浙江大学计算机学院

Java 程序设计课程报告

2018—2019 学年　秋冬学期

题目	斗兽翻翻棋
学号	
学生姓名	
所在专业	
所在班级	

目　　录

1. 引言

本程序灵感来自“玩吧”app 的双人斗兽棋游戏，它的规则相比普通斗兽棋多了翻面棋子，只有正面朝上才能确定棋子势力，这个游戏主要考验运气，偶尔需要简单的思考，很适合消遣。

1．1　设计目的

斗兽棋是一个脍炙人口的游戏，本程序取名斗兽翻翻棋，是在其基础上增加了翻面未知身份的棋子的功能，增加了游戏的趣味性。本实验加深了我对 Java 的 GUI、多线程、网络通信的了解。

游戏规则如下：象>狮>虎>豹>狼>狗>猫>鼠，鼠能吃象；红方先手，蓝方后手。可以选择翻面灰棋或移动自家动物（只能上下左右走一步）；灰色为翻面状态，不可以吃翻面状态的棋子；当一方全灭则游戏结束。

(1) 主界面有单人游戏、双人游戏、联网对战三种模式。

(2) 选择单人游戏是和电脑进行对战，己方为红色先手，电脑充当蓝方低配玩家，可以修改电脑思考时间。

(3) 选择多人游戏是一个程序由两个人轮流控制，红方先手，蓝方后手，由用户自行分配。

(5) 选择联网对战需要连接服务器，由服务器收发两方的移动棋子行为，获胜状态在本地判断。

1．2　设计说明

本程序采用 Java 程序设计语言，在 Eclipse Oxygen 平台下编辑、编译与调试。图片和音效素材来自互联网。

2. 总体设计

2．1　功能模块设计

本程序需实现的主要功能有：

(1) 用户可以选择单人模式和 AI 下棋。

(2) 用户可以选择双人模式在本地进行双人对战。

(3) 用户可以选择联网对战与随机匹配的玩家进行游戏。

程序的总体功能如图 1 所示：

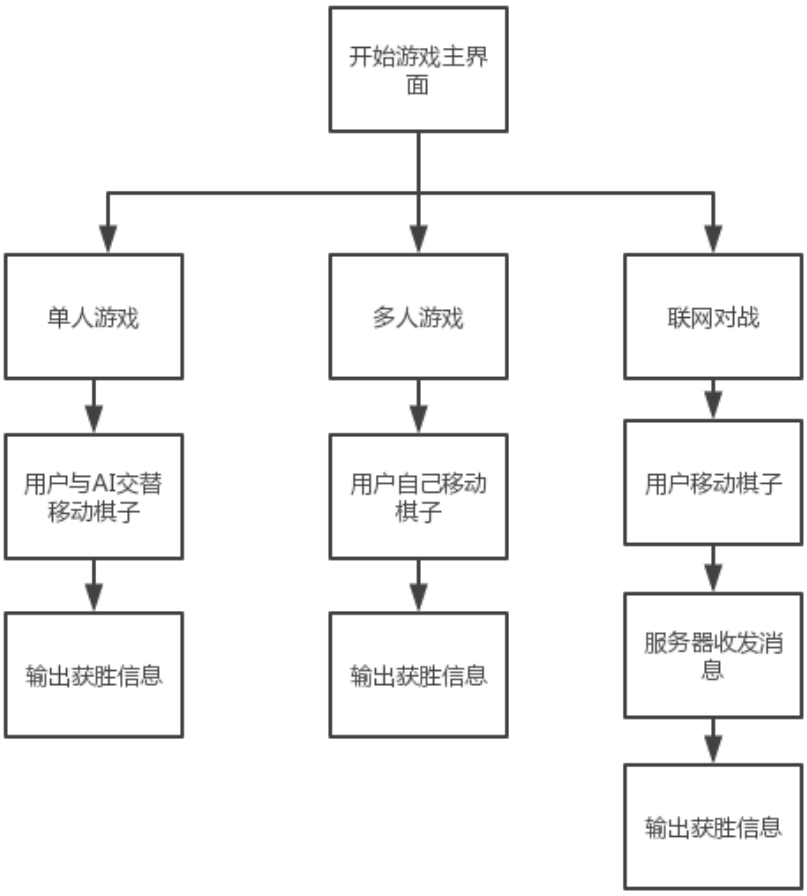

图 1 总体功能图

2. 2　流程图设计

程序总体流程如图 2 所示：

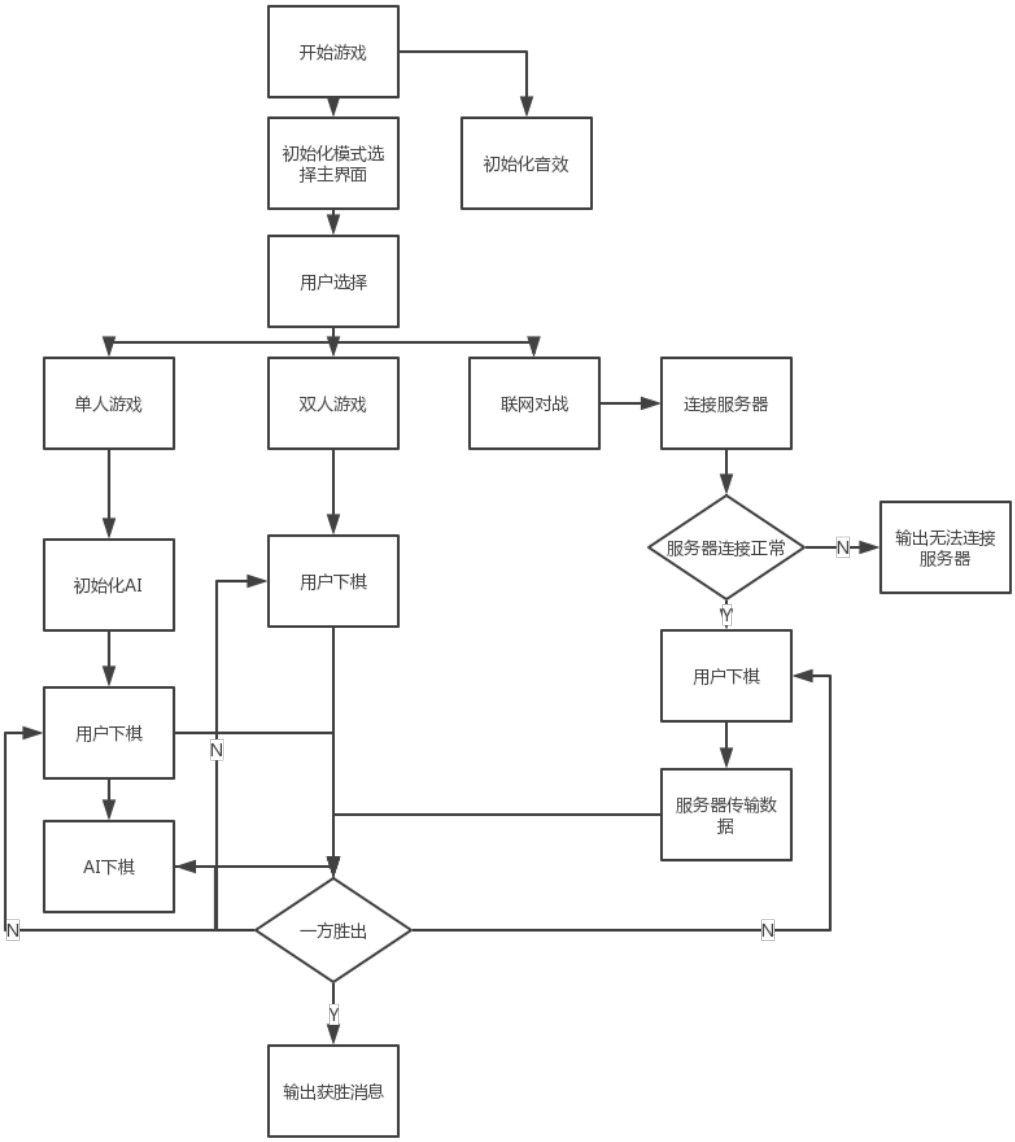

图 2 总体流程图

3. 详细设计

3．1 开始界面

开始界面 Layout 设置为网格，主要考虑只有一个背景图、一个“斗兽翻翻棋”标题和三个自上而下的按钮，每个按钮设置监听事件，分别进入单人模式、双人模式、联网模式。

表 3-1 开始界面的 CRC 卡

类：Start	
说明：	
新建开始界面	
职责	**协作者**
Start()布局	
Background()设置背景图	
Choose()设置三个按钮	Initial

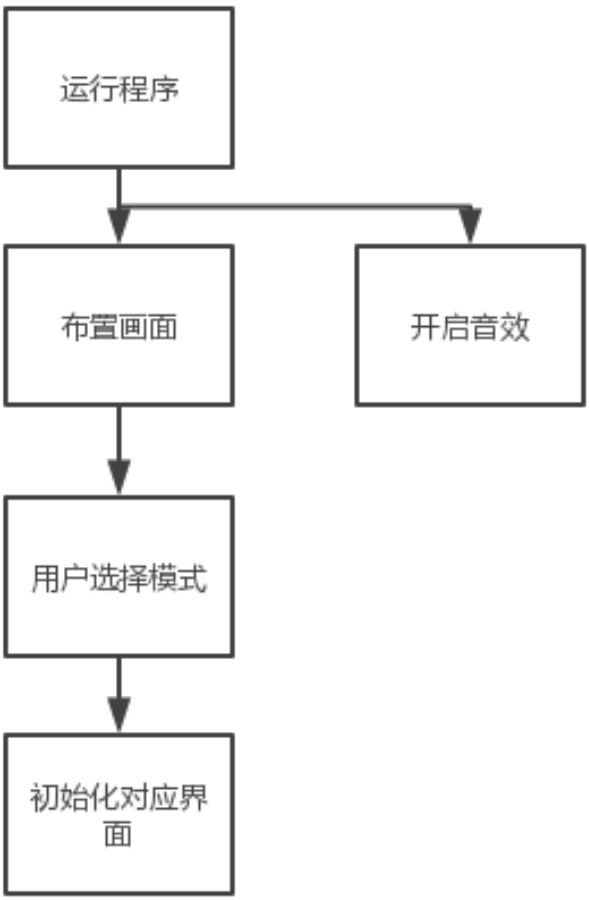

图 3-1 开始界面流程图

3．2 初始化游戏界面

Initial()由 start()函数启动，主要功能为布置棋盘，安放棋子，判断用户移动棋子的合法性，为每个棋子设置键盘监听事件，还有设置网格布局的函数、判断用户点击次数的函数等小函数。

表 3-2 初始化界面的 CRC 卡

类：Initial	
说明：	
新建棋盘界面	
职责	**协作者**
Initial()布局	
AddChesspiece()放置随机棋子	
GBC()设置网格布局参数	

Board()放置记分板	
actionPerformed()	

以下是 CRC 卡中有关数据和方法的详细说明：

（1）成员变量

① chess 作为棋盘的 Jpanel。

② recorder 作为记分板的 Jpanel。

③ Mode 设置在 start 界面选择的模式（单人、双人、联网）。

④ ConnectToServer 是连接服务器的类。

（2）方法

① Initial()生成生个布局。

② AddChesspiece()生成随机棋子。

③ actionPerformed()为每个棋子设置鼠标监听事件，判断棋子移动的合法性和后续动作。

流程图如图 3-2 所示：

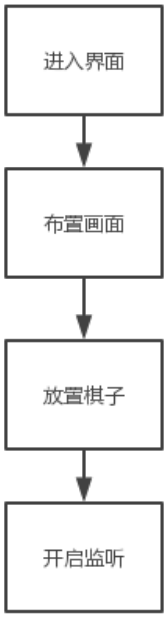

图 3-2 初始化游戏界面流程图

3．3　记分板界面

记分板界面显示目前状态，包括联网模式的 waiting 等待对手连接，red、blue 显示目前移动方，本类还包括一个根据棋盘棋子判断是否有人获胜的函数。

表 3-3 记分板界面的 CRC 卡

类：BoardText	
说明：	
新建记分板界面	
职责	**协作者**
BoardText()布局	
change()改变输出内容	
judge()判断是否有人胜出	

以下是 CRC 卡中有关数据和方法的详细说明：

（1）成员变量

① Turn 为记录红方回合或蓝方回合。

（2）方法

① change()改变输出内容。

② judge()判断是否有人胜出。

3．4 棋子类

棋子类为每个棋子设置事件，主要由棋子点击后发生，包括移除动物、隐藏动物、清除动物等。

表 3-4 棋子的 CRC 卡

类：Initial	
说明：	
棋子的功能	
职责	**协作者**
Chesspiece()生成棋子	
showAnimal()显示动物	
removeAnimal()移除动物	
hideAnimal()隐藏动物	

以下是 CRC 卡中有关数据和方法的详细说明：

（1）成员变量

① type 记录棋子类型，从-8 到 8 代表不同动物。

② width 记录棋子大小，由界面决定。

（2）方法

① showAnimal()显示动物。

② removeAnimal()移除动物。

③ hideAnimal()隐藏动物。

3．5 地图类

地图类储存棋子的状态，包括是否翻面、动物类型。

表 3-5 地图的 CRC 卡

类：Initial	
说明：	
棋子的功能	
职责	**协作者**
shuffle()打乱棋子顺序	

以下是 CRC 卡中有关数据和方法的详细说明：

（1）成员变量

① map 记录动物类型，从-8 到 8，0 为空位置。

② stateMap 记录动物状态，-1 为翻面，1 为正面，0 为空位置。

③ animalMap 储存 3.4 中的棋子类。

（2）方法

① shuffle()在初始化阶段打乱棋子。

3．6 AI 类

AI 类是在用户选择单人模式后的电脑动作，它作出判断之后能直接修改 Map 类并调用 Chesspiece 类中的函数来改变 UI。他的主要实现思路是扫描整个棋盘，查看己方棋子能否吃掉对方棋子，如果有则行动，如果没有则查看己方棋子能否

逃脱被对方棋子吃掉，如果有则心动，如果没有则任意翻面一个为翻面的棋子，如果都翻面了就任意走一步。

他的流程图如图 3-6 所示：

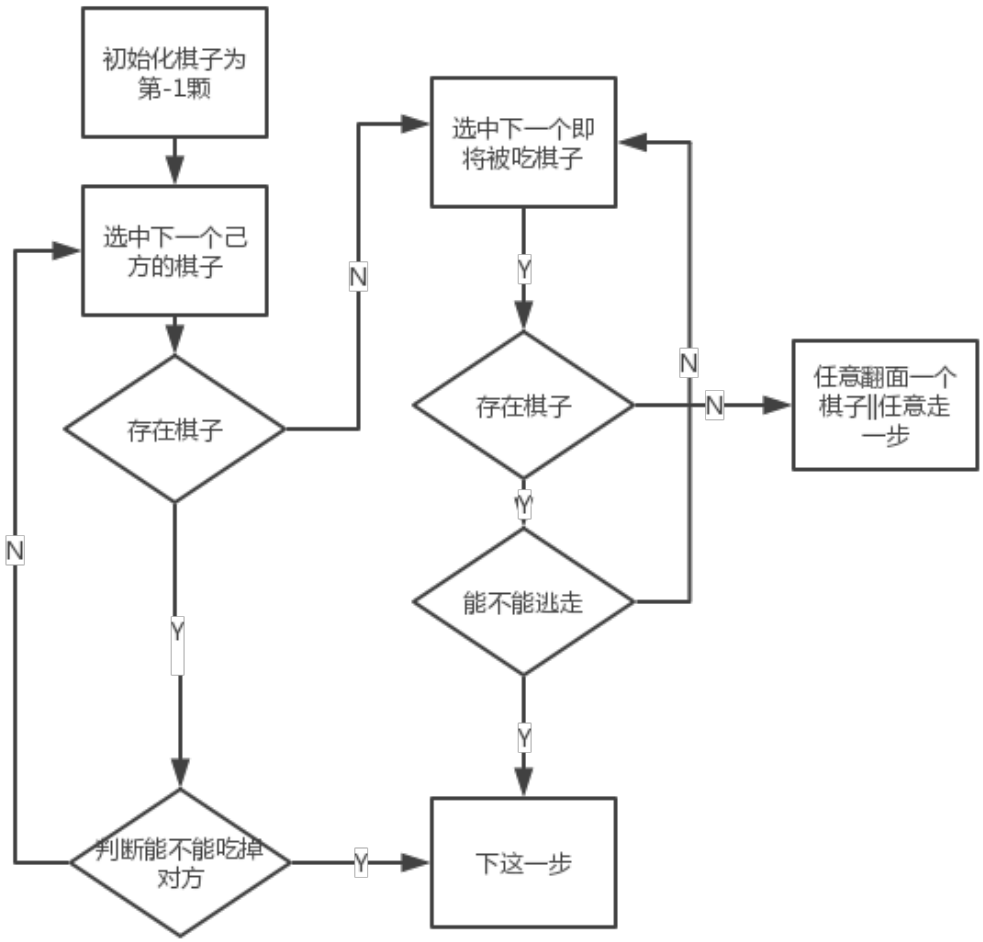

图 3-6 AI 流程图

3．7　连接服务器类

ConnectToServer 类记录服务器信息，连接服务器并接受服务器的消息再布置棋盘，同时能把地图信息和字符串相互转化并和服务器建立通信。

本服务器使用 python3 编写，主要考虑代码量小，方便测试。生成的 server.py 文件运行后才能进行联网对战。

表 3-7 连接服务器类的 CRC 卡

类：ConnectToServer

说明：	
新建棋盘界面	
职责	**协作者**
ConnectToServer()建立连接	
Connection()获取服务器信息	Map
passMesg()传递信息	
changeToStr()把地图信息转为字符串	
changeToMap()把字符串转为地图	

以下是 CRC 卡中有关数据和方法的详细说明：

（1）成员变量

① socket 建立 TCP 通信

② IP 和 PORT 记录服务器地址。

③ turn 记录行动人员。

（2）方法

① Connection()开启一个线程获取服务器信息并转化为 UI 棋子状态。

② passMesg()在下完一步后传递地图。

③ changeToStr()和 changeToMap()转化地图和字符串。

3．8　音乐类

本类由 Music 和 MusicMove 组成，都开启一个线程播放音乐，一个播放背景音，一个播放棋子移动的声音。

4. 测试与运行

4．1　程序测试

程序分模块从 Start()、Initial()、BoardText()、Chesspiece()、Map()、AI()、ConnectToServer() 完成编码或测试，合成后没有出现明显的错误和漏洞，但是由于时间和精力受限，没有完善一些细节方面的内容，比如在游戏中可以加入联

网玩家语音对话功能、倒计时功能等，在游戏完成的时候可以回退按钮等，还有就是可以扩展一下用户注册、登录的功能等。总的来说本次设计在单人、双人、联网功能上已经基本达到要求，其他细节方面有待完善。

4．2　程序运行

程序运行选择界面如图 4-1 所示：

图 4-1　程序选择面

初始化棋盘如图 4-2 所示：

图 4-2 棋盘

此时选择双人模式，翻开棋子后如图所示：

图 4-3 翻开棋子

其中，蓝方已经走完，消息框提示红方走。

如下图所示是本局战况：

图 4-4 下棋流程

之后是单人模式，每下一次后电脑会自动走一步，合乎逻辑，没有漏洞。但是仍然不够智能，玩家很容易获得胜利。

图 4-5 单人模式测试

然后是联网对战，初始化界面如图：

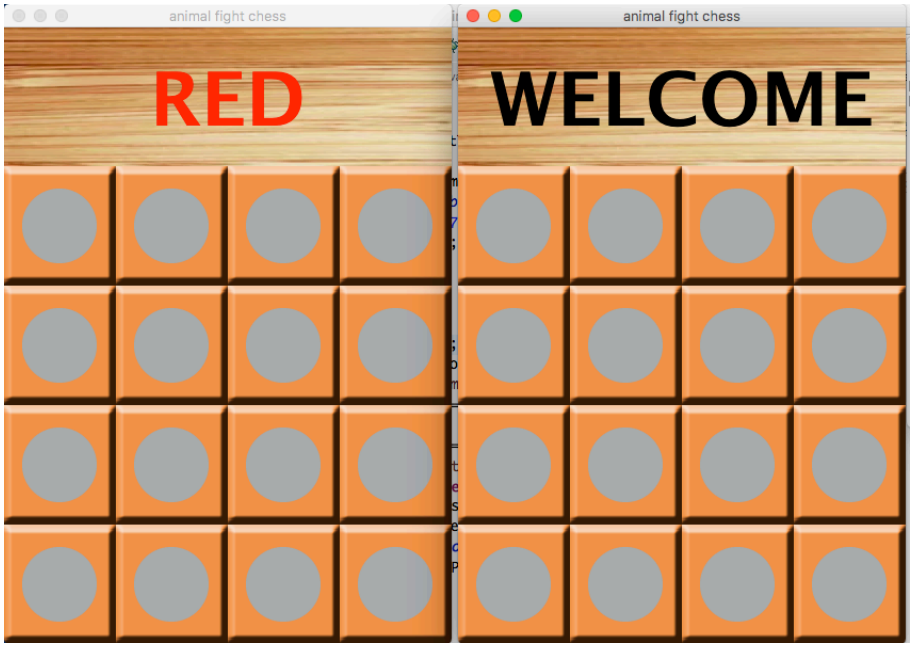

图 4-6 初始化联网界面

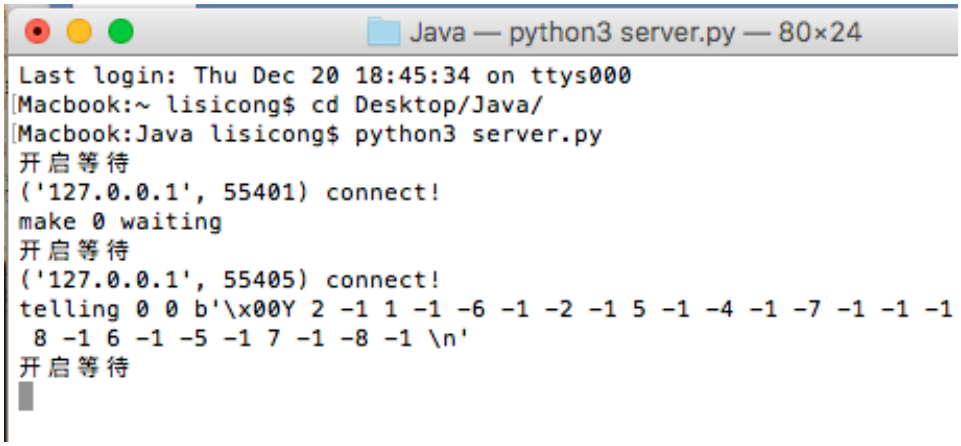

图 4-7 服务器输出

每走一步棋，两边都能同步。

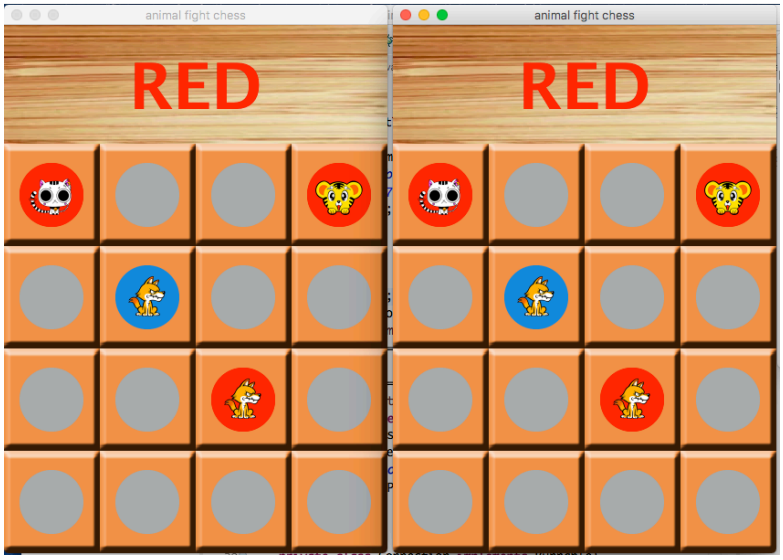

图 4-7 两边同步

服务器传递消息内容如图所示，包括连接客户端的 IP 和端口，传送的地图数据。

```
开启等待
('127.0.0.1', 55401) connect!
make 0 waiting
开启等待
('127.0.0.1', 55405) connect!
telling 0 0 b'\x00Y 2 -1 1 -1 -6 -1 -2 -1 5 -1 -4 -1 -7 -1 -1 -1 3 -1 -3 -1 4 -1
 8 -1 6 -1 -5 -1 7 -1 -8 -1 \n'
开启等待
telling 0 1 b'\x00X 2 1 1 -1 -6 -1 -2 -1 5 -1 -4 -1 -7 -1 -1 -1 3 -1 -3 -1 4 -1
8 -1 6 -1 -5 -1 7 -1 -8 -1 \n'
telling 0 0 b'\x00W 2 1 1 -1 -6 -1 -2 -1 5 -1 -4 -1 -7 -1 -1 -1 3 -1 -3 -1 4 1 8
 -1 6 -1 -5 -1 7 -1 -8 -1 \n'
telling 0 1 b'\x00V 2 1 1 -1 -6 -1 -2 -1 5 -1 -4 -1 -7 -1 -1 -1 3 -1 -3 -1 4 1 8
 -1 6 1 -5 -1 7 -1 -8 -1 \n'
telling 0 0 b'\x00U 2 1 1 -1 -6 -1 -2 -1 5 -1 -4 1 -7 -1 -1 -1 3 -1 -3 -1 4 1 8
-1 6 1 -5 -1 7 -1 -8 -1 \n'
```

图 4-8 服务器内容

如图为游戏结束，蓝方胜利界面。

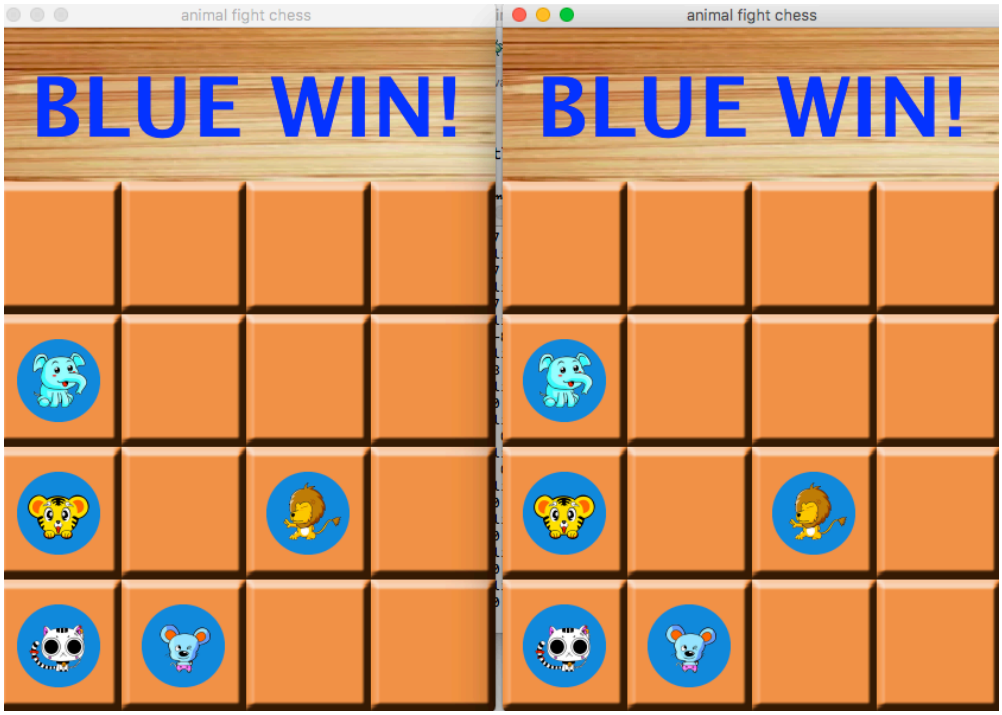

图 4-9 蓝方胜利

5. 总结

本实验是对 GUI、网络编程、并发线程、Photoshop 的综合使用，对我的编码能力有很大的提升。编码中也遇到不少问题。首先是对网格布局使用不熟悉，在棋盘上放棋子按钮之后棋盘会被拉伸。然后我调整了 ipadx、ipady 才能正常显示。然后是 Java 竟然不支持 mp3 的播放，但是转换成 wav 文件会变大，于是我导入了一个叫 jl 的 jar 包，相比内置的音频处理还是少了很多函数，一个 loop 需要自己写循环实现。之后是 python 的服务器向 Java 程序发送数据无论如何也接受不到，直到最后在数据末加\n 才能收到，这可能与 readline()需要读取\n 有关，但是当时没仔细思考，debug 错了方向。然后是 AI 的逻辑还有很大的提升空间，目前只能考虑一步，之后可以考虑多走几步。游戏结束的返回机制因为精力有限没有完善，menu bar 因为找不到用处也觉得没有必要加。所以本程序还有提升空间。