庸 |
| 谋略 | | 驱虎吞狼，离间武将，策反武将，蛊惑民众 |
| 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，字体等进行设置 |

## 代码分工

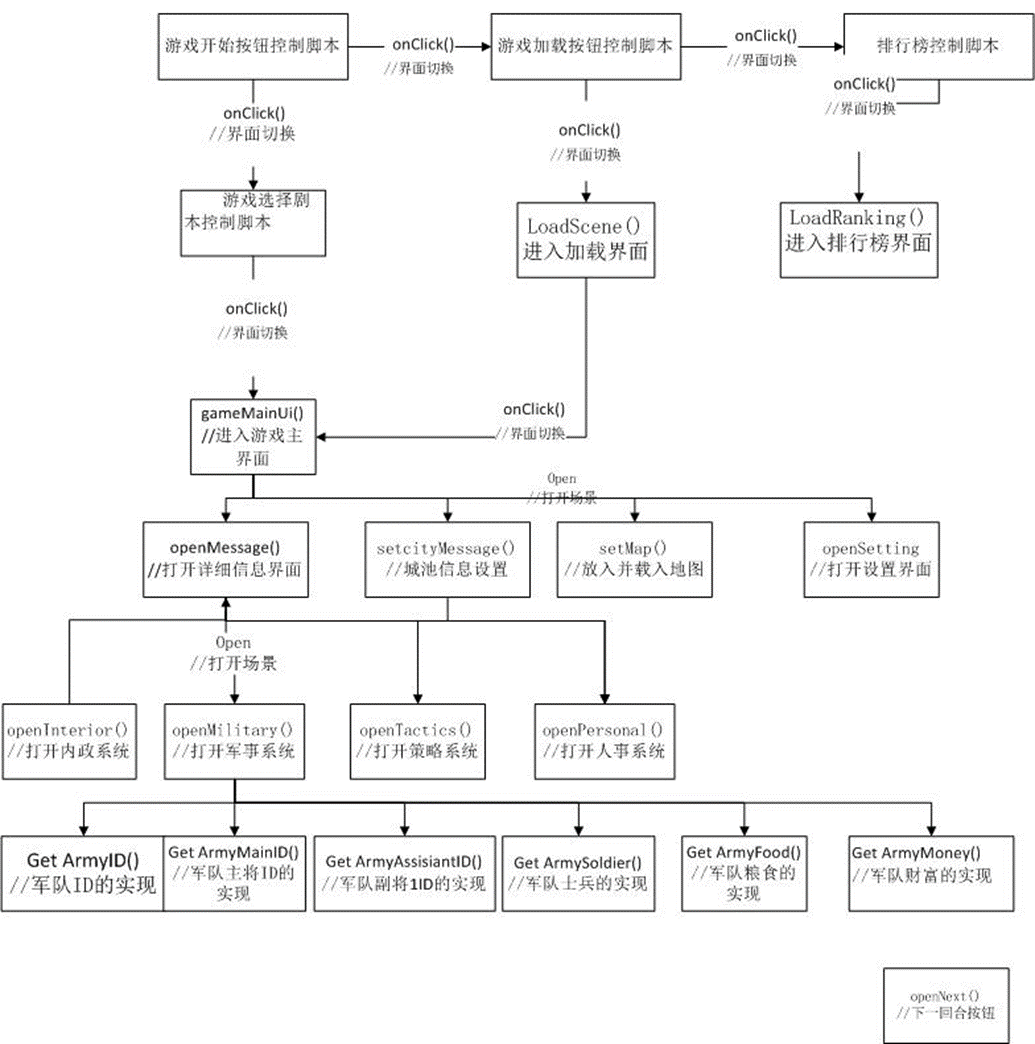
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 一级分类 | 二级分类 | 三级分类 | 最终分类 | 分工 |
| 前端 | UI用户界面设计 | 界面 | 界面设计 | 王华怿 |
| 界面实现 | 王仕杰 |
| 逻辑 | 逻辑设计 |
| 逻辑实现 |
| 美工 | 美工设计 |
| 美工实现 |
| 美工素材 | 可自制素材 | 素材制作 |
| 不可自制素材 | 素材寻找 | 王华怿 |
| 素材加工 | 吴帅毅 |
| 后端 | 数据库交互 | 本地数据库 | 本地数据库设计 | 王华怿 |
| 本地数据库实现 | 吴帅毅 |
| 云服务器数据库 | 云服务器数据库连接 | 王华怿 |
| 云服务器数据库设计 |
| 云服务器数据库实现 | 吴帅毅 |
| 游戏运行逻辑 | 前后端对接 | 前后端对接设计 | 王华怿 |
| 前后端对接实现 | 吴帅毅 |
| 核心运行 | 游戏运行逻辑设计 | 王华怿 |
| 游戏运行函数实现 | 吴帅毅 |
| 类设计及实现 | 常量 | 常量设计 | 王华怿 |
| 类常量实现 | 吴帅毅 |
| 函数 | 函数设计 | 王华怿 |
| 类函数实现 | 吴帅毅 |

## 功能关系





# 详细代码



## 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