# 第十三组圈地大作战设计文档

## 前端设计

### 1.编写目的

该文档的目的是说明圈地大作战的功能和设计方向目标等描述游戏项目的整体框架的实现。

### 2.功能描述

1. 界面设计技术

选取了比较宏大的图片作为背景，颇具古风的建筑作为主页面的选项按钮。游戏界面简洁清楚，在主页面上有相应的游戏模块的划分，不同模块对应不同的模式或功能。简洁优美的画风，通过相应的美术设计，包括界面的整体布局，各个图标的大小，颜色，形状，操控时的提示等等都需要让用户感到赏心悦目，进而提升游戏的艺术性和可玩性。在游戏中，我们添加了摇杆来控制圈地时的方向，由于摇杆控制在手游中也更加符合用户习惯，而且使得操作简单化和精细化，是一个非常好的选择。在游戏中，各种道具和商店的使用也操作也要设计得简单易上手，可以通过相关得提示来指导用户进行操作。

1. 游戏功能划分

游戏的基本功能包括菜单处理，程序逻辑处理，数据处理。

1. 菜单处理包括进入游戏，创建房间，进入商店等
2. 程序的逻辑处理就是涵盖了除菜单，数据外几乎所有功能。
3. 数据处理就是记录游戏数据，排名以及商店数据
4. 游戏流程图

主菜单

创建房间

人机训练

实时对战

进入商店

购买道具

选择对战模式

进入游戏

结束游戏后可回到主菜单查看排名

1. 功能描述

当玩家打开程序时，在主页面有以下几个选项：

1. 实时对战：选择模式后随机匹配对手进入游戏
2. 创建房间：选择模式后可以邀请好友一起对战
3. 人机训练：选择模式后与机器人进行对战
4. 排行榜：游戏结束后可通过排行榜看自己的名次
5. 商店：右上角有自己的血量以及金钱，可在商店购买复活次数和加速道具
6. 游戏设计

游戏总共有3种模式，3v3，2v2，1v1，共有两种颜色。

3v3的地图是六边形，2v2和1v1的地图是正方形。2v2的地图比1v1的大一些，地图以墙壁为边界。

玩家会生成在地图顶点，玩家需要在一定的时间内尽可能多的扩大自己的领地。当玩家在移动时，自己走过的路会被当成自己的领地，当自己走了一个圈后，圈内全部成为自己领地。同时自己和敌人能够互相侵占领地，如果自己被敌人圈起来，那么就会死亡；当你割断了对方的线，使对方不与他或他的队友的出生点联通时，对方死亡。死亡后会有5秒复活时间。

你和队友的线形成的环也可以圈成地。

当两个人迎面相撞时，如果倾角是180°，两人都不会死，否则一方会死。如果撞到墙，也不会死，而是会贴着墙壁走。

地图为白色地板，玩家圈起来的地为棕色。

游戏的唯一结束条件是时间结束，一局大概5分钟，结束时按照圈地的面积计算积分。

游戏中有一种货币。如果游戏胜利可增加货币，游戏失败不会减少货币。商城中有两种道具，一种是加速道具：拖动的速度变为2倍 另一种是立即复活道具：点击后立即复活，不用等5秒。这些道具需要在游戏开始前购买，然后在游戏中点击使用。

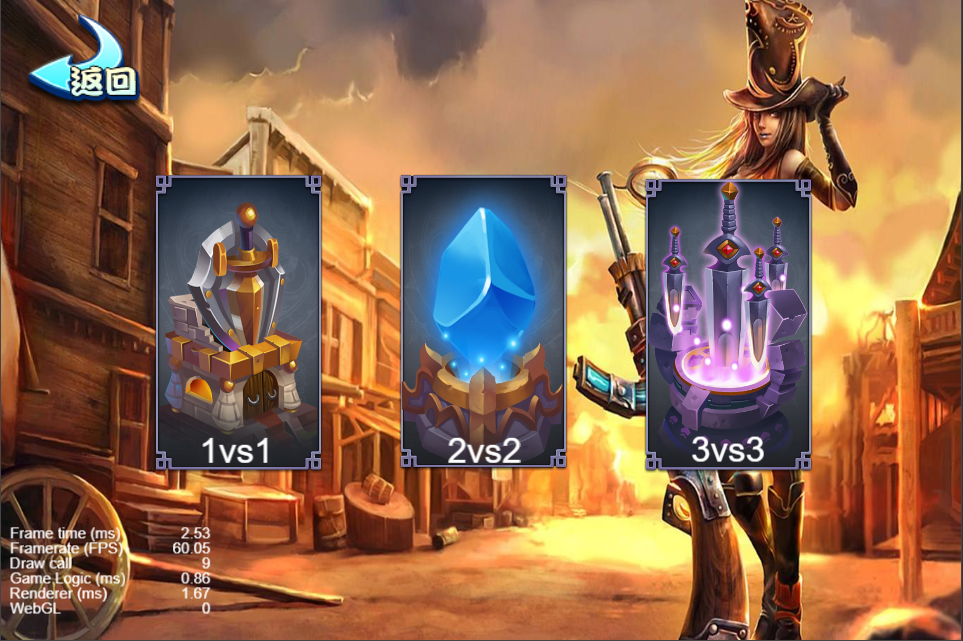
游戏有房间系统，可以邀请好友来进行游戏。

每次游戏结束后，都可以去排行榜看自己上一把游戏的排行。

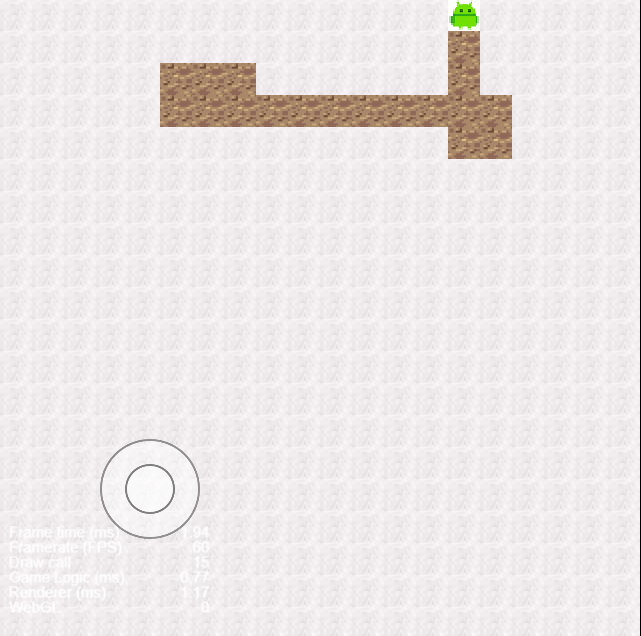
主页面如下：



选择模式页面如下：



游戏页面如下：

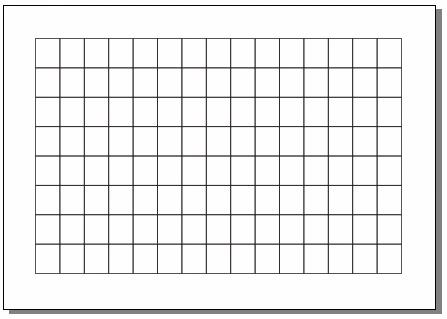


## 后端设计

### 1.游戏抽象

首先，选取了几款同类型的游戏进行试玩，决定使用栅格化形式对地图进行划分（不显示呈现栅格），玩家每次的行动为向某个方向移动一格，即如下：

可以针对玩家数和画质对栅格数进行调整



每个格有两个属性（值为0则是未涂色）（多个玩家颜色用1，2，3即可）

1 base（实色，领地）

2 face（虚色，走过但还未占领）

只需记录这个n\*n的棋盘就行了

### 2.主要算法之圈地算法

**阶段一：**

最开始考虑扫描线算法，即一行一行进行扫描，会产生两个交点，交点内即为封闭区，但此算法对不封闭区间会误判。此外，最重要的是，考虑到圈地游戏可能经常会同时产生多个封闭区域，扫描线算法解决此类情况难度较大，换为搜索算法。

**阶段二：**

考虑搜索算法

1. 从一个未被涂色的块开始，进行深度优先搜索。
2. 当遇到被涂色块或边界时回退，被搜索过的块要进行标记避免进行重复搜索。
3. 搜索完本片区域所有可探索到的块后，从下一个未被涂色且未被标记的块进行搜索，直到没有这样的块。
4. 至此，整个地图被搜索完毕，被划分为若干个连通片，联通片能到边界，则不是闭区域，否则则是闭区域。
5. 经游戏测试与分析，当碰到自己的领地时构成圈地条件，（涂色阶段1）此时首先将走过的路径涂实，（涂色阶段2）再将闭联通片涂实即完成圈地。
6. 以上为单个玩家的描述，多玩家只需各自针对自己的颜色执行一次算法即可。

**阶段三：**

对边界情况的考虑，分析可以看出，边界只有在玩家经过，并且对路径变实的时候变为领地，不可能是被圈在里面成为领地的情况，对搜索算法无影响。

对于特殊情况的考虑，向外走再原路返回，也会变为自己的领地，显然该情况在涂色阶段一得到了解决。

此外，多人游戏时存在杀死其他玩家情况，此类情况简单，在玩家碰到另一玩家虚线路径时触发，清除被触碰玩家全部实线和虚线，玩家死亡。

**阶段四：**

对时间性能进行考量

由于使用了标记，在每次算法执行时每个未被涂色方格最多执行一次，如果划分为n\*n的方格，那么每次搜索数都<=n\*n

同时使用了算法触发判断条件（当玩家碰到自己的领地），所以算法不会无意义的执行，只会执行较少的次数，大大提高了效率

### 后端使用的主要技术和工具：

javascript(nodejs)

cocos creator(v2.3.3)

java(jdk8,eclipse2020)

tomcat(v8.0),websocket (服务器端)

mysql 8.0.19 (数据库)

### 后端的设计思路：（单人游戏的实现思路）

1. 打开游戏：

打开游戏时让客户端与服务器连接。数据库连接设计成连接池模式。

为什么使用连接池：

　数据库连接是一种关键的有限的昂贵的资源，这一点在多用户的网页应用程序中体现得尤为突出。 一个数据库连接对象均对应一个物理数据库连接，每次操作都打开一个物理连接，使用完都关闭连接，这样造成系统的性能低下。

连接池模式介绍：

数据库连接池的解决方案是在应用程序启动时建立足够的数据库连接，并讲这些连接组成一个连接池(简单说：在一个“池”里放了好多半成品的数据库连接对象)，由应用程序动态地对池中的连接进行申请、使用和释放。对于多于连接池中连接数的并发请求，应该在请求队列中排队等待。并且应用程序可以根据池中连接的使用率，动态增加或减少池中的连接数。

　　连接池技术尽可能多地重用了消耗内存地资源，大大节省了内存，提高了服务器地服务效率，能够支持更多的客户服务。通过使用连接池，将大大提高程序运行效率，同时，我们可以通过其自身的管理机制来监视数据库连接的数量、使用情况等。

1. 登录游戏：

注册账号时客户端发送账号密码给服务器，服务器将其存入数据库中，并初始化货币和道具数目。而一旦登录游戏之后就需要将这个连接和账号绑定起来。

1. 游戏中：

圈地算法已解释过。

使用道具，购买道具都需要对数据库进行操作。

1. 游戏结束：

释放数据库连接和websocket的连接。