**面向家庭的俄罗斯方块娱乐系统**

**System Design**

**组长：苗子建**

**组员：毕浥尘**

**刘宇航**

**李鸿祺**

**许阳**

目录

[0.项目Github仓库地址 3](#_Toc38300358)

[1.设计要求 3](#_Toc38300359)

[1.1游戏内容模块 3](#_Toc38300360)

[1.1.1单人俄罗斯方块游戏逻辑设计 3](#_Toc38300361)

[1.1.2单人俄罗斯方块游戏逻辑的扩展 5](#_Toc38300362)

[1.1.3人合作游戏逻辑设计 5](#_Toc38300363)

[1.1.4人对抗游戏逻辑设计 6](#_Toc38300364)

[1.2数据库管理模块 6](#_Toc38300365)

[1.2.1数据库表的整体设计 6](#_Toc38300366)

[1.2.2数据库表的详细设计 7](#_Toc38300367)

[1.3界面设计模块 11](#_Toc38300368)

[1.3.1 界面转换关系 11](#_Toc38300369)

[1.3.2 登录、注册界面设计 11](#_Toc38300370)

[1.3.3 主界面设计 12](#_Toc38300371)

[1.3.4 游戏界面设计 13](#_Toc38300372)

[1.4网络通信模块 14](#_Toc38300373)

[1.4.1状态同步 14](#_Toc38300374)

[1.4.2客户端逻辑结构 15](#_Toc38300375)

[1.4.3服务器逻辑结构 15](#_Toc38300376)

[2.Sprint(冲刺开发)任务 16](#_Toc38300377)

[2.1游戏内容模块 16](#_Toc38300378)

[2.2数据库管理模块 17](#_Toc38300379)

[2.3界面设计模块 17](#_Toc38300380)

[2.4网络联机模块 17](#_Toc38300381)

# 0.项目Github仓库地址

<https://github.com/GeniusMiaozz/Tetris-entertainment-system-for-family>

# 1.设计要求

## 1.1游戏内容模块

### 1.1.1单人俄罗斯方块游戏逻辑设计

1.方块与地图的联系

经典的俄罗斯方块中一共有七种形状的方块，形状分别类似“I、J、L、O、S、T、Z”每一种的方块由几个形状、大小均完全相同正方形小方块组成。如下图：



图：七种形状的“俄罗斯方块”

将游戏地图设定为一个二维数组，一个“俄罗斯方块”（以下简称方块组）中的一个正方形小方块（以下简称为小方块）占一个单位。即二维数组中的每一个元素为一个小方块。

2.生成游戏地图与方块组

游戏地图初始化为一个二维数组，整个二维数组就是方块可存在的而区域，数组的元素为一个小方块。用一个一维数组来存储七种方块组，这个一维数组中的每一个元素是一个方块组，通过生成随机数的方式在游戏地图中的指定位置生成方块组。

3.判断方块是否在边界内

在左右移动方块、方块自然下落或手动加速下落时，需要获取方块组中每个子方块的坐标、边界的坐标，进行比较以判断方块组是否在边界内。若方块组中的所有小方块均在游戏地图的边界内，则更新坐标；如果不在边界内，进行相应处理。

相应处理办法简述：如果是因为旋转之后小方块出现在了边界外，则逆向旋转回之前的状态；如果是因为移动之后小方块出现在了边界外，则逆向移动至之前的状态。

4.判断是否碰到其他方块

遍历游戏地图的二维数组，检测方块组要到达的地方是否为空，若为空，

继续移动，更新方块组位置的坐标数据；若不为空，停止本次移动。这一操作应在方块实际更换位置之前进行。

5.判定方块组位置的改变

方块组由始至终均需要进行自然下落，从方块组生成开始计时，在每一帧进行时间的检测，如果当前时间与上一次记录的时间（初始记录时间为0）超过了1秒，则进行一次二维数组中数据的更新，如果新数据（即当前方块组的新坐标）未超出边界也为碰到其他方块，则实际更新这一次方块移动操作。

对于玩家的操作（包括加速下落、左方向移动、右方向移动、逆时针旋转、顺时针旋转），同样进行一次二维数组中数据的更新，如果新数据（即当前方块组的新坐标）未超出边界也为碰到其他方块，则实际更新这一次方块移动操作。

6.检查横向上方块是否满行

在游戏地图中，所有位于同一行的方块的纵坐标均相同，遍历数组中纵坐标相同的数据，如果均被小方块填满，即均不为空，则当前纵坐标位置的一行已满行。

7.检查纵向上方块是否满列

在游戏地图中，每一次生成新方块组的时候，检测新方块组中的各小方块坐标是否超出上边界。如果方块组中有任一小方块的坐标超出了边界，则方块存在满列，游戏结束。

8.删除填满的行

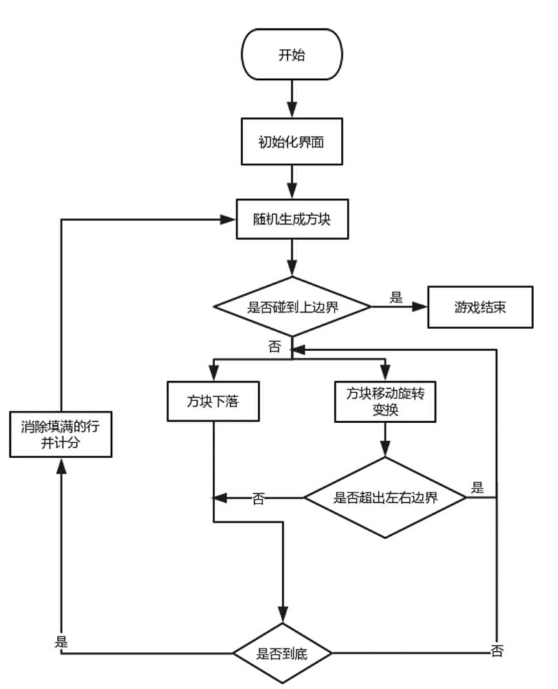
首先删除满行的数组元素，将数组中该行的数据全部值为空。然后依次遍历上方的行，将所有行的纵坐标减一，达到下落的效果。

9.提示即将出现的方块组

如果是第一次生成方块组，则生成两个随机数，一个是当前生成的方块组的编号，一个是下一个待生成的方块组的编号；若不是第一次生成方块组，则将下一个待生成的方块组的编号赋值给当前的方块组，再生成一个随机数赋值给下一个方块组的编号。

10.提示当前分数

每当成功填满一行并消行后，进行分数的累加统计工作。



图：基本俄罗斯方块游戏逻辑图

### 1.1.2单人俄罗斯方块游戏逻辑的扩展

Design1：玩家操作与实际效果反转。如：当玩家按下控制方块组向左移动的按键时，方块组实际会向右移动；当玩家按下控制方块组顺时针旋转的按键时，方块组实际会进行一次逆时针旋转。

Design2：改变方块下落速度。对控制方块下落一次的时间间隔变量进行改变，该变量设置越小，则方块下落速度越快。实际中可根据玩家选择“方块下落速度慢、正常、快、超快”进行变量不同值的设定。

### 1.1.3多人合作游戏逻辑设计

Design1：一人控制方块移动、另一人控制方块旋转模式

该模式下，系统为两个玩家分配同一场游戏对局的信息，对于一个玩家关闭其控制方块顺时针、逆时针旋转的功能；对于另一个玩家关闭其控制方块移动的功能，每个玩家的操作都会在自己的游戏界面及另一个玩家的游戏界面得到展示，二人合作完成单人游戏模式下的所有内容。

Design2：一人方块可见但不可操作、另一人方块可操作但不可见模式

该模式下，系统为两个玩家分配同一场游戏对局的信息，对于一个玩家关闭其控制方块移动、旋转、加速下落等一切操控方块的功能，此玩家为指挥玩家，指挥玩家只能看到方块的下落状况，但无法改变，“可读不可写”；对于另一个玩家关闭其可见方块的功能，该玩家为操作玩家，其游戏界面中不显示方块，只会显示游戏地图、分数等其他信息，“可写不可读”。指挥玩家通过观察界面中的方块信息告诉操作玩家如何进行下一步操作，操作玩家的操作都会在自己的游戏界面及指挥玩家的游戏界面得到展示，二人合作完成单人游戏模式下的所有内容。

### 1.1.4多人对抗游戏逻辑设计

Design1：实时战况可视化模式

该模式下单人的游戏逻辑与单人模式无差异。建立一个游戏房间，所有准备对战的玩家均加入这个房间，待所有人准备好后在同一时间开始游戏。游戏系统要负责每次生成随机方块，并将生成情况的数据传给所有玩家，通过同样的时间、同样的内容确保游戏公平性。同时，游戏系统还要为每个玩家展示出一个实时排行榜，排行榜上会显示所有当前对战玩家已获得的分数。

Design2：实时战况不可视化模式

在实时战况可视化的基础上，取消掉排行榜展示这一功能，在游戏计时计数后，再为所有玩家展示所有玩家各自的分数。

Design3：结合多人合作的多人对抗模式

多个玩家进行合作模式游戏，然后多组玩家进行对抗模式游戏。游戏逻辑无较大差异，系统需要正确处理不同玩家之间的关系。

## 1.2数据库管理模块

### 1.2.1数据库表的整体设计

系统中需要将在游戏中注册的家庭和家庭成员的用户信息，以及他们的游戏信息存储在数据库当中，方便管理以及存取。数据库设计分为六张表，分别为用户表、家庭表、地区表、好友表、游戏成就表和历史记录表，每张表的具体内容见下表所示：

表1.2.1 数据库表存储信息简介

|  |  |  |
| --- | --- | --- |
| 序号 | 表名(英文) | 说明 |
| 1 | 用户（user） | 存储每个用户的基本信息 |
| 2 | 家庭（family） | 存储用户所属的家庭信息 |
| 3 | 地区（region） | 存储家庭所在的地区信息 |
| 4 | 好友（friends） | 存储具有好友关系的两个用户的信息 |
| 5 | 游戏成就（gameachievement） | 存储与游戏成就相关的信息 |
| 6 | 历史记录（history） | 存储用户的历史游戏相关信息 |

### 1.2.2数据库表的详细设计

（1）用户表

用户表中主要存储用户个人信息，包括用户名、密码、头像、个人简介、游戏相关信息等，同时也与家庭表相关联，具体设计如下表所示：

表1.2.2.1 用户表详细设计

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 属性含义 | 备注 |
| u\_id | char(8) | 用户的ID号 | 主键 |
| u\_name | char(20) | 用户名 | 非空 |
| u\_password | char(20) | 用户密码 | 非空 |
| u\_sex | char(4) | 性别 | 非空 |
| u\_headportrait | mediumblob | 用户头像 |  |
| u\_introduction | char(50) | 用户个人简介 |  |
| u\_familyid | char(8) | 用户家庭ID | 非空 |
| u\_identity | char(6) | 家庭身份 |  |
| u\_gametime | char(10) | 游戏时长 |  |
| u\_recentlogintime | char(15) | 最近登录时间 |  |
| 其它约束：  1）性别只能为“男”、“女”和“神秘”  2）用户家庭ID为外键，关联家庭表中的f\_id | | | |

（2）家庭表

家庭表中主要存储注册的家庭的相关信息，包括家庭名，家庭密码，家庭头像，家庭所在地区的相关信息，具体设计如下表所示：

表1.2.2.2 家庭表详细设计

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 属性含义 | 备注 |
| f\_id | char(8) | 家庭的ID号 | 主键 |
| f\_name | char(20) | 家庭名 | 非空 |
| f\_password | char(20) | 家庭密码 | 非空 |
| f\_headportrait | mediumblob | 家庭头像 |  |
| f\_introduction | char(50) | 家庭简介 |  |
| f\_regionid | char(8) | 家庭所在地区ID | 非空 |
| 其它约束：  1）家庭所在地区ID为外键，关联地区表中的r\_id | | | |

（3）地区表

地区表中主要存储家庭所在地区的相关信息，具体设计如下表所示：

表1.2.2.3 地区表详细设计

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 属性含义 | 备注 |
| r\_id | char(8) | 地区的ID号 | 主键 |
| r\_name | char(10) | 地区名 | 非空 |
| 其它约束：无 | | | |

（4）好友表

好友表主要存储的是所有用户之间的对应关系，由于一个用户可以有多名好友，因此同一个用户可以有多名好友，并且表中存储的是单向好友关系，如果两个用户互为好友，则在表中需要有两条相应的记录表示互为好友关系。

好友表具体设计如下：

表1.2.2.4 好友表详细设计

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 属性含义 | 备注 |
| uf\_id | char(8) | 好友关系ID号 | 主键 |
| uf\_userid | char(8) | 用户ID号 | 非空 |
| uf\_friendid | char(8) | 好友的ID号 | 非空 |
| 其它约束：  1）用户ID号为外键，关联用户表中的u\_id;  2）好友的ID号为外键，关联用户表中的u\_id | | | |

（5）游戏成就表

游戏成就表当中存储用户的游戏成就相关的信息，包含了用户在不同难度下获得的最高成绩以及获取时间，具体设计如下表所示：

表1.2.2.4 游戏成就表详细设计

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 属性含义 | 备注 |
| g\_id | char(8) | 游戏成就ID号 | 主键 |
| g\_userid | char(8) | 用户ID号 | 非空 |
| g\_simplescore | char(10) | 简单模式最高分 |  |
| g\_simpletime | char(15) | 简单模式最高分  获得时间 |  |
| g\_ordinaryscore | char(10) | 普通模式最高分 |  |
| g\_ordinarytime | char(15) | 普通模式最高分  获得时间 |  |
| g\_difficultscore | char(10) | 困难难度最高分 |  |
| g\_difficulttime | char(15) | 困难难度最高分  获得时间 |  |
| g\_hellscore | char(10) | 地狱难度最高分 |  |
| g\_helltime | char(15) | 地狱难度最高分  获得时间 |  |
| 其它约束：  1）用户ID号为外键，关联用户表中的u\_id; | | | |

（6）历史记录表

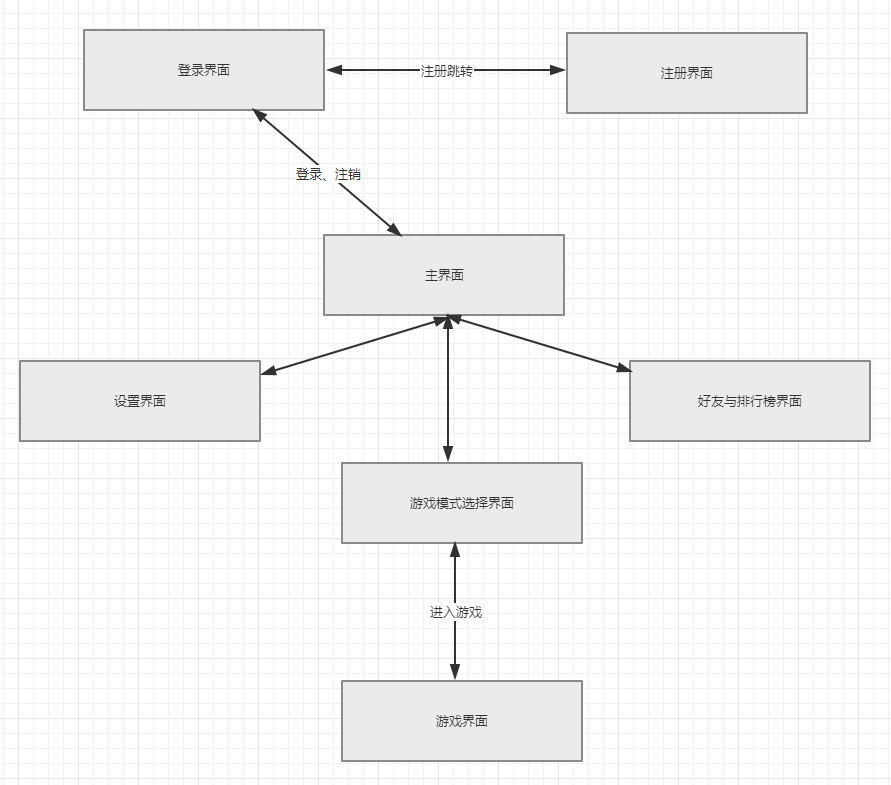
历史记录表当中主要存储用户近期的历史游戏记录，便于用户查看，详细设计如下表所示：

表1.2.2.4 历史记录表详细设计

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 数据类型 | 属性含义 | 备注 |
| h\_id | char(8) | 历史记录ID号 | 主键 |
| h\_userid | char(8) | 用户ID号 | 非空 |
| h\_playtime | char(15) | 游戏时间 | 非空 |
| h\_gamescore | char(10) | 游戏分数 | 非空 |
| h\_degree | char(8) | 游戏难度 | 非空 |
| 其它约束：  1）用户ID号为外键，关联用户表中的u\_id;  2）游戏难度只能从“简单”“普通”“困难”“地狱”中进行选择。 | | | |

## 1.3界面设计模块

### 1.3.1 界面转换关系

****

### 1.3.2 登录、注册界面设计

登陆的界面采用两级登陆形式，先进行家庭登陆再进行成员选择。

首次登录会提醒用户进行注册，注册界面会进行家庭的注册再进行家庭成员的添加。

注册之后的家庭成员添加与删除或是修改，在二级登陆界面跳转至注册界面完成。

注册界面与登录界面的结构如下图所示：



### 1.3.3 主界面设计

通过主界面可以进入排行榜界面查看好友信息以及排行榜信息

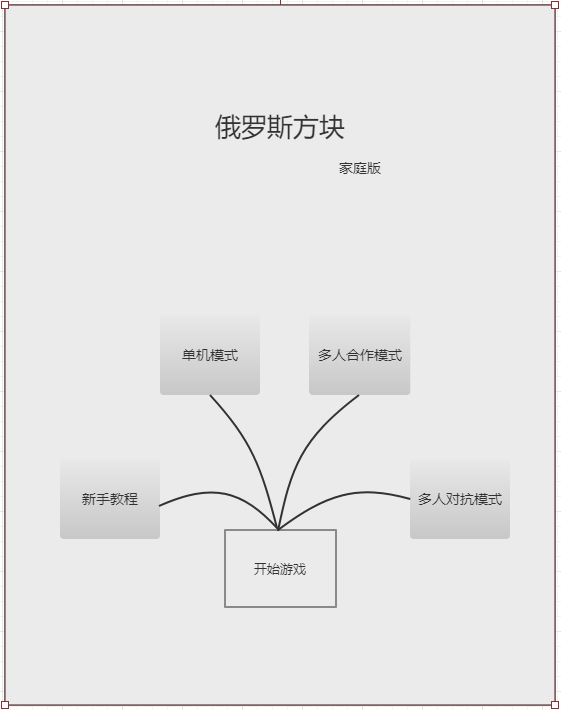
进入设置界面可以调整语言、音乐、画面等设置

点击开始游戏则可以开始进行模式选择



### 1.3.4 游戏界面设计

游戏的选择界面提供四种选择：



游戏内的界面大致为下图：

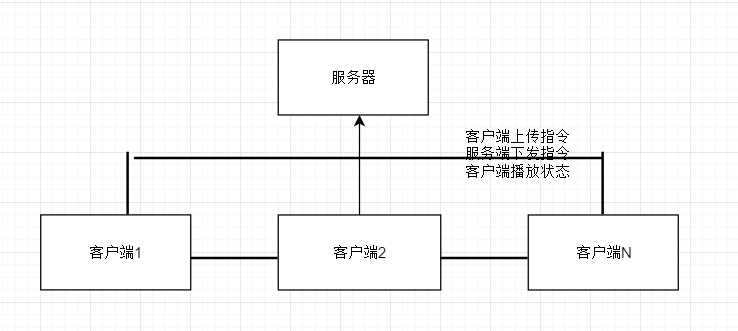
根据不同的模式会有不同的模式界面的特点



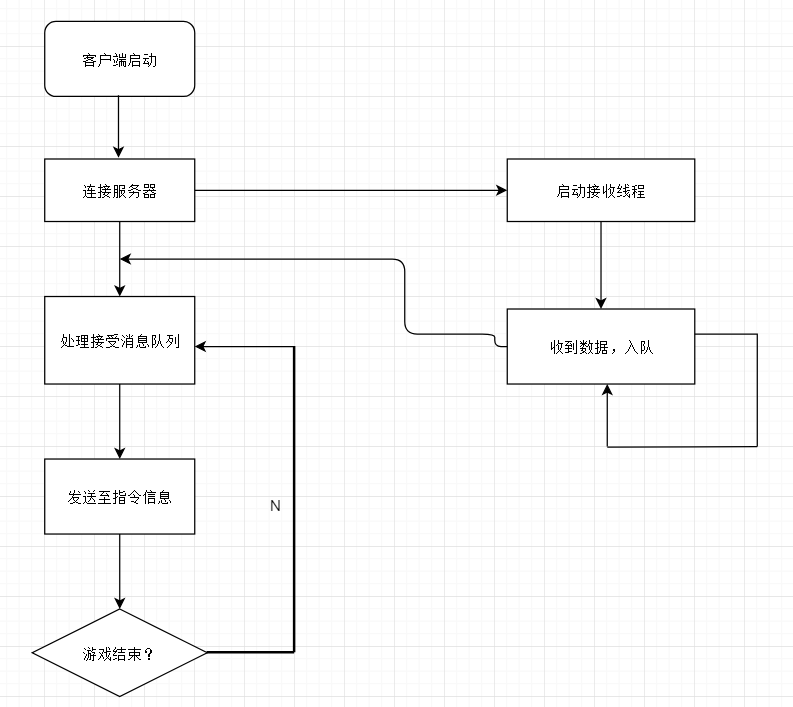
## 1.4网络通信模块

### 1.4.1状态同步

简单的来理解就是所有的数据在服务端进行计算和校验，客户端将操作上发到服务器，服务器不断的告诉客户端计算结果，由客户端进行展现。

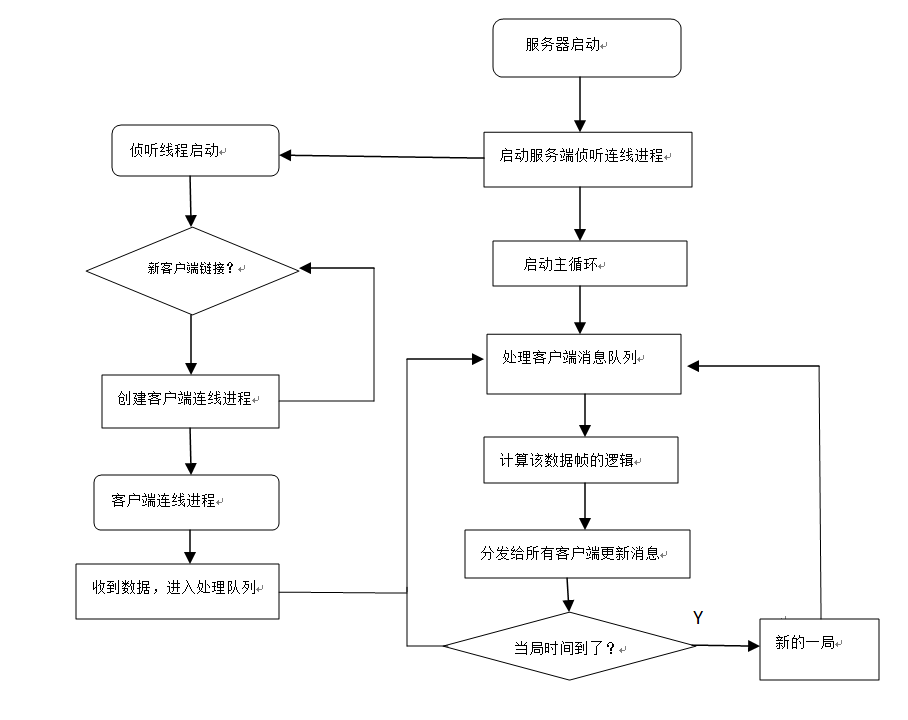


### 1.4.2客户端逻辑结构



### 1.4.3服务器逻辑结构

通信协议使用tcp，那么服务端就是一个典型的处理连接、处理请求并分发数据的逻辑结构。



# 2.Sprint(冲刺开发)任务

## 2.1游戏内容模块

1.完成游戏地图与方块组内容等基础内容的开发

2.完成检测方块组位置是否合理的开发

3.完成处理方块正确移动的开发

4.完成检测方块行、列是否满方块的开发

5.完成删除满行并处理其它方块下移的开

6.完成满列时正确结束游戏的开发

7.完成下一个方块信息显示的开发

8.完成分数信息的显示、更新开发

9.完成其它单人游戏辅助内容的开发

10.完成单人游戏扩展内容的开发

11.完成多人合作模式内容的开发

12.完成对人对抗模式合作的开发

13.完成与其他功能对接的整合

## 2.2数据库管理模块

1.完成数据库中关系表的创建

2.完成各张数据表之间的关系约束

3.完成数据库数据的增删查改语句设计

4.完成家庭和用户注册时数据存入数据库的开发

5.完成家庭和用户修改基本信息时数据库更新的开发

6.完成游戏结束后将游戏记录写入数据库的开发

7.完成用户打破自己记录数据库更新的开发

8.完成用户查看好友排行读取数据库的开发

## 2.3界面设计模块

1.完成各个界面的框架开发

2.完成各个界面的控件内容开发

3.完成各个界面的布局开发

4.完成各个界面的元素美化开发

5.完成界面之间的逻辑开发

6.完成界面内功能实现的开发

7.完成设置界面中的画面设置的开发

## 2.4网络联机模块

1.完成架构设计

2.完成多线程交互问题

3.完成断网时的检测

4.完成游戏通信协议

5.完成发送队列解决操作速率，发送速率，接受速率的不匹配问题。

6.完成断网离线时的操作处理

7.完成速度优化