摘　要

乒乓球是一项老少皆宜、休闲健身的体育活动，在我国深受国民喜欢。

疫情当下，为了减少人员流动以避免疫情的进一步蔓延，各地纷纷响应“静态管理”，防范区居民“足不出户”。在家中，一些乒乓爱好者很难获得与朋友切磋的机会。

本游戏立足于此，复现了 Atari 经典游戏 pong。在此基础上，本游戏支持人机对战和玩家对战。此外，用户可根据自身水平，选择不同难度的人机进行切磋。本游戏使用一个矩形代替乒乓球拍，一个白色小球代表乒乓球。

此外，本游戏支持玩家自定义游戏模式，即玩家可以设置左右球拍的控制权归属玩家还是电脑，此外，还可以设置取胜条件（单局定胜负、三局两胜制、五局三胜制、七局四胜制等）。

本游戏基于 python 的 pygame 模块，采用面向对象技术实现了乒乓球、球拍、按钮等各个元素，通过对背景、球拍和小球等的渲染以及碰撞检测等优化了游戏的功能，是游戏更富可玩性和趣味性。

相较于其他同类游戏，本游戏的游戏模式、游戏体系更完善，也更真实地模拟了现实中可能出现的各种情况，优化了击球算法，能够让玩家身临其境感受乒乓球的魅力。

此外，电脑的控制算法优化也极大程度地提升了用户的游戏体验，能够吸引用户长期游戏。

关键词：乒乓球；python；pygame；击球算法；控制算法

目录

第一章 绪 论

1.1 乒乓球游戏介绍

1.1.1 历史与发展

乒乓球被誉为中国的“国球”，在我国，深受广大人民群众的喜爱。每当漫步公园，走进健身区，都会发现不少人在乒乓球桌前打得火热朝天。

21 世纪以来，随着互联网的迅速发展和电脑的普及，出现了一系列以乒乓球为背景的游戏。Atari 上一款以 pong 命名的游戏就是以乒乓球为背景开发的。一经推出，深受网友的喜爱，掀起了一股“乒乓热”。

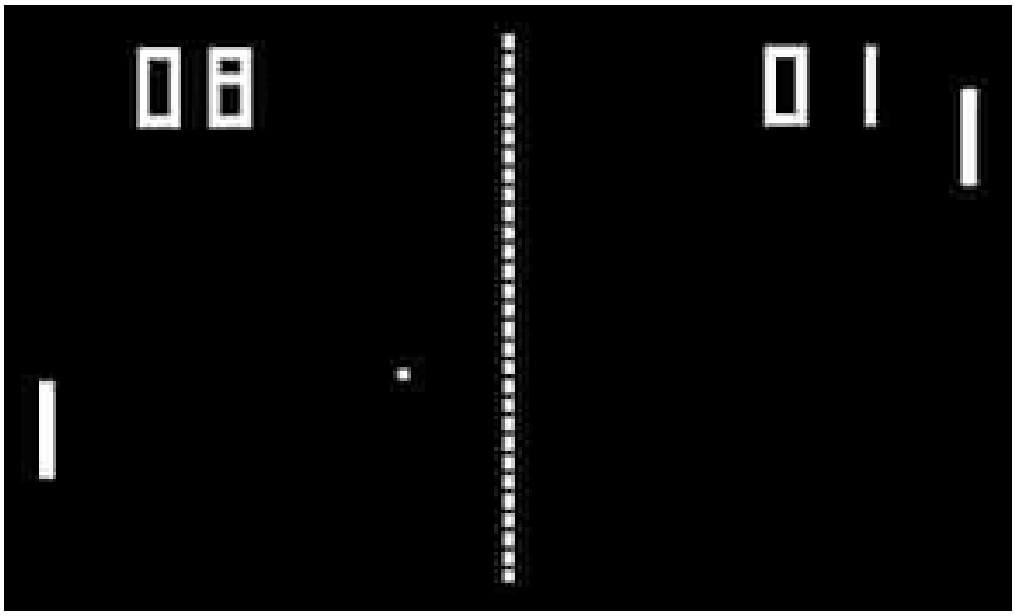


图 1-1 Atari **经典游戏** pong

近年来，随着人工智能的算法的发展，强化学习成为了人们研究的热点之一。作为逻辑较为简单的游戏，pong 很适合作为入门练习，因此这款游戏再度火了起来。

1.1.2 操作方式

玩家通过键盘的方向上键和方向下键操控球拍，实现球拍的上下移动。球拍移动到球的位置可实现击球，球会反弹回去，直到某一方未能成功接球，成功方加 1 分。当某一方先达到 11 分以上并且领先另一方 2 分时小局结束。

1.2 Pygame 模块介绍

Pygame 是一组为编写视频游戏而设计的跨平台 Python 模块。它包括设计用于 Python 编程语言的计算机图形和声音库。

Pygame 最初是由 Pete Shinners 编写的，用于在开发停滞后取代 PySDL 。它自 2000 年以来一直是一个社区项目，并根据自由软件 GNU Lesser General Public License 发布（它“提供 Pygame 与开源和商业软件一起分发”）。

Pygame 使用 Simple DirectMedia Layer (SDL) 库，旨在允许实时计算机游戏开发，而无需 C 编程语言及其衍生语言的低级机制。这是基于这样的假设，即游戏中最昂贵的功能可以从游戏逻辑中抽象出来，从而可以使用 Python 等高级编程语言来构建游戏。

SDL 确实具有的其他功能包括矢量数学、碰撞检测、2D 精灵场景图管理、MIDI 支持、相机、像素阵列操作、转换、过滤、高级自由字体支持和绘图。

使用 Pygame 的应用程序可以通过使用 Pygame Subset for Android (pgs4a) 在 Android 手机和平板电脑上运行。Android 支持声音、振动、键盘和加速度计[1]。

第二章 系统分析

2.1 功能分析

2.1.1 基本功能

1.小球与球拍、边界的碰撞检测及正确反弹。

2.使用键盘方向上键和方向下键对球拍进行上下移动。

3.电脑控制另一个球拍移动，实现人机对战。

2.1.2 拓展功能

1.游戏菜单的制作，可实现开始游戏、设置、退出三个功能。

2.实现自主选择左右两边乒乓球拍的控制权。

3.实现人机难度的控制（简单、中等、困难三个等级）。

4.电脑控制算法的优化。

2.2 类关系分析

整个游戏大致分为 main 函数，paddle 类，ball 类和 button 类。

paddle 类用于创建球拍，游戏中的左右球拍每一个都是一个 paddle 对象。在 paddle 类中，定义了球拍的颜色、移动速度、坐标，同时，定义了球拍移动的成员方法 move 和复原方法 reset。

ball 类用于创建乒乓球。在 ball 类中，定义了乒乓球的颜色、移动速度、坐标，同时，也定义了乒乓球移动的成员方法 move 和复原方法 reset。

button 类主要用于创建菜单按钮。在 button 类中，定义了颜色等一系列按钮的属性值。在成员方法方面，定义了 checkForInput，用于检测鼠标是否进入了按钮的周边区域；同时，与之配套的还有 changeColor，用于改变按钮颜色。两个方法配套使用，从而，达到了鼠标移近按钮区域按钮变色这一功能。

系统结构如下图（图 2-1）所示：

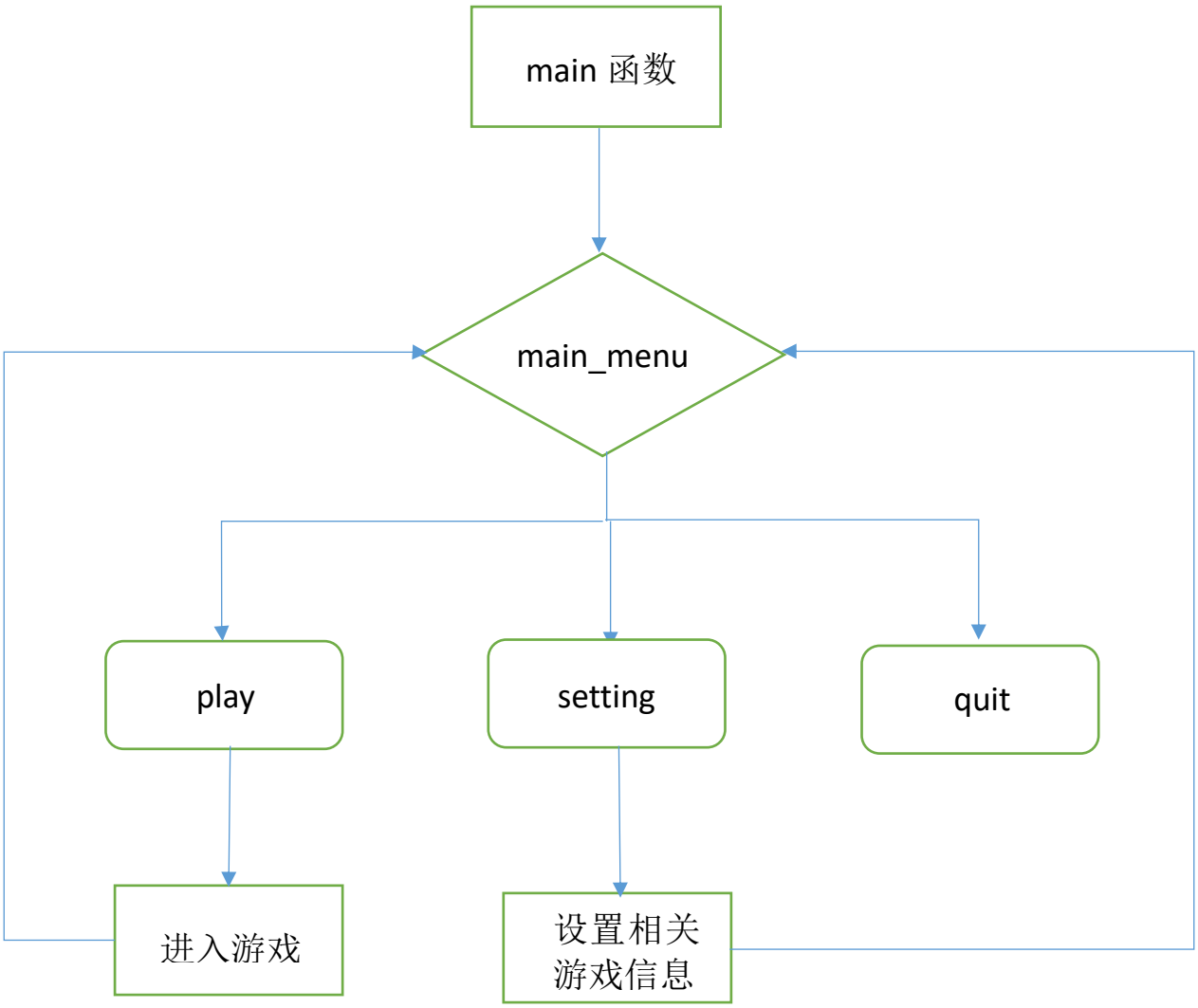


图 2-1 乒乓球游戏程序结构展示

第三章 详细设计及实现

3.1 界面设计

3.1.1 主菜单

主菜单上提供了三个按钮：play，setting 和 quit。分别表示开始游戏、游戏设置和退出游戏。当鼠标聚焦在按钮上时，按钮会改变颜色。

图 3-1 乒乓球游戏主界面

改变颜色功能的实现借助于 button 类里的 checkForInput 方法和 changeColor 方法，其关键逻辑在于检测鼠标位置来决策是否改变颜色。

图 3-2 鼠标聚焦 play 变色效果展示

3.1.2 游戏设置

游戏设置里提供了游戏难度的设置和游戏模式的设置。

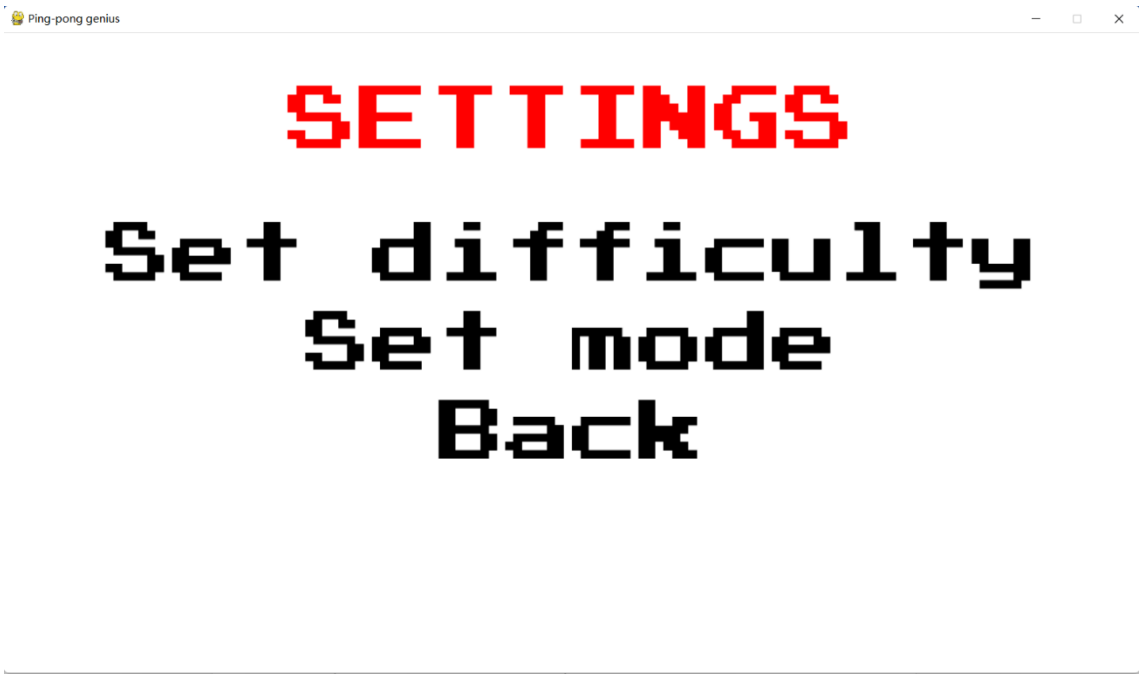


图 3-3 游戏设置界面

在游戏难度的设置中，玩家可自主调节电脑的强弱，包括 easy（简单模式）、middle（中等模式）、hard（困难模式）。

图 3-4 游戏难度设置界面

游戏的难度在 main.py 中用全局变量 level 进行存储。level=1 表示简单模式；level=2 表示中等模式；level=3 困难模式。level 缺省值为 1，在电脑控制板块（后续在 3.2.3 电脑控制逻辑中详细介绍），电脑将根据 level 值的不同采取不同的移动策略。

在游戏模式设置界面，用户可以对竞争模式（可选择左右球拍的控制权）和获胜条件（包括七局四胜、五局三胜、三局两胜和单局定胜负四种模式）进行设置。

图 3-5 游戏模式设置界面

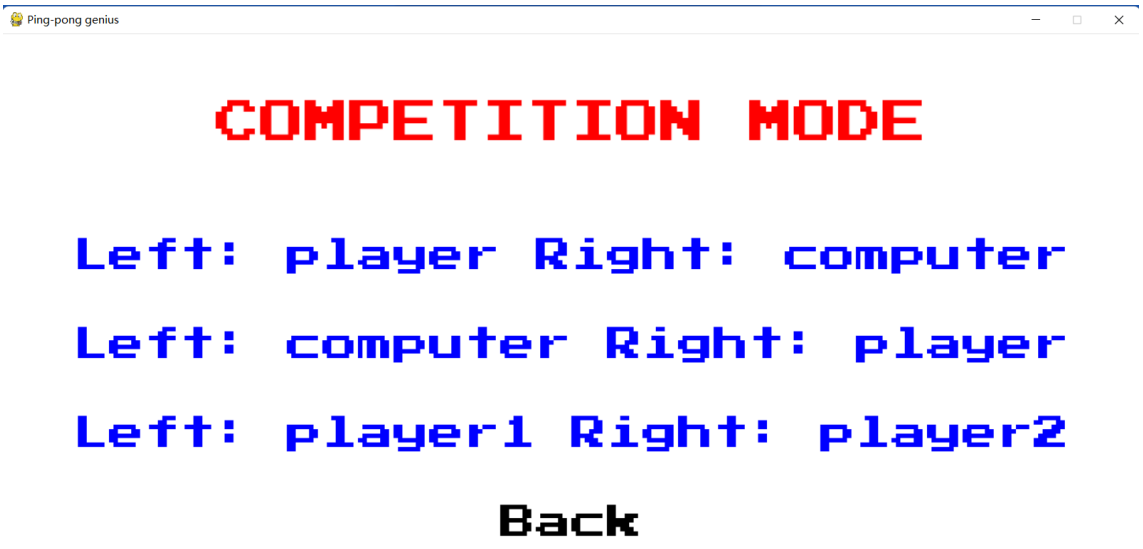


图 3-6 竞争模式设置界面

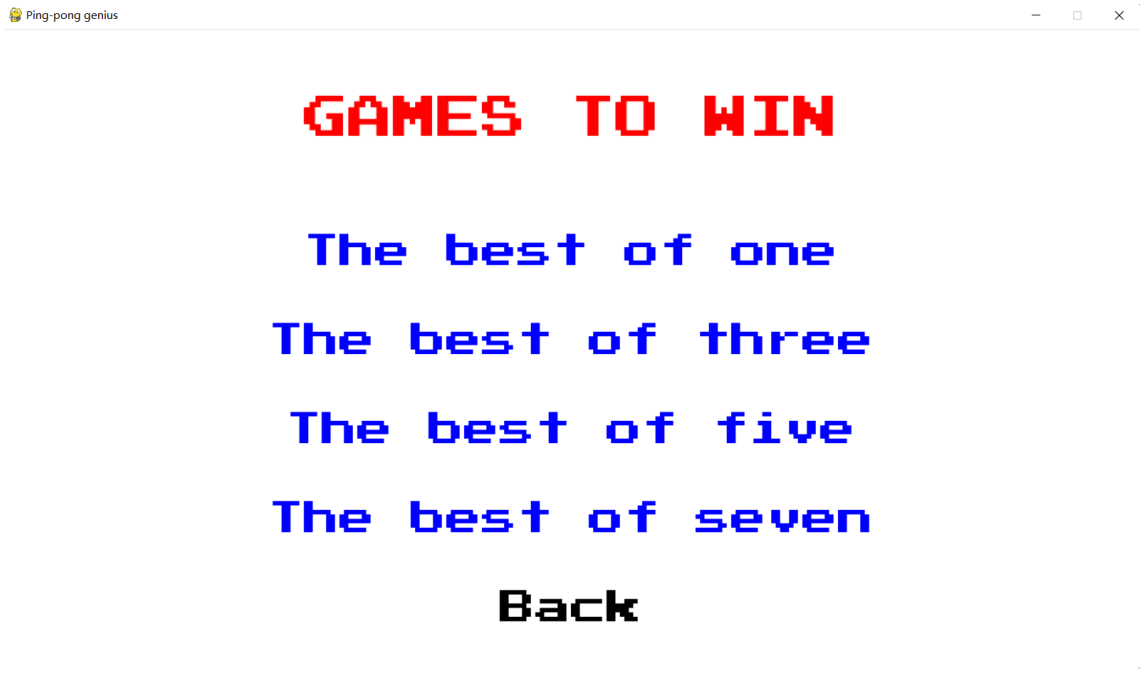


图 3-7 获胜条件设置界面

3.1.3 游戏界面

游戏界面背景选取黑色，在屏幕中心有一条白色虚线表示球网。左右两端的白色矩形表示球拍，玩家可通过方向上键和方向下键对球拍进行移动从而完成击球。

屏幕上方用蓝色小字标识着左右玩家的得分和赢下的小局数，其中 wins 表示赢下的小局数，balls 表示当前小局得分。

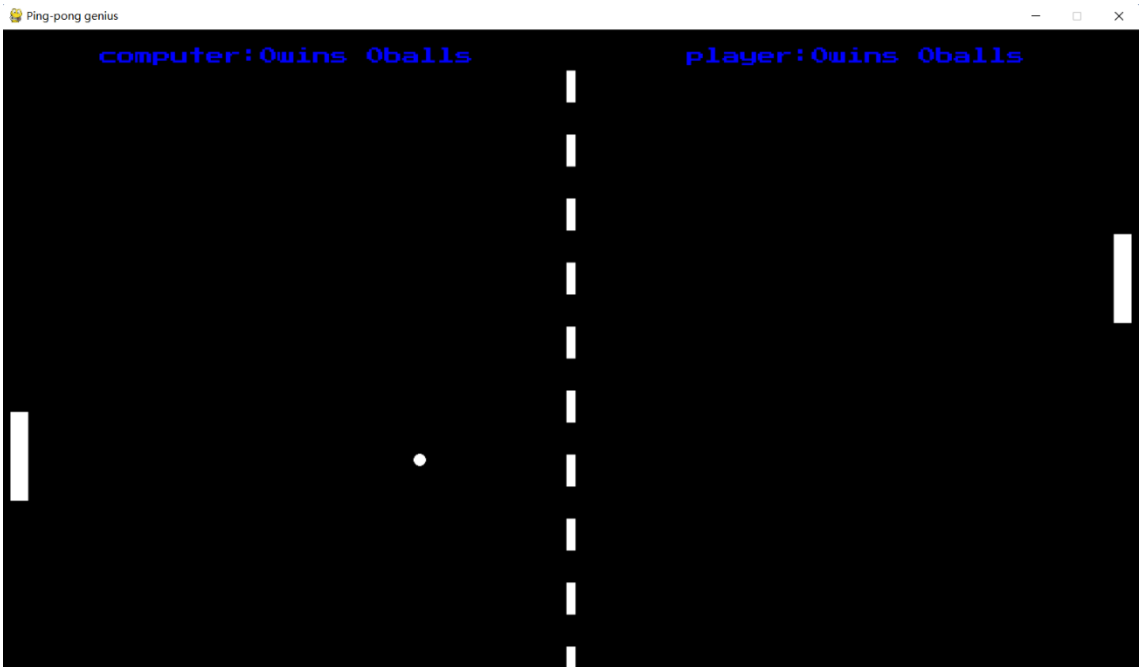


图 3-8 游戏界面展示

小局的获胜条件与当前乒乓球比赛规则一致，即分数第一个达到 11 分及以上且领先对手两球的玩家获胜。

当小局结束时，若整场游戏胜负已分，则界面可供玩家选择返回主菜单或开始新游戏；若胜负未分，下局游戏不会立即开始，会等待玩家选择返回主菜单或继续当前游戏。

图 3-9 胜负已分界面

图 3-10 胜负未分界面

3.2 游戏逻辑

3.2.1 游戏总体规则

本游戏计分和获胜规则与现行乒乓球比赛规则一致，即：当一方未能接住另一方的传球时，对方得 1 分。当双方得分之差大于等于 2 分并且任意一方得分大于等于 11 分时，小局结束。玩家可在游戏设置中设置游戏的获胜条件，如七局四胜、五局三胜、三局两胜、单局定胜负等。当满足获胜条件时，游戏会结束并显示获胜信息。

3.2.2 乒乓球反弹逻辑

游戏中的设定是：当乒乓球触及上下边界时，乒乓球会进行反弹（即 x 方向速度不变，y 方向速度反向）。检测发生碰撞的条件只需要检测乒乓球当前坐标是否处于上下边界即可。整个反弹过程的示意图如图 3-11 所示，红色箭头为碰前乒乓球速度方向，黄色箭头为碰后乒乓球速度方向。可以看出，小球 x 方向速度不变，y 方向速度反向，大小不变。

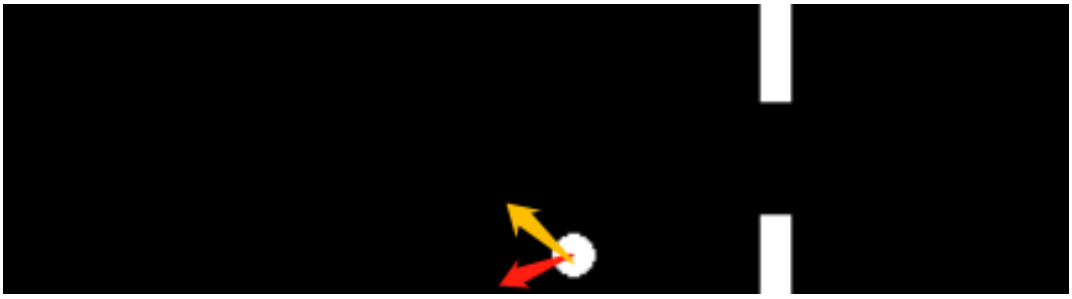


图 3-11 上下边界反弹示意图

对于左右边界，乒乓球不会自动发生反弹，只会在到达边界时检测是否有球拍，若有则进行反弹，若无则判定为未接住球，对手得 1 分。

对于球拍击球逻辑，为了与现实生活中的情况更加接近，本游戏的反弹算法考虑了球拍不同位置击球所带来的影响。

在现实生活中，我们会发现，当乒乓球打在球拍的中心时，乒乓球会近乎垂直于拍面进行反弹；当乒乓球打在球拍的边缘时，乒乓球会几乎沿着竖直方向飞开。不难发现，击球点越接近球拍中部，球反弹后的竖直速度越小；击球点越靠近球拍边缘，球反弹后的竖直速度越大。

因此，本项目的解决方法是：在 ball 类中规定了乒乓球的移动速度，那么球竖直方向分速度一定会在 0 到 v 之间。于是，我们可以计算乒乓球到球拍中心的距离 d_1 和球拍上端到球拍中心的距离 d_2，因为 $d_1 \le d_2$，因此，我们可以利用 d_1 和 d_2 计算出一个 0 到 1 之间的系数 $k = d_1 / d_2$。

那么，最终的竖直方向分速度大小表达式可以写作：

$$v_y = kv_{\max}$$

水平方向分速度大小与原来相等，方向相反即可。如图 3-12 所示：

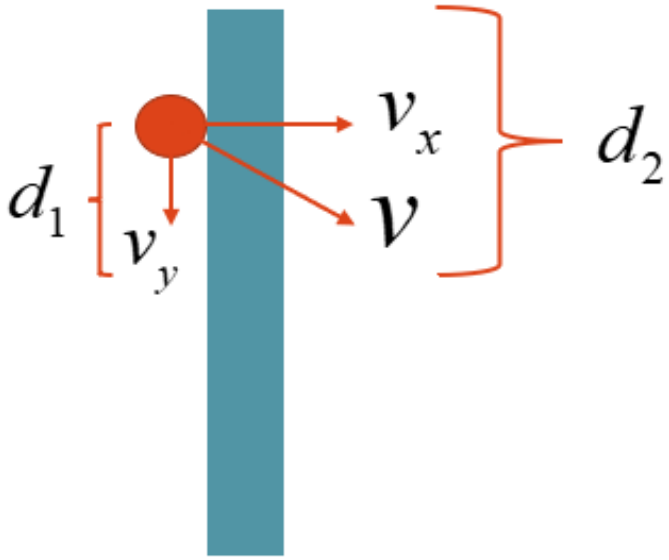


图 3-12 击球算法原理示意图

3.2.3 电脑控制逻辑

本游戏提供了简单、中等和困难三种模式。在实现上，main.py 中定义了一个全局变量 level，level 可取 1、2、3 三个值。当 level=1 时为简单模式；level=2 时为中等模式；level=3 时为困难模式。

在最初设计时，电脑的控制逻辑是：电脑根据乒乓球的坐标判断当前球拍与乒乓球的相对位置关系。若此时乒乓球在球拍上方，则电脑控制球拍向上移动；若此时乒乓球在球拍下方，则电脑控制球拍向上移动。

不过，这样的设计逻辑过于粗暴，会给用户带来不好的游戏体验，因为用

户极难赢球。在之后的设计中，采取了两个措施对这一控制逻辑进行优化：

首先，我们在电脑对球拍进行控制前随机生成 1 个 1-100 的随机数，再根据游戏难度模式的不同选取不同的阈值。若为简单模式，阈值选取为 80；若为中等模式，阈值选取为 90；若为困难模式，阈值则选取为 98。然后，程序将判断生成的随机数是否超过阈值：若超过，则不做任何移动；若不超过，则判断当前球拍与乒乓球的相对位置进行球拍移动。

措施一的实际意义是很形象的，即：在简单模式下，电脑有 80%的概率会做出正确的移动；在中等模式下，电脑有 90%的概率会做出正确的移动；在困难模式下，电脑有 98%的概率会做出正确的移动。

阈值的选取策略是依据小范围玩家测试体验后征集意见的答案。当然，这并不意味着，玩家在面对简单人机的胜率是 20%。因为，这仅仅是单次循环的内部逻辑判断，乒乓球从左边飞到右边可能经过很多个循环，最终球拍的位置是多个循环作用的累加。

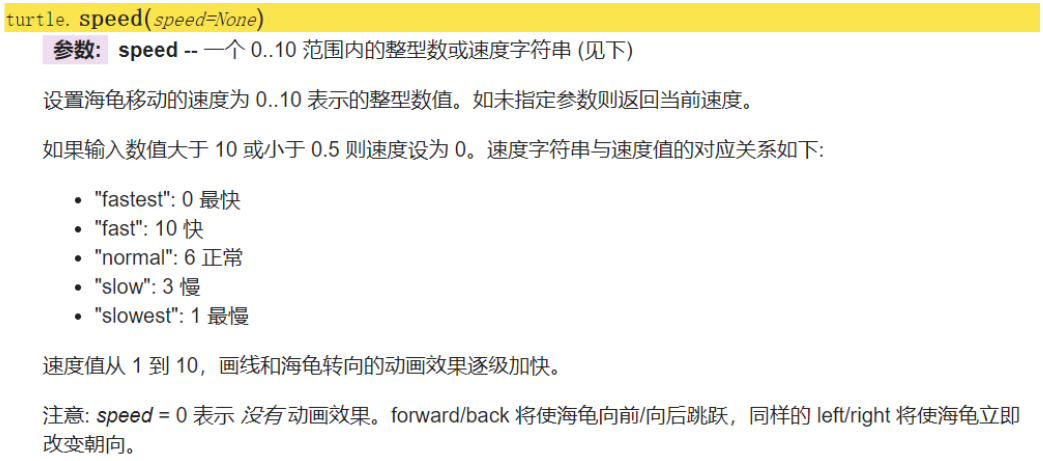
除此之外，3.2.2 所提到的击球算法的优化也能优化用户对战体验。由于球拍移动的速度是一个定值，那么，只要用户击球角度选择得当，打出急转球（即竖直速度接近 max 的球），那么，即使是最高难度的人机也难以击球。

此外，由于上下边界墙壁的存在，也给游戏带来了不确定性，给玩家带来了趣味性。因此，在这两个措施的基础上，用户的游戏体验得到极大的改善。

3.3 其他细节优化

3.3.1 游戏速度控制

在最初的版本中，本游戏是使用 python 的 turtle 模块进行开发的，游戏速度依赖于 speed 方法。但是，speed 方法里参数过少，可提供的速度过少，导致在最初的调试过程中要么速度过慢，玩家无聊度大大提升，要么速度过快，玩家几乎难以做出及时的反应。用户体验感极差。官方文档中的 speed 方法介绍如图 3-13：


turtle. **speed**(*speed=None*)

参数: **speed** -- 一个 0..10 范围内的整型数或速度字符串 (见下)

设置海龟移动的速度为 0..10 表示的整型数值。如未指定参数则返回当前速度。

如果输入数值大于 10 或小于 0.5 则速度设为 0。速度字符串与速度值的对应关系如下:

- "fastest": 0 最快
- "fast": 10 快
- "normal": 6 正常
- "slow": 3 慢
- "slowest": 1 最慢

速度值从 1 到 10，画线和海龟转向的动画效果逐级加快。

注意: *speed* = 0 表示 *没有* 动画效果。forward/back 将使海龟向前/向后跳跃，同样的 left/right 将使海龟立即改变朝向。

图 3-13 官方文档中的 speed 方法

在后来的开发中，本游戏选择了 python 的 pygame 模块。这样，仅需依赖 FPS（屏幕刷新率）即可轻松对游戏速度进行调节。并且，选择性大大提升，最终通过用户测试选择了最佳 FPS60。

3.3.2 实现多页面切换

在项目开发中，菜单界面的实现与切换也是一个比较复杂的问题。pygame 提供的仅有一个窗口，本游戏在实现多页面切换时的逻辑是：在玩家点击按钮，需要进入按钮所对应的界面时，将屏幕以同色填充，覆盖掉之前的画面。在此基础上再进行游戏画面的绘制。

第四章 测试

4.1 游戏应用环境的构建

4.1.1 游戏需要的硬件环境

普通 PC

CPU：1GHz 以上

内存：256MB 以上

分辨率：推荐 1024*768 像素

4.1.2 游戏需要的软件环境

操作系统：Windows 10 及以上

开发环境：Pycharm

需要安装的模块：pygame

4.2 游戏界面显示

图 4-1 显示了游戏的初始画面：

图 4-1 乒乓球游戏主界面

图 4-2 显示了当鼠标聚焦在 PLAY 按钮上的变色效果：

图 4-2 鼠标聚焦 play 变色效果展示

4.3 设置界面显示

图 4-3 显示了设置主界面：

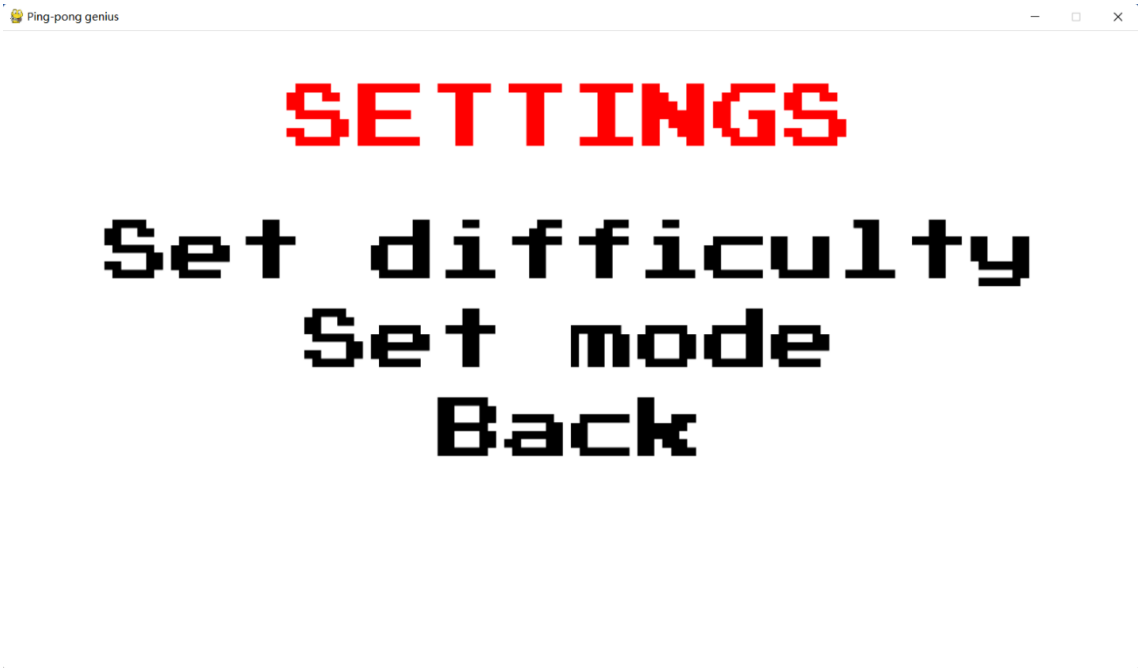


图 4-3 游戏设置界面

图 4-4 显示了难度设置界面：

图 4-4 游戏难度设置界面

图 4-5 显示了游戏模式设置界面：

图 4-5 游戏模式设置界面

图 4-6 展示了竞争模式设置界面：

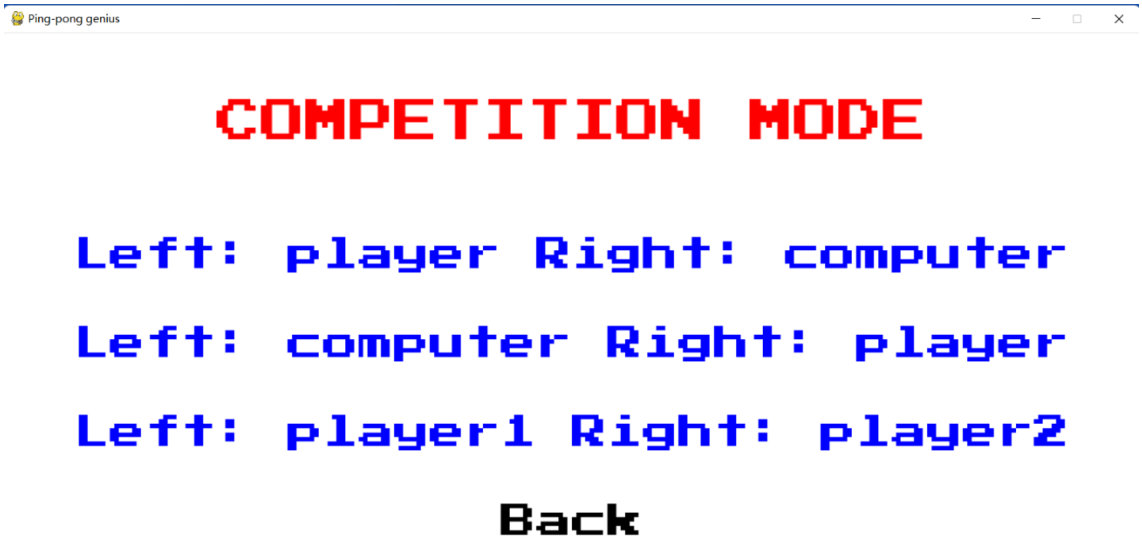


图 4-6 竞争模式设置界面

图 4-7 展示了获胜条件设置界面：

图 4-7 获胜条件设置界面

4.4 游戏中界面显示

图 4-8 展示了游戏中的界面：

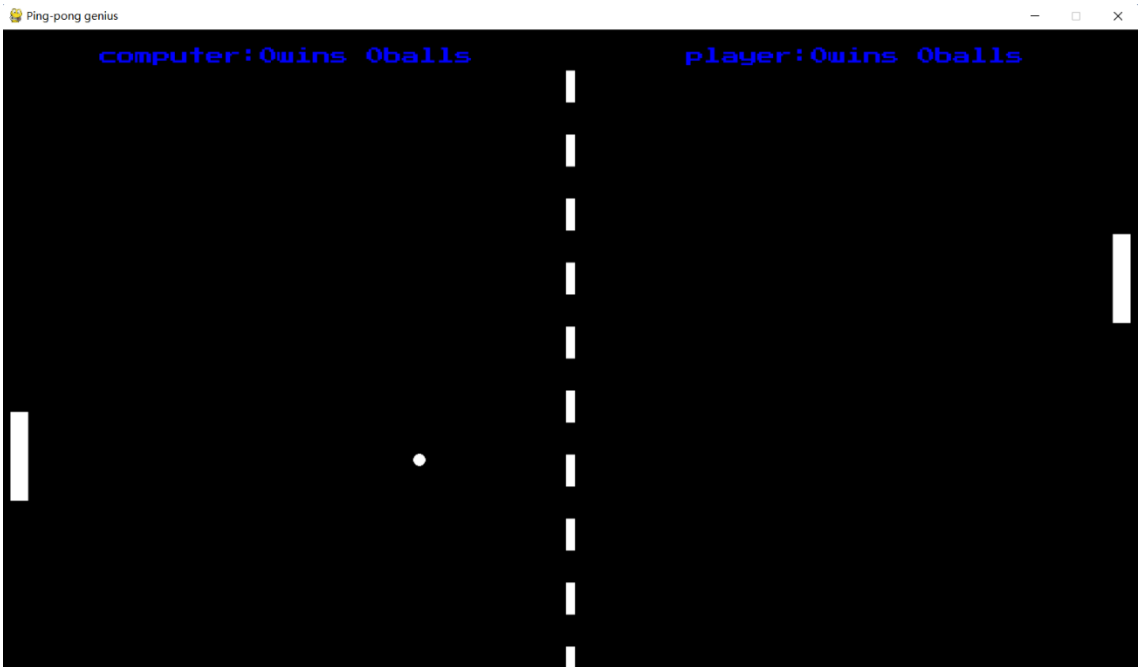


图 4-8 游戏中界面

图 4-9 展示了小局结束（未分胜负）的界面：

图 4-9 小局结束（未分胜负）界面

图 4-10 展示了小局结束（已分胜负）的界面：

图 4-10 小局结束（已分胜负）界面

4.5 小结

本项目完成了既定的设计目标，即实现乒乓球游戏，使得玩家通过操纵乒乓球拍上下移动，与电脑实现人机对战，同时电脑具有了一定的智能。

除此之外，本项目还加入了游戏菜单，支持人机对战、玩家对战两种模式。本游戏还支持玩家对人机难度和获胜条件进行设置。在体验优化方面，本游戏采用了算法优化和最大程度还原真实情况两个策略来提升用户的游戏体验。

当然，本项目还有一定的不足。一是在于电脑控制算法上，此算法控制下的电脑略显单调，趣味性不强。第二，游戏界面还有待优化。

其实本项目的初衷就是使用强化学习进行模型训练，从而完成人机模式的设置。但是经过一段时间学习后，发现绝大多数启蒙教程基于现有的 gym 库，支持的游戏也仅局限于 atari 中的游戏，考虑到学习成本等问题，这一问题将在下一版本中解决。

下一阶段的任务（主要在暑假）准备自学运维相关知识，把本项目进行多端部署。此外，将继续学习强化学习策略，争取在下一版本中人机能用上训练出的

模型。另外，也将继续开展小规模的用户测试，邀请他们为游戏的优化提出宝贵的意见，最大程度满足用户需求。最后，会引入更多模式，增强趣味性，如平衡乒乓球，满足用户的新鲜感。

参考文献

[1] wiki 百科. https://en.wikipedia.org/wiki/Pygame.