天天爱消除

软件体系结构描述文档

目录

[天天爱消除 1](#_Toc385496925)

[软件体系结构描述文档 1](#_Toc385496926)

[1．引言 2](#_Toc385496927)

[1.1编制目的 2](#_Toc385496928)

[1.2词汇表 2](#_Toc385496929)

[1.3参考资料 3](#_Toc385496930)

[2.产品概述 3](#_Toc385496931)

[3.逻辑视角 3](#_Toc385496932)

[4.组合视角 5](#_Toc385496933)

[4.1开发包图 5](#_Toc385496934)

[4.2运行时进程 9](#_Toc385496935)

[4.3物理部署 9](#_Toc385496936)

[5.接口视角 10](#_Toc385496937)

[5.1模块的职责 10](#_Toc385496938)

[5.2 View层的分解 13](#_Toc385496939)

[5.2.1 View层模块的职责 13](#_Toc385496940)

[5.2.2 View层模块的接口规范 14](#_Toc385496941)

[5.2.3 用户界面模块设计原理 19](#_Toc385496942)

[5.3 Control层的分解 20](#_Toc385496943)

[5.3.1 Control层模块的职责 20](#_Toc385496944)

[5.3.2 Control层模块的接口规范 20](#_Toc385496945)

[5.4 Model层的分解 27](#_Toc385496946)

[5.4.1 Model层的职责 27](#_Toc385496947)

[5.4.2 Model层模块的接口规范 27](#_Toc385496948)

[6 信息视角 34](#_Toc385496949)

[6.1 数据持久化对象 34](#_Toc385496950)

[6.2 持久化保存 34](#_Toc385496951)

[7 变更历史 34](#_Toc385496952)

# 1．引言

## 1.1编制目的

本报告详细完成对天天爱消除软件程序的概要设计，达到指导设计和开发的目的，同时实现和测试人员及用户的沟通，为以后的详细设计和软件构建建立基础。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| **词汇名称** | **词汇含义** | **备注** |
| View | 游戏界面模块 | MVC中的V模块 |
| Control | 游戏控制器模块 | MVC中的C模块 |
| Model | 游戏模型模块 | MVC中的M模块 |

## 1.3参考资料

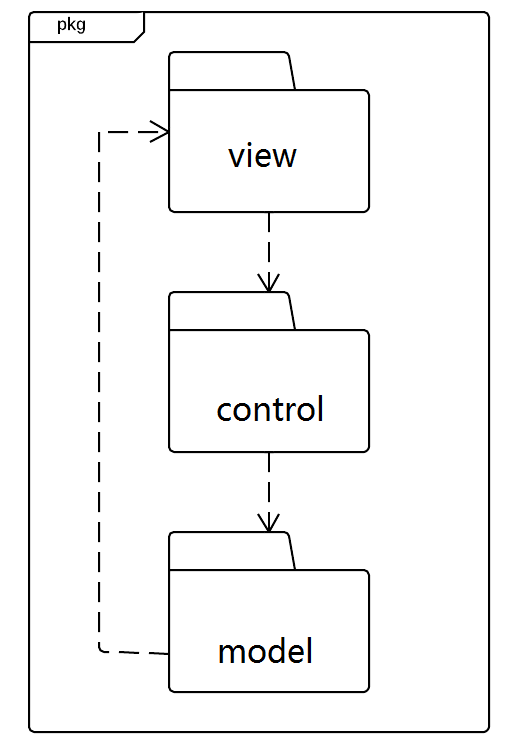
《软件开发的技术基础》

# 2.产品概述

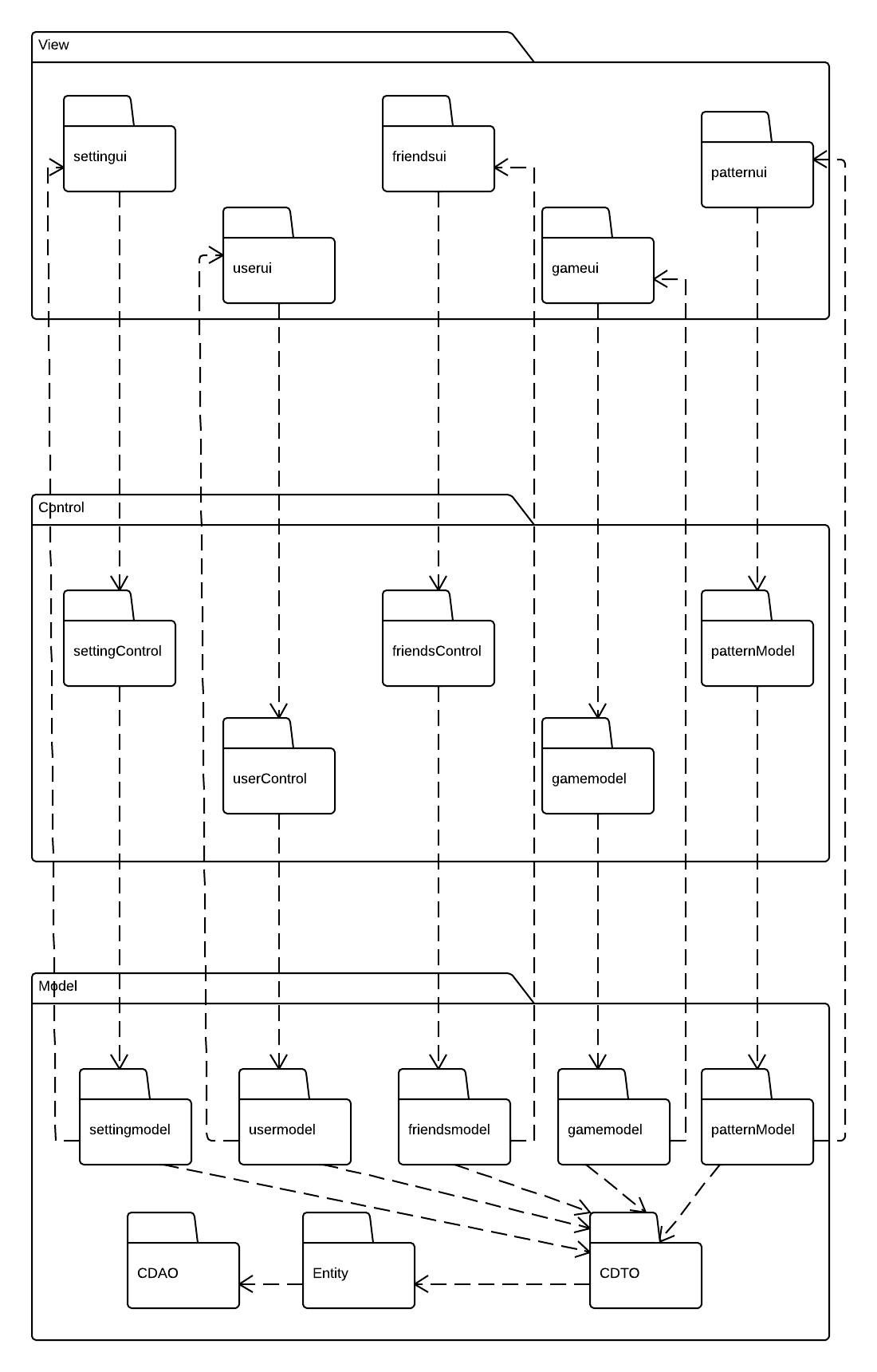
参考天天爱消除用例文档和天天爱消除软件系统规格说明中对于产品的概述。

# 3.逻辑视角

天天爱消除软件系中，选择了MVC体系结构风格，将系统分为3个组成部分（View, Control, Model），能够很好的示意整个高层抽象，展示包含View页面的实现，model负责业务逻辑和数据处理保存，而control则用于对于view中命令的解析。MVC结构的逻辑视角和逻辑设计方案分别如图1和图2所示。



#### 图1 逻辑视角



#### 图2 逻辑设计方案

# 4.组合视角

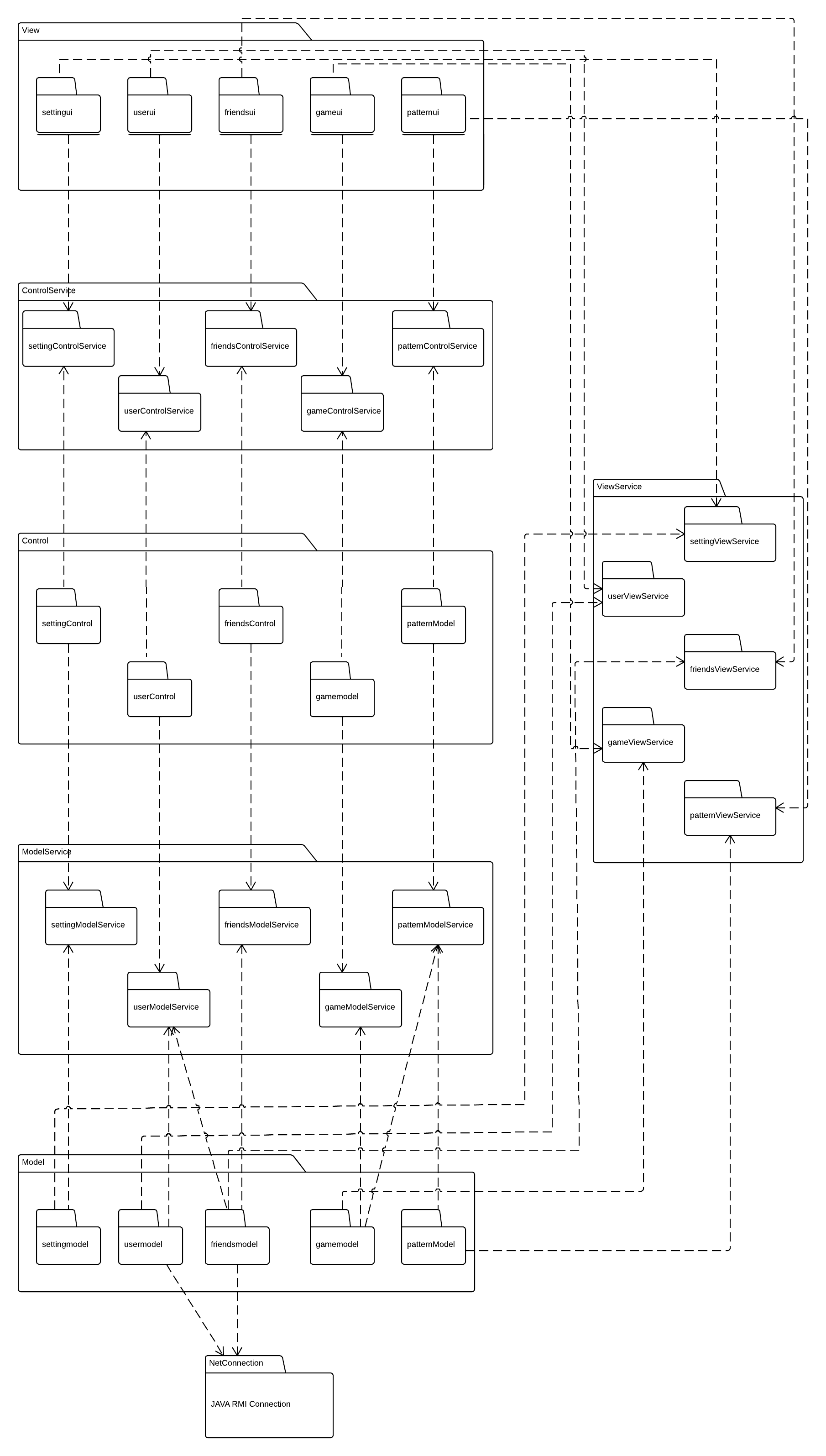
## 4.1开发包图

天天爱消除的最终开发包设计如表1所示

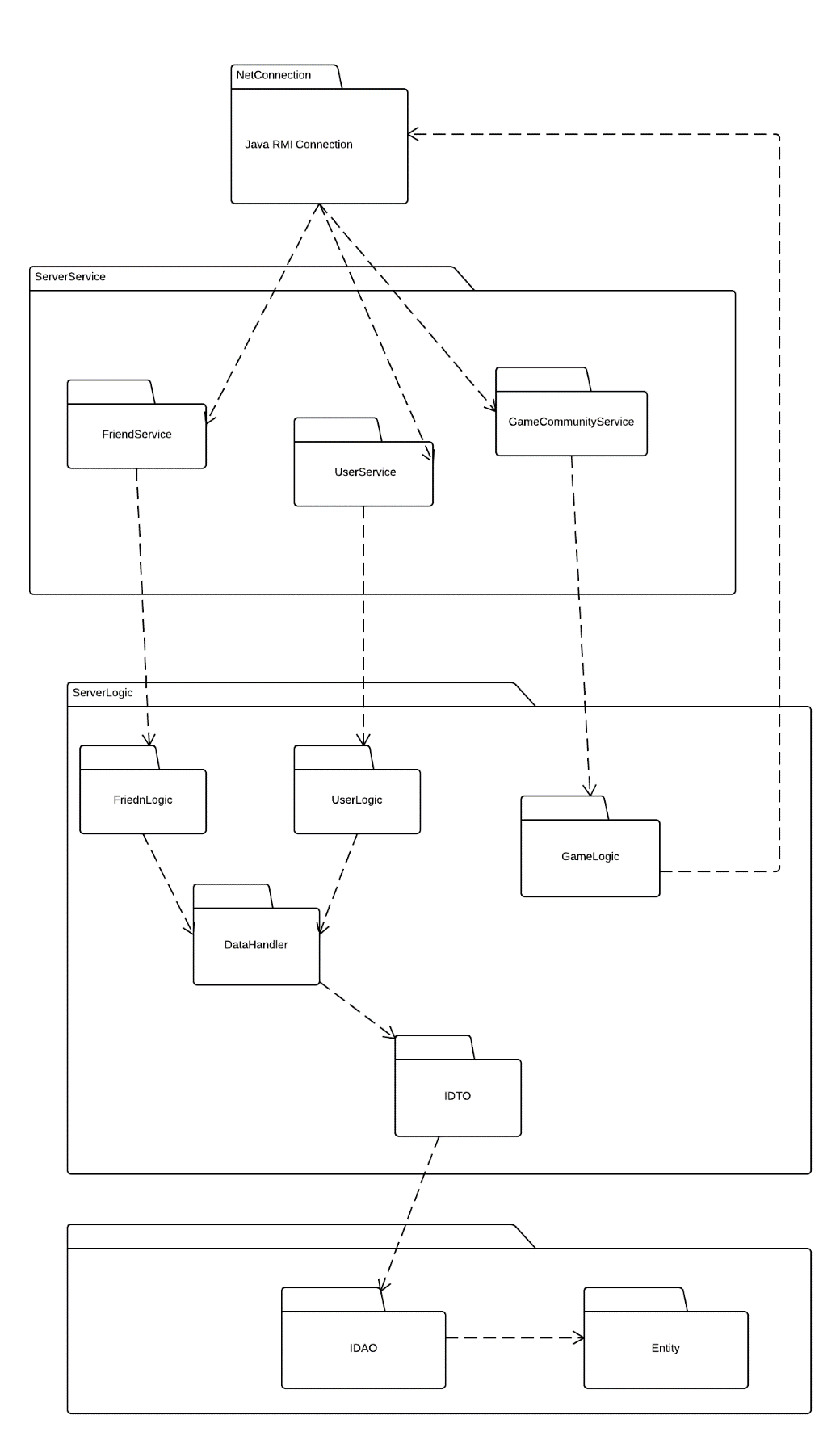
#### 表1 开发包设计表

|  |  |
| --- | --- |
| **开发（物理）包** | **依赖的其他开发包** |
| settingUI | settingControlService,settingViewService,界面类库包,vo |
| settingControlService | vo |
| settingControl | settingModelService,settingControlService,vo |
| settingModelService | vo |
| settingModel | settingModelService,settingViewService,vo |
| settingViewService | vo |
| userUI | userControlService,userViewService,界面类库包,vo |
| userControlService | vo |
| userControl | userModelService,userControlService,vo |
| userModelService | vo |
| userModel | userModelService,userViewService,vo,po,Java RMI,userService |
| userViewService | vo |
| userService | vo,po,Java RMI,userLogic |
| userLogic | DataHandler,vo |
| friendUI | friendControlService,friendViewService,界面类库包,vo |
| friendControlService | vo |
| friendControl | friendModelService,friendControlService,vo |
| friendModelService | vo |
| friendModel | friendModelService,userModelService,friendViewService,vo,po,JavaRMI,friendService |
| friendViewService | vo |
| friendService | vo ,po,Java RMI,friendLogic |
| frienLogic | DataHandler,vo |
| gameUI | gameControlService,gameViewService,界面类库包,vo |
| gameControlService | vo |
| gameControl | gameModelService,gameControlService,vo |
| gameModelService | vo |
| gameModel | gameModelService,gameViewService,patternModelService,friendModelService,vo |
| gameViewService | vo |
| patternUI | patternControlService,patternViewService,界面类库包,vo |
| patternControlService | vo |
| patternControl | patternModelService,patternControlService,vo |
| patternModelService | vo |
| patternModel | patternModelService,patternViewService,vo |
| patternViewService | vo |
| DateHandler | vo,IDTO |
| GameCommunityService | GameLogic,Java RMI |
| GameLogic | Java RMI |
| IDTO | IDAO |
| IDAO | Entity |
| Entity |  |
| vo |  |
| po |  |
| Java RMI |  |
| 界面类库包 |  |

天天爱消除客户端如图3所示，服务器端开发包图如图4所示



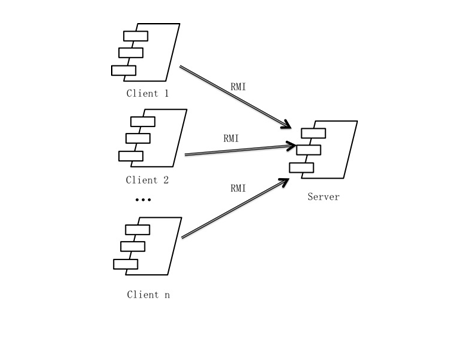
#### 图3 客户端开发包图



#### 图4 服务器端开发包图

## 4.2运行时进程

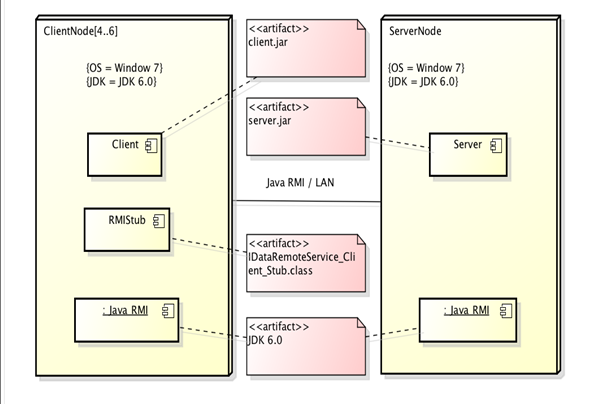
在天天爱消除软件系统中，会有很多个客户端进程和一个服务端进程，其进程图如图5所示，结合部署图，客户端进程是在客户端机器上运行，服务器端在服务器端机器上运行。



#### 图5 进程图

## 4.3物理部署

天天爱消除软件系统中客户端构件是位于客户端机器上，服务器端构件是放在服务器端。在客户端节点上，还要部署基于RMI的网络连接通信构件。部署图如图6所示。

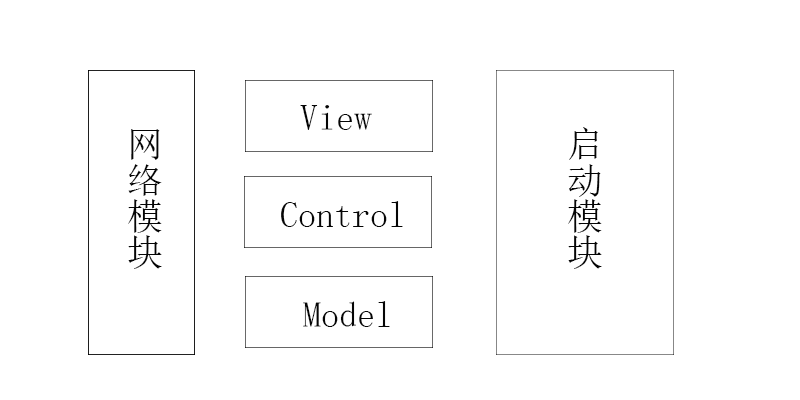


#### 图6 部署图

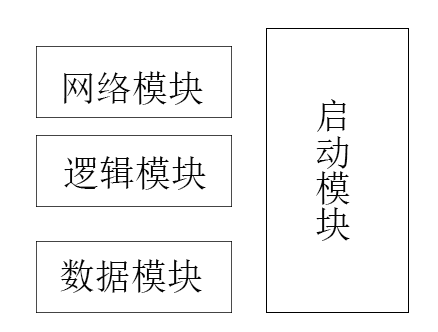
# 5.接口视角

## 5.1模块的职责

客户端模块和服务器端视图分别如图7和图8所示，客户端各模块和服务器端各模块的职责分别如表2和表3所示。



#### 图7 客户端模块视图



#### 图8 服务器端模块视图

#### 表2 客户端各模块职责表

|  |  |
| --- | --- |
| **模块** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 网络模块 | 负责网络通信的维护和信息的传递 |
| View | 基于窗口的天天爱消除客户端用户界面 |
| Control | 负责接收View层的命令，进行方法的解析和任务的分配 |
| Model | 负责游戏逻辑处理和数据的处理保存 |

#### 表3 服务器端各模块职责表

|  |  |
| --- | --- |
| **模块** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动服务器端管理员界面 |
| 网络模块 | 负责网络通信的维护和信息的传递 |
| 逻辑模块 | 负责业务逻辑的操作 |
| 数据模块 | 负责用户数据的存储与读取 |

View模块会调用Control模块的接口；control模块调用View模块和Model模块的接口；model模块会调用View模块的接口。模块与模块之间的接口如表4所示

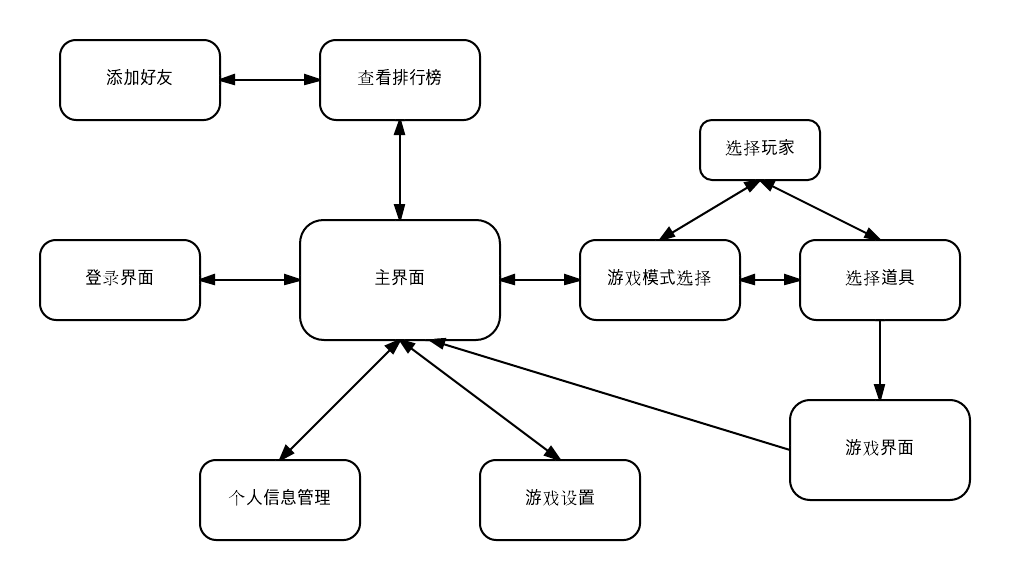
#### 表4 层之间调用的接口

|  |  |  |
| --- | --- | --- |
|  | **服务调用方** | **服务提供方** |
| userViewService  settingViewService  friendViewService  gameViewService  patterViewService | 客户端模型层 | 客户端界面层 |
| userControlService  settingControlService  friendControlService  gameControlService  patterControlService | 客户端界面层 | 客户端控制层 |
| userModelService  settingModelService  friendModelService  gameModelService  patterModelService | 客户端控制层 | 客户端模型层 |

#### 图9 移动图案用例层之间调用的接口

## 5.2 View层的分解

根据需求，天天爱消除存在10个用户界面，登陆界面，程序主界面，设置界面，游戏选择界面，选择玩家界面，游戏初始化界面，单机游戏进行界面，对战游戏界面，对战游戏界面。图片跳转如图10所示。



#### 图10 界面跳转图

#### 图11 view类图

### 5.2.1 View层模块的职责

如表5所示为View模块的职责

#### 表5 View模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainUI | 界面frame，负责排行榜的显示和界面跳转 |
| FriendUI | 负责与好友管理有关的界面的显示和跳转 |
| GameUI | 负责与游戏初始化有关的界面的显示和跳转 |
| PatternUI | 负责与游戏过程进行有关的界面的显示和跳转 |
| SettingUI | 负责与游戏设置有关的界面的显示和跳转 |
| UserUI | 负责与用户信息管理有关的界面的显示和跳转 |

### 5.2.2 View层模块的接口规范

View层模块的接口规范如下各表所示

#### 表6 View层friend模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FriendUI.refresh | 语法 | Public void refreshFriend（ArrayList<FriendVO> friendList） |
| 前置条件 | 用户选择了与好友管理有关的操作 |
| 后置条件 | 系统保存该操作效果，并更新好友列表 |
| MainUI.refresh | 语法 | Public void refreshFriend（ArrayList<FriendVO> friendList） |
| 前置条件 | 用户回到系统主界面，即排行榜界面 |
| 后置条件 | 系统显示当前该玩家排行榜 |

|  |  |  |
| --- | --- | --- |
| **需要的服务（需接口）** | | |
| FriendController.addFriend | 语法 | Public void addFriend(UserVO uv) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| FriendController.findUser | 语法 | Public UserVO findUser(String id) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| FriendController.dealWithMessage | 语法 | Public void dealWithMessage(String answer) |
| 前置条件 | 用户成功登录并收到好友消息 |
| 后置条件 | 无 |
| FriendController.modifyFriend | 语法 | Public void modifyFriend（FriendVO fv） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |
| FriendController.friendInfo | 语法 | Public RecordVO friendInfo（string friendId） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |

#### 表7 View层pattern模块的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PatternUI.  refreshPattern | 语法 | | Public void refreshPattern(ArrayList<patternVO> patternsList) | |
| 前置条件 | | 用户的单击对界面产生影响 | |
| 后置条件 | | 无 | |
| PatternUI.  RefreshScore | 语法 | | Public void refreshScore(int score) | |
| 前置条件 | | 用户的游戏积分发生改变 | |
| 后置条件 | | 无 | |
| PatternModel.  propsA | 语法 | | Private void propsA(point p) | |
| 前置条件 | | 用户单击的坐标是一个道具A | |
| 后置条件 | | 消除周围8个图案 | |
| patternUI.showTips | 语法 | | Private void patternUI.showTips (point p) | |
| 前置条件 | | 界面生成消除提示 | |
| 后置条件 | | 无 | |
| 需要的服务（需接口） | | | | |
| PatternController.  watchMouse | | 语法 | | Public void watchMouse(MouseEvent mouseEvent) |
| 前置条件 | | 游戏已初始化，用户在游戏界面有鼠标操作 |
| 后置条件 | | 根据不同的鼠标事件调用相应的逻辑操作 |

#### 表8 View层setting模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务接口（供接口）** | | |
| SettingUI.refreshVolume | 语法 | Public void refreshVolume(int volume) |
| 前置条件 | 用户修改了游戏音量 |
| 后置条件 | 系统更新界面的音量数字显示 |
| SettingUI.showWindow | 语法 | Public void showWindow() |
| 前置条件 | 用户选择关闭/打开游戏音效 |
| 后置条件 | 系统弹出提示窗口 |
| SettingUI.showInfo | 语法 | Public void showInfo() |
| 前置条件 | 用户已经查看游戏相关信息 |
| 后置条件 | 系统显示出游戏相关信息的界面 |
| **需要的服务接口（需接口）** | | |
| SettingController.increaseVolume(int volume) | 处理来自界面增大音量的输入 | |
| SettingController.decreaseVolume(int volume) | 处理来自界面减小音量的输入 | |
| SettingController.changeStyle(int style) | 处理来自界面选择的具体的游戏风格 | |
| SettingController.changeWay(int way) | 处理来自界面选择的具体的图案下落方式 | |

#### 表9 View层game模块的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务接口（供接口）** | | | | |
| GameUI.initialTable | 语法 | | Public void initialTable() | |
| 前置条件 | | 用户单击开始本局游戏 | |
| 后置条件 | | 显示初始化的棋盘，初始化棋盘中不允许已有超过三个以上可消除的图案 | |
| GameUI.selectGameMode | 语法 | | Public void selectGameMode(int gameMode) | |
| 前置条件 | | 用户单击开始游戏 | |
| 后置条件 | | 用户选择本局游戏使用的道具 | |
| GameUI. selectItem | 语法 | | Private void selectItem(int itemNum) | |
| 前置条件 | | 用户已选择游戏模式 | |
| 后置条件 | | 系统初始化棋盘，跳转到游戏界面 | |
| **需要的服务（需接口）** | | | | |
| GameController.creatGame | | 语法 | | Public void creatGame(Game g) |
| 前置条件 | | 用户选择完游戏模式、道具并单击开始游戏 |
| 后置条件 | | 显示游戏界面 |

#### 表10 View层user模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| UserUI.loginRefresh | 语法 | Private void loginRefresh() |
| 前置条件 | UserModel模块返回注册成功 |
| 后置条件 | 由注册界面跳转到登录界面 |
| UserUI.loginFail | 语法 | Private void loginFail() |
| 前置条件 | UserModel模块返回登录失败 |
|  | 界面提示登录失败，请重新登录 |
| RegisterUI.refresh | 语法 | Private void refresh() |
| 前置条件 | 用户选择注册功能 |
| 后置条件 | 跳转到注册界面 |
| RegisterUI.fail | 语法 | Private void fail() |
| 前置条件 | UserModel模块返回用户已存在 |
|  | 界面提示用户已存在，请重新填写注册信息 |
| **需要的服务（需接口）** | | |

|  |  |  |
| --- | --- | --- |
| UserController.login | 语法 | public userPO login(String userName,String password) |
| 前置条件 | password符合输入规则 |
| 后置条件 | 查找是否存在相应的user，根据输入的password返回登录验证的结果 |
| UserController.register | 语法 | public boolean register(String userName,String password,String gender) |
| 前置条件 | 各项信息符合输入规则 |
| 后置条件 | 查找是否有重复，返回添加结果 |
| UserController.  changePassword | 语法 | public String changePassword(String userName,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果 |
| UserController.isIDExist | 语法 | public boolean isIDExist(String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回查找结果 |
| UserController.  checkGameRecord | 语法 | Public GameRecord checkGameRecord(String id) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 返回查找结果 |
| MainUI.refresh | 语法 | public void refresh() |
|  | 前置条件 | 用户登录成功 |
|  | 后置条件 | 界面跳转到主界面 |

### 5.2.3 用户界面模块设计原理

用户界面利用Java的Swing和AWT库来实现。

## 5.3 Control层的分解

控制层包括多个针对界面的控制处理对象。例如，User对象负责处理登录界面的控制逻辑；friend对象负责好友管理的控制逻辑。控制层的设计如图12所示。

#### 图12 控制层的设计

### 5.3.1 Control层模块的职责

控制层模块的职责如表8所示

#### 表8 控制层模块的职责

|  |  |
| --- | --- |
| **模块** | **职 责** |
| userControl | 负责实现对应登录和个人信息管理界面所需要的服务 |
| friendControl | 负责实现对应好友管理界面所需要的服务 |
| patternControl | 负责实现对应游戏界面移动图案所需要的服务 |
| gameControl | 负责实现开始游戏界面所需要的服务 |
| settingControl | 负责实现系统设置界面所需要的服务 |

### 5.3.2 Control层模块的接口规范

Control层模块的接口规范如下各表所示。

#### 表9 friend模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FriendController.addFriend | 语法 | Public void addFriend(UserVO uv) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| FriendController.findUser | 语法 | Public UserVO findUser(String id) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| FriendController.dealWithMessage | 语法 | Public void dealWithMessage(String answer) |
| 前置条件 | 用户成功登录并收到好友消息 |
| 后置条件 | 无 |
| FriendController.modifyFriend | 语法 | Public void modifyFriend（FriendVO fv） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |
| **需要的服务（需接口）** | | |
| FriendModel.addFriend | 语法 | Public void addFriend(UserPO, up,FriendPO fp) |
| 前置条件 | 用户成功登录并请求添加好友 |
| 后置条件 | 无 |
| UserModel.findUser | 语法 | Public UserPO findUser（UserPO up） |
| 前置条件 | 用户成功登录并查找用户 |
| 后置条件 | 无 |
| GameModel.createGame | 语法 | Public void createGame（） |
| 前置条件 | 用户成功登录并接受游戏邀请 |
| 后置条件 | 无 |
| FriendModel.modifyFriend | 语法 | Public void modifyFriend（FriendPO fp） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |
| FriendModel.sendMessage | 语法 | Public void sendMessage(message) |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |

#### 表10 pattern模块的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务（供接口） | | | | |
| PatternController.  watchMouse | 语法 | | Public void watchMouse(MouseEvent mouseEvent) | |
| 前置条件 | | 游戏已初始化，用户在游戏界面有鼠标操作 | |
| 后置条件 | | 根据不同的鼠标事件调用相应的逻辑操作 | |
| 需要的服务（需接口） | | | | |
| PatternModel.  mouseClick | | 语法 | | Public void mouseClick(point cilckPoint) |
| 前置条件 | | 用户在游戏界面的鼠标操作为单击 |
| 后置条件 | | 根据单击的坐标判断后续逻辑操作 |
| PatternModel.  mouseMove | | 语法 | | Public void moveMouse(point p1,point p2) |
| 前置条件 | | 用户在游戏界面的鼠标操作为拖动鼠标 |
| 后置条件 | | 根据鼠标拖动的前后坐标判断后续逻辑操作 |
| PatternModel.showTips | | 语法 | | Public void showTips() |
| 前置条件 | | 用户2s内无任何操作 |
| 后置条件 | | 调用PatternView.showTips() |

#### 表11 setting模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务接口（供接口）** | | |
| SettingController.increaseVolume | 语法 | Public void increaseVolume(MouseEvent mouseEvent,int volume) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据不同的鼠标操作调用相应的逻辑操作 |
| SettingController.decreaseVolume | 语法 | Public void decreaseVolume(MouseEvent mouseEvent,int volume) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据不同的鼠标操作调用相应的逻辑操作 |
| SettingController.changeStyle | 语法 | Public void changeStyle(MouseEvent mouseEvent,int style) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据不同的鼠标操作调用相应的逻辑操作 |
| SettingController.changeWay | 语法 | Public void changeStyle(MouseEvent mouseEvent,int style) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据不同的鼠标操作调用相应的逻辑操作 |
| SettingController.getInfo | 语法 | Public void getInfo(MouseEvent mouseEvent) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 游戏信息内容不因用户的操作发生改变 |
| SettingController.quit | 语法 | Public void quit（MouseEvent mouseEvent） |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据鼠标操作完成相应的逻辑操作 |
| **需要的服务接口（需接口）** | | |
| SettingModel.increaseVolume  (int volume) | 增大游戏系统音量 | |
| SettingModel.decreaseVolume(int volume) | 减小游戏系统音量 | |
| SettingModel.closeVoice(boolean isOpen) | 打开或关闭游戏音效 | |
| SettingModel.changeStyle(int style) | 更改游戏图案风格 | |
| SettingModel.changeWay(int way) | 更改游戏图案下落方向 | |
| SettingModel.getInfo( ) | 获得游戏相关信息 | |
| SettingModel.quit( ) | 退出游戏设置 | |

#### 表12 game模块的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务（供接口）** | | | | |
| GameController.creatGame | 语法 | | Public void creatGame(Game g) | |
| 前置条件 | | 用户选择完游戏模式、道具并单击开始游戏 | |
| 后置条件 | | 显示游戏界面 | |
| **需要的服务（需接口）** | | | | |
| GameModel.creatTableModel | | 语法 | | Public void creatTableModel() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 创建棋盘初始矩阵 |
| GameModel.loadItem | | 语法 | | Public void loadItem() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 加载本局可使用的道具 |
| GameModel.sentGameInvitation | | 语法 | | Public void sentGameInvitation() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 发送游戏邀请 |
| GameModel.sentGame | | 语法 | | Public void sentGame(String IP) |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 发送游戏参数 |

#### 表13 user模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| UserController.login | 语法 | public userPO login(String userName,String password) |
| 前置条件 | password符合输入规则 |
| 后置条件 | 查找是否存在相应的user，根据输入的password返回登录验证的结果 |
| UserController.register | 语法 | public boolean register(String userName,String password,String gender) |
| 前置条件 | 各项信息符合输入规则 |
| 后置条件 | 查找是否有重复，返回添加结果 |
| UserController.  changePassword | 语法 | public boolean changePassword(String userName,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果 |
| UserController.isIDExist | 语法 | public boolean isIDExist(String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回查找结果 |
| UserController.  checkGameRecord | 语法 | Public GameRecord checkGameRecord(String id) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 返回查找结果 |
| **需要的服务（需接口）** | | |
| UserModel.login | 语法 | public String login(String userName,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果 |
| UserModel.  changePassword | 语法 | public String changePassword(String userName,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果 |
| UserModel.register | 语法 | public boolean register(String userName,String password,String gender) |
| 前置条件 | 各项信息符合输入规则 |
| 后置条件 | 查找是否有重复，返回添加结果 |
| UserModel.isIDExist | 语法 | public boolean isIDExist(String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回查找结果 |
| UserModel.getSelfInfor | 语法 | public UserPO getSelfInfor(String id) |
| 前置条件 | 用户已成功登录 |
| 后置条件 | 返回该用户的个人信息 |
| UserModel.  checkGameRecord | 语法 | public GameRecord checkGameRecord(String id) |
| 前置条件 | 用户已成功登录 |
| 后置条件 | 返回该用户的游戏记录 |

## 5.4 Model层的分解

Model层主要给Control层提供各种主要的逻辑处理和数据操作服务，并在数据有更新时及时同时View层。

### 5.4.1 Model层的职责

Model层模块的职责如表14所示。

#### 表14 Model层模块的职责

|  |  |
| --- | --- |
| **模块** | **职 责** |
| userModel | 负责实现登录和个人信息管理的数据操作逻辑 |
| friendModel | 负责实现对应好友管理所需的数据和逻辑操作服务 |
| patternModel | 负责实现对应游戏界面移动图案所需要的逻辑和数据操作服务 |
| gameModel | 负责实现初始化界面所需的逻辑和数据操作服务 |
| settingModel | 负责实现系统设置所需的逻辑和数据操作服务 |

### 5.4.2 Model层模块的接口规范

Model层模块的接口规范如下各表所示。

#### 表15 Model层friend模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FriendModel.addFriend | 语法 | Public void addFriend(FriendPO fp) |
| 前置条件 | 用户成功登录并请求添加好友 |
| 后置条件 | 无 |
| UserModel.findUser | 语法 | Public UserPO findUser（UserPO up） |
| 前置条件 | 用户成功登录并查找用户 |
| 后置条件 | 无 |
| UserModel.gameRecord | 语法 | Public recordPO gameRecord(string userId) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| GameModel.createGame | 语法 | Public void createGame（） |
| 前置条件 | 用户成功登录并接受游戏邀请 |
| 后置条件 | 无 |
| FriendModel.modifyFriend | 语法 | Public void modifyFriend（FriendPO fp） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |
| **需要的服务（需接口）** | | |
| FriendUI.refresh | 语法 | Public void refreshFriend（ArrayList<FriendVO> friendList） |
| 前置条件 | 用户选择了与好友管理有关的操作 |
| 后置条件 | 系统保存该操作效果，并更新好友列表 |
| MainUI.refresh | 语法 | Public void refreshFriend（ArrayList<FriendVO> friendList） |
| 前置条件 | 用户回到系统主界面，即排行榜界面 |
| 后置条件 | 系统显示当前该玩家排行榜 |

#### 表16 Model层pattern模块的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务（供接口）** | | | | |
| PatternModel.  mouseClick | | 语法 | | Public void mouseClick(point cilckPoint) |
| 前置条件 | | 用户在游戏界面的鼠标操作为单击 |
| 后置条件 | | 根据单击的坐标判断后续逻辑操作 |
| PatternModel.  moveMouse | | 语法 | | Public void moveMouse(point p1,point p2) |
| 前置条件 | | 用户在游戏界面的鼠标操作为拖动鼠标 |
| 后置条件 | | 根据鼠标拖动的前后坐标判断后续逻辑操作 |
| **需要的服务（需接口）** | | | | |
| PatternUI.  refreshPattern | 语法 | | Public void refreshPattern(ArrayList<patternVO> patternsList) | |
| 前置条件 | | 用户的单击对界面产生影响 | |
| 后置条件 | | 无 | |
| PatternUI.  RefreshScore | 语法 | | Public void refreshScore(int score) | |
| 前置条件 | | 用户的游戏积分发生改变 | |
| 后置条件 | | 无 | |
| patternUI.showTips | 语法 | | Private void patternUI.showTips (point p) | |
| 前置条件 | | 界面生成消除提示 | |
| 后置条件 | | 无 | |

#### 表17 Model层setting模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务接口（供接口）** | | |
| SettingModel.increaseVolume | 语法 | Public void increaseVolume(int volumn) |
| 前置条件 | 用户选择增大音量 |
| 后置条件 | 系统增大游戏音量，并且调用SettingUI.refreshSetting |
| SettingModel.decreaseVolume | 语法 | Public void decreaseVolume(int volumn) |
| 前置条件 | 用户选择减小音量 |
| 后置条件 | 系统减小游戏音量，并且调用SettingUI.refreshSetting |
| SettingModel.closeVoice | 语法 | Public void closeVoice(boolean isOpen) |
| 前置条件 | 用户选择关闭/打开音效 |
| 后置条件 | 系统关闭/打开游戏音效，并且调用SettingUI.remindWindow |
| SettingModel.changeStyle | 语法 | Public void changeStyle(int style) |
| 前置条件 | 用户选择某种风格 |
| 后置条件 | 系统根据选择调换风格，内部修改风格属性 |
| SettingModel.changeWay | 语法 | Public void changeWay(int Way) |
| 前置条件 | 用户选择改变游戏下落方向 |
| 后置条件 | 系统根据选择的下落方向，内部进行游戏属性的修改 |
| SettingModel.getInfo | 语法 | Public void getInfo( ) |
| 前置条件 | 用户选择查看游戏相关信息 |
| 后置条件 | 系统调用游戏相关信息，并且调用SettingUI.showInfo |
| SettingModel.quit | 语法 | Public void quit( ) |
| 前置条件 | 用户放弃修改游戏设置 |
| 后置条件 | 系统离开游戏设置界面，并且调用gameUI.refresh |
| **需要的服务接口（需接口）** | | |
| SettingUI.refreshVolume(int volume) | 更新游戏音量数字 | |
| SettingUI.showWindow( ) | 显示游戏音效更改的提示窗口 | |
| SettingUI.showInfo( ) | 显示游戏的相关信息窗口 | |
| mainUI.refresh | 刷新游戏主界面 | |

#### 表18 Model层game模块的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **提供的服务（供接口）** | | | | |
| GameModel.creatTableModel | | 语法 | | Public void creatTableModel() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 创建棋盘初始矩阵 |
| GameModel.loadItem | | 语法 | | Public void loadItem() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 加载本局可使用的道具 |
| GameModel.sentGameInvitation | | 语法 | | Public void sentGameInvitation() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 发送游戏邀请 |
| GameModel.sentGame | | 语法 | | Public void sentGame(String IP) |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 发送游戏参数 |
| **需要的服务（需接口）** | | | | |
| GameUI.initialTable | 语法 | | Public void initialTable() | |
| 前置条件 | | 用户单击开始本局游戏 | |
| 后置条件 | | 显示初始化的棋盘，初始化棋盘中不允许已有超过三个以上可消除的图案 | |
| GameUI.  selectGameMode | 语法 | | Public void selectGameMode(int gameMode) | |
| 前置条件 | | 用户单击开始游戏 | |
| 后置条件 | | 用户选择本局游戏使用的道具 | |
| GameUI. selectItem | 语法 | | Private void selectItem(int itemNum) | |
| 前置条件 | | 用户已选择游戏模式 | |
| 后置条件 | | 系统初始化棋盘，跳转到游戏界面 | |

#### 表19 Model层user模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| UserModel.login | 语法 | public String login(String userName,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果 |
| UserModel.  changePassword | 语法 | public String changePassword(String userName,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果 |
| UserModel.register | 语法 | public boolean register(String userName,String password,String gender) |
| 前置条件 | 各项信息符合输入规则 |
| 后置条件 | 查找是否有重复，返回添加结果 |
| UserModel.isIDExist | 语法 | public boolean isIDExist(String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回查找结果 |
| UserModel.findUser | 语法 | public UserPO findUser(UserPO userPO); |
| 前置条件 | 用户查找好友 |
| 后置条件 | 查找是否有相应的User，返回查找结果 |
| UserModel.getSelfInfor | 语法 | public UserPO getSelfInfor(String id) |
| 前置条件 | 用户已成功登录 |
| 后置条件 | 返回该用户的个人信息 |
| UserModel.  checkGameRecord | 语法 | public GameRecord checkGameRecord(String id) |
| 前置条件 | 用户已成功登录 |
| 后置条件 | 返回该用户的游戏记录 |
| **需要的服务（需接口）** | | |
| UserUI.loginRefresh | 语法 | Private void loginRefresh() |
|  | 前置条件 | UserModel模块返回注册成功 |
|  | 后置条件 | 由注册界面跳转到登录界面 |
| UserUI.loginFail | 语法 | Private void loginFail() |
|  | 前置条件 | UserModel模块返回登录失败 |
|  |  | 界面提示登录失败，请重新登录 |
| RegisterUI.refresh | 语法 | Private void refresh() |
|  | 前置条件 | 用户选择注册功能 |
|  | 后置条件 | 跳转到注册界面 |
| RegisterUI.fail | 语法 | Private void fail() |
|  | 前置条件 | UserModel模块返回用户已存在 |
|  |  | 界面提示用户已存在，请重新填写注册信息 |

# 6 信息视角

## 6.1 数据持久化对象

系统的PO类就是对应的相关的实体类，在此只做简单的介绍。

UserPO类包含用户的用户名、密码、性别、积分以及以往的游戏记录。

friendsPO类包含用户的好友列表以及对应的昵称。

## 6.2 持久化保存

持久化对象采用序列化的方法进行保存。

# 7 变更历史

变更历史如表20所示。

#### 表20 变更历史表

|  |  |  |
| --- | --- | --- |
| **变更者** | **变更内容** | **变更日期** |
| 徐仪萍 | 添加了Friend模块几个层的接口，用于对好友申请的处理 | 2014-4-27 |