

浙江大学

程序设计专题

大程序报告



大程名称: 台球之王 (Billiards King)

小组成员:

1. 姓名 : 管嘉瑞 学号: 3200102557 电话:

2. 姓名 : 杨锐洁 学号: 3180105810 电话:

3. 姓名 : 王舒杨 学号: 3190102452 电话:

指导老师: 钱抱清

2020~2021 春夏学期 2021 年 6 月 15 日

报告撰写注意事项

- 1) 图文并茂。文字通顺，语言流畅，无错别字。
- 2) 书写格式规范，排版良好，内容完整。
- 3) 存在拼凑、剽窃等现象一律认定为抄袭；0分
- 4) 蓝色文字为说明，在最后提交的终稿版本，请删除这些文字。

目 录

1 大程序简介.....	4
1.1 选题背景及意义	4
1.2 目标要求	4
1.3 术语说明	5
2 需求分析.....	5
2.1 业务需求	5
2.2 功能需求	6
2.3 数据需求	8
2.4 性能需求	8
3 程序开发设计	9
3.1 总体架构设计	9
3.2 功能模块设计	9
3.3 数据结构设计	13
3.4 源代码文件组织设计	15
3.5 函数设计描述	18
4 部署运行和使用说明	38
4.1 编译安装	38
4.2 运行测试	41
4.3 使用操作	40
5 团队合作.....	47
5.1 任务分工	47
5.2 开发计划	47
5.3 编码规范	48
5.4 合作总结	46
5.5 收获感言	53
6 参考文献资料	54

《台球之王》大程序设计项目

1 大程序简介

1.1 选题背景及意义

本项目是 2021-2022 春夏学期浙江大学《程序设计专题》课程的课设作业，通过一系列的训练让学生们对 C 语言有更进一步的了解。本项目就是基于课程组提供的 libgraphics 图形库，通过深入学习和对库文件的了解熟悉，完成一个简易的图形界面台球游戏，更好地掌握所学的知识。

本小组选题为：基于 libgraphics，编写一个台球游戏程序。为了能够充分地学习应用 libgraphics、应用丰富的专题知识，同时考虑到台球小游戏的趣味性和挑战性，本小组选择了这个题目。此题目能够充分锻炼程序编写和项目开发能力，帮助掌握专题知识，同时在编写程序的过程中也能够激发小组成员编程的兴趣。

1.2 目标要求

本项目基于 libgraphics 图形库，实现一个简易的台球游戏程序，其功能包括以下几个方面：

- 实现基本的台球规则，提供单人简单模式、双人简单模式、双人九球模式和双人十六球模式，与实际台球规则基本相符
- 实现与鼠标、键盘的交互（控制蓄力击球、自由置球等）
- 实现台球游戏开始、暂停功能；
- 实现台球游戏保存、读取功能，并且支持二进制文件和文本文件的读取和存储。

1.3 术语说明

本程序所用的术语大多为台球游戏中常见的语言。这里对其中一些不易理解的词作一些说明：

- **目标球：**目标球即为玩家要击入洞的球，根据设定的规则。在双人九球模式下，目标球是当前场上编号最小的彩球。每次击球，母球第一个碰到的球必须是目标球。双人十六球模式下，场上球分为白色母球、单色球(1~7号)、双色球(9~15号)和黑色八号球。开始时所有彩球(不包括黑8)都是目标球，当有一名玩家最先击进一颗彩球时(无论是否犯规)，该玩家所击进的彩球类型是其目标彩球花色类型(单色小号/双色大号)，另一种花色类型则为对方目标彩球类型。当双方都没有将自己的目标彩球全部击入洞内时，黑8都不是目标球。直至击进所有自己的目标彩球后，才可以将黑球作为目标球。
- **续杆：**即双人模式下，玩家击打球后若获得续杆即可再次击打一次球。双人简单和双人九球模式下，若一方在未犯规的情况下进彩球，则其可以继续击球；双人十六球模式规则下，若一方在未犯规的情况下击目标彩球进洞(对方目标彩球不算)，则其可以继续击球。
- **自由球：**此时，白球可以自定义在球桌上任意位置，玩家可以通过键盘的 WASD 来控制移动白球的位置。

2 需求分析

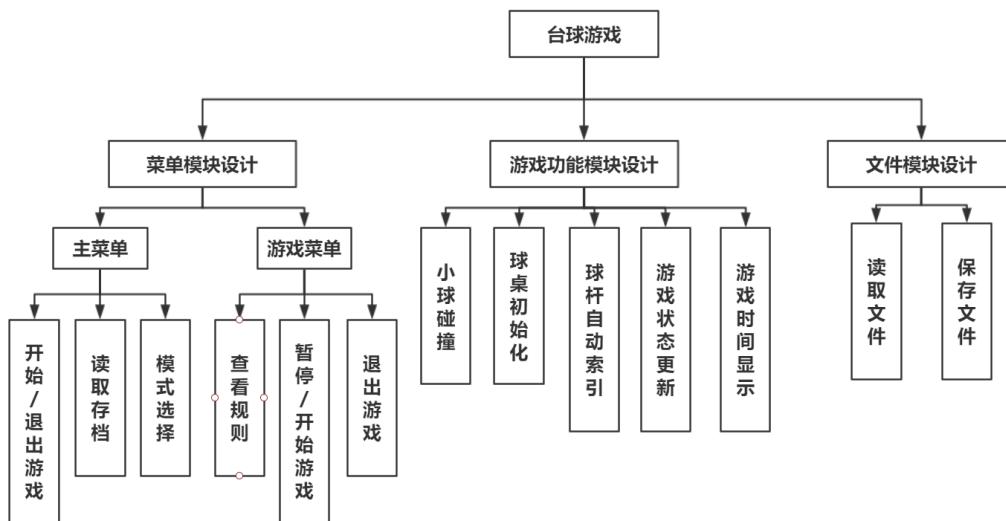
2.1 业务需求

由于现代社会生活节奏的加快，人们可能会感到烦躁无聊，本小组开发的台球小游戏旨在帮助人们缓解工作学习压力，在人们无聊时能够通过小游戏排解烦

恼、打发时间。从而达到帮助人们提高工作学习的效率、身心愉悦的效果。鼠标和键盘的使用也可以帮助活动手指关节，同时双人对战模式能够增加可玩性，提升好友关系。除此之外，因为提供了丰富的游戏模式和规则，本小游戏还可以作为台球初学者的工具，其拥有了相对完善的游戏规则，可以帮助初学者学习台球知识和规则。

2.2 功能需求

本项目的功能主要分为三个模块，分别是菜单模块、游戏模块和文件模块，其详细的功能架构图如下图所示：



本项目的具体功能描述如下：

- 主菜单界面设计

开始游戏：进入新建游戏（选择四种模式中的一种），或读取存档继续游戏

游戏规则：说明各种模式下的游戏规则

玩法说明：说明游戏的玩法，包括如何击球、存档等

关于游戏：说明游戏开发信息（具体内容暂时保密）

退出游戏：关闭图形界面控制台

- 游戏菜单设计

查看规则：查看当前游戏模式下的具体规则

返回主菜单：返回主菜单且不保存游戏进度

退出游戏：退出程序且不保存游戏进度

暂停：暂停游戏（按暂停后按钮变成继续游戏，点击后可以继续）

重新开始：当有人获胜时，游戏暂停，暂停键变成重新开始，点击后可以重新开始一局当前模式的游戏

保存当前进度：保存游戏进度，只有再次保存时才会覆盖原存档

返回：返回上一层菜单且不保存游戏进度

● 游戏界面信息提示

当前击球方（红字）：双人模式下提示击球方

上一轮：提示玩家上一次击球是否犯规、进球情况、连续犯规次数

A 的上一轮：判断 A 是否击中球、是否犯规、是否需要换杆、犯规次数

B 的上一轮：判断 B 是否击中球、是否犯规、是否需要换杆、犯规次数

玩家 A 的进球数：记录玩家 A 进球个数

玩家 B 的进球数：记录玩家 B 进球个数

玩家 A 的目标球：提示玩家击球目标

玩家 B 的目标球：提示玩家击球目标

当前轮次：记录击球轮次（开始时为轮次 1，一次击球轮次数加一）

● 游戏功能实现

游戏界面初始化：包括球桌、球杆、球槽、提示文本、菜单界面、小球的坐标、颜色、编号等初始状态实现

小球碰撞：实现杆击球的碰撞、小球间的碰撞过程、小球和球桌的碰撞

主要应用物理模型和向量运算

击球指示器：为了更好地帮新手适应游戏，我们开发了指示器功能，实现球杆自动索引白球并绘制提示线（提示将要击打的对象、击打对象是否是目标球（白色/红色）和击打后白球、被击球的方向（实线/虚线））

● 文件保存读取：

通过对控制游戏进程的全局变量的分批存取，分别保存游戏进度到二进制文件 binaryfiles.txt、文本文件 txtfiles.txt，以及从二进制文件 binaryfiles.txt、文本文件 txtfiles.txt 中读取游戏进度

2.3 数据需求

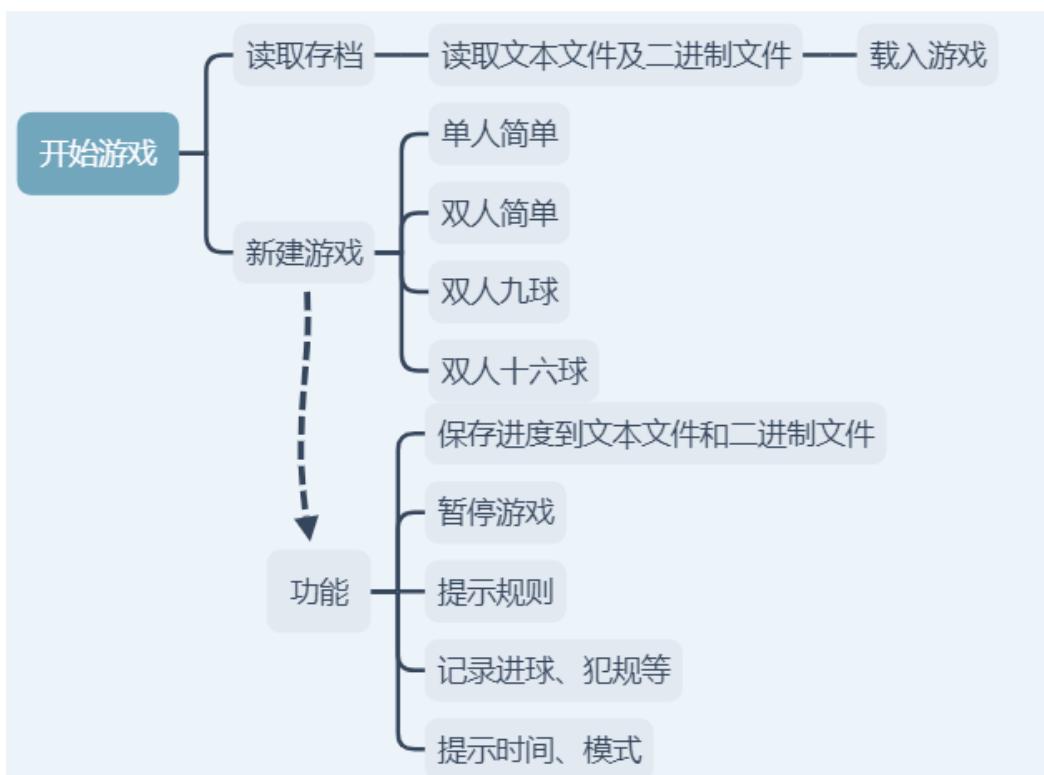
根据代码规范，实现的大程序应该满足这些要求：全局变量建议在模块的初始化函数中初始化，禁止在定义时赋初值；不得出现魔鬼数字（0 除外），应采用有意义的枚举或宏定义常量；判断条件内必须是布尔表达式。另外为了更好地应用所学知识，大程序应该尽量使用更多的专题知识，数据要求上如使用链表、二进制文件读写等。

2.4 性能需求

本程序对性能要求不高，算法主要复杂度在于遍历球判断碰撞，正常的设备都能运行，台球的运动有时可能会有些许卡顿。另外，在某些设备上可能出现尺寸不对、文本重叠的情况。

3 程序开发设计

3.1 总体架构设计



3.2 功能模块设计

为了实现符合要求并具有丰富可玩性的台球游戏，设计出实现程序界面初始化、游戏规则说明、玩法说明、模式选择、游戏窗口信息记录提示、游戏过程保存读取等功能的程序。

3.2.1 程序界面初始化

运行项目后会得到游戏的初始界面。Billiards King 即项目名称；该功能界面下置五个按钮，按钮 1“开始游戏”可以让用户进入游戏选择页面；按钮 2“游戏规则”可以向用户说明不同模式游戏的规则；按钮 3“玩法说明”可以帮助用户快速入门游戏的玩法；按钮 4“关于游戏”则说明了程序开发相关信息，这部分由于匿名

已匿去；按钮 5“退出游戏”即关闭 exe 程序。另外还有两个小球绕着菜单转圈，增添了台球风格。程序初始界面如下：



3.2.2 游戏规则说明

点击主界面的游戏规则按钮即可得到游戏规则丰富的提示说明，四种模式的规则说明都很详细，可以很好地帮助用户熟悉和快速入门这四种模式。另外，在每个模式的游戏界面，我们还设置了“查看规则”按钮，对应着当前模式下游戏规则的说明，这让用户不需要返回主页面查看规则，可以直接在游戏时查看规则，提高了效率。

游戏规则

X

单人简单模式规则：

- 1.单人一直击球，直至所有彩球进洞。
- 2.空杆犯规：如果一次击球后母球直至停止后都没有碰到任何球，则为空杆犯规。
- 3.进白球犯规。

双人简单模式规则：

- 1.AB双方轮流击球，最后击进彩球数量多者获胜。
- 2.空杆犯规：如果一次击球后母球直至停止后都没有碰到任何球，则为空杆犯规。
- 3.进白球犯规。
- 4.续杆：若一方在未犯规的情况下进彩球，则其可以继续击球。
- 5.自由球：若一方犯规，则下一轮另一方可以任意放置母球(键盘WASD控制)，然后再击球。另外第一轮刚开始时先击球方可打自由球。
- 6.在一局中犯规并进彩球，由进的彩球仍算数。
- 7.连续犯规达到五次的一方直接判负（优先级高于进球数量判定）。

双人九球模式规则：

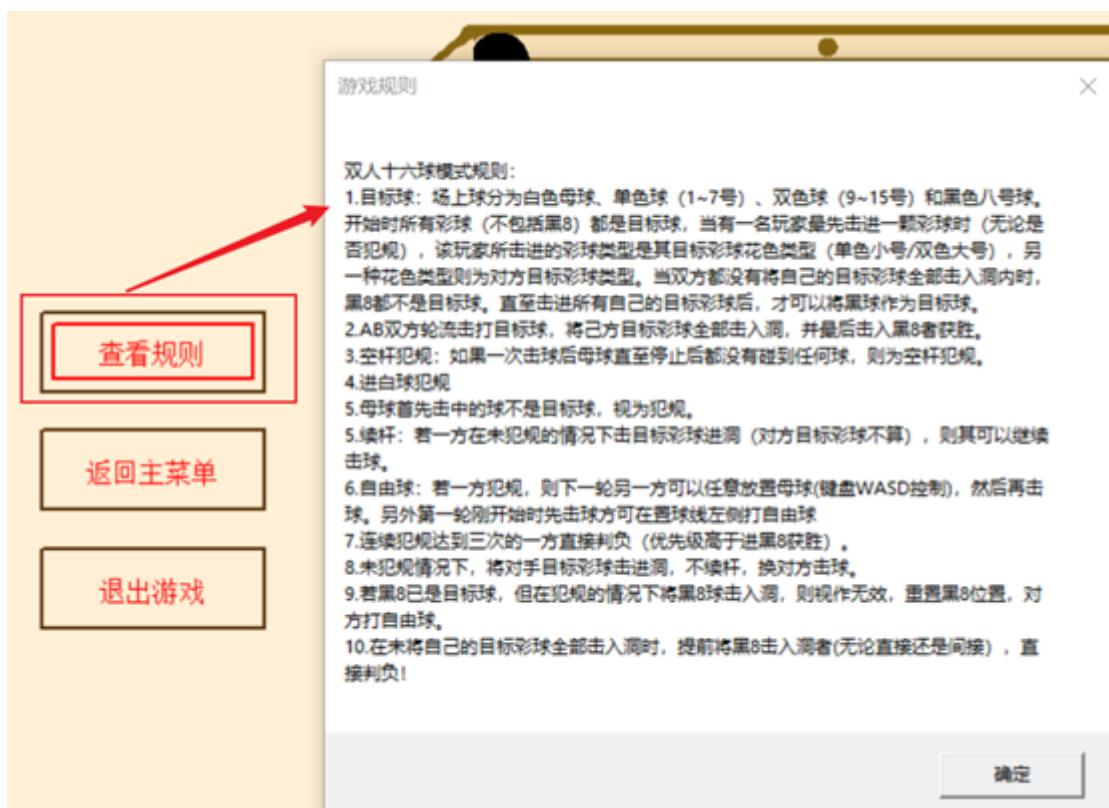
- 1.目标球：目标球是当前场上编号最小的彩球。每次击球，每球第一个碰到的球必须是目标球。
- 2.AB双方轮流击打目标球，最终将9号球击进洞者获胜。
- 3.空杆犯规：如果一次击球后母球直至停止后都没有碰到任何球，则为空杆犯规。
- 4.进白球犯规。
- 5.母球首先击中的球不是目标球，视为犯规。
- 6.续杆：若一方在未犯规的情况下进彩球，则其可以继续击球。
- 7.自由球：若一方犯规，则下一轮另一方可以任意放置母球(键盘WASD控制)，然后再击球。另外第一轮刚开始时先击球方可打自由球。
- 8.连续犯规达到三次的一方直接判负（优先级高于进球数量判定）。
- 9.每次进洞的不必是目标球，无论场上目标球是多少，不犯规进九号球就算获胜。
- 10.若在犯规的情况下将9号球击入洞，则视作无效，重置9号球位置，对方打自由球。

双人十六球模式规则：

- 1.目标球：场上球分为白色母球、单色球（1~7号）、双色球（9~15号）和黑色八号球。开始时所有彩球（不包括黑8）都是目标球，当有一名玩家最先击进一颗彩球时（无论是白球犯规），该玩家所击进的彩球类型是其目标彩球花色类型（单色小号/双色大号）。另一种花色类型则为对方目标彩球类型。当双方都没有将自己的目标彩球全部击入洞内时，黑8都不是目标球，直至击进所有自己的目标彩球后，才可以将黑球作为目标球。
- 2.AB双方轮流击打目标球，将己方目标彩球全部击入洞，并最后击入黑8者获胜。
- 3.空杆犯规：如果一次击球后母球直至停止后都没有碰到任何球，则为空杆犯规。
- 4.进白球犯规。
- 5.母球首先击中的球不是目标球，视为犯规。
- 6.续杆：若一方在未犯规的情况下击目标彩球进洞（对方目标彩球不算），则其可以继续击球。
- 7.自由球：若一方犯规，则下一轮另一方可以任意放置母球(键盘WASD控制)，然后再击球。
- 8.连续犯规达到三次的一方直接判负（优先级高于进黑8获胜）。
- 9.若黑8已是目标球，但在犯规的情况下将黑8球击入洞，则视作无效，重置黑8位置，对方打自由球。
- 10.在未将自己的目标彩球全部击入洞时，提前将黑8击入洞者（无论直接还是间接），直接判负！

确定

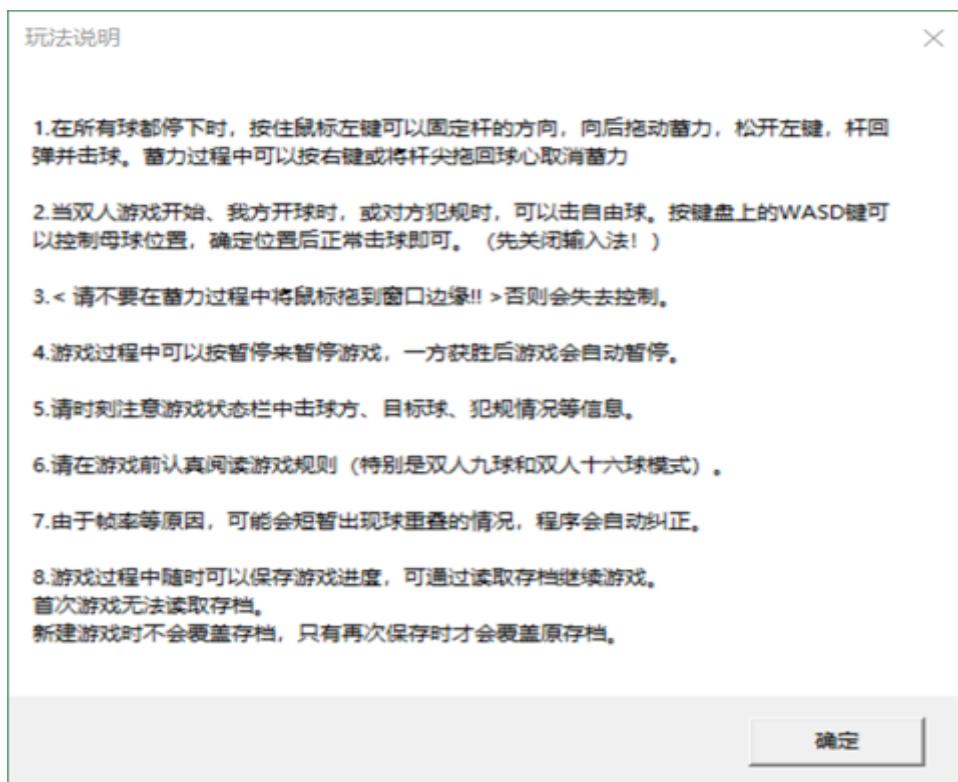
上图为游戏开始界面的规则结束，包含所有模式的详细规则。



上图为游戏进行界面的规则介绍，玩家可以在不同模式下随时点击按钮查看当前模式的具体规则。

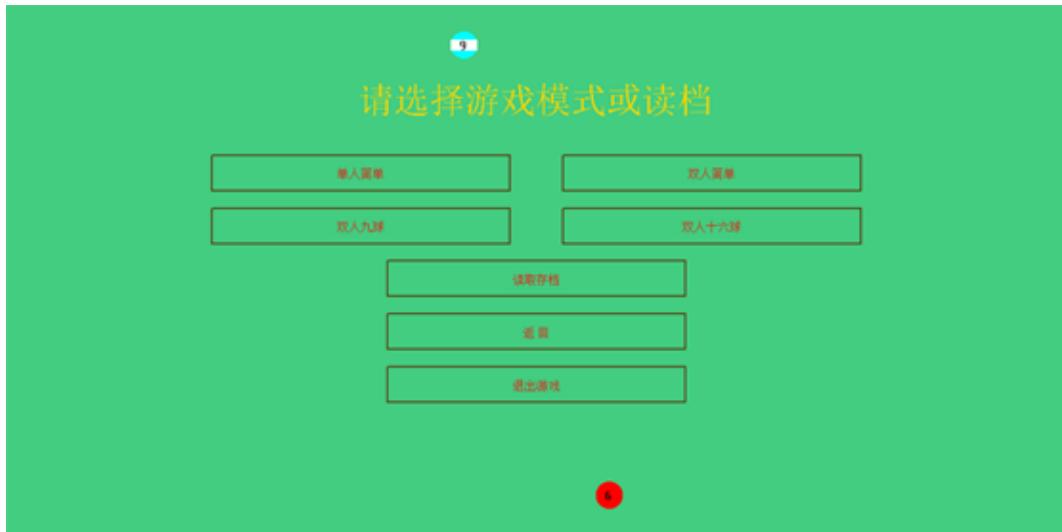
3.2.3 玩法说明

在主界面下，第三个按钮“玩法说明”向玩家说明了击球、双人游戏键盘控制、蓄力注意事项、暂停功能说明、信息提示功能介绍、游戏存档读档说明等事项。另外在进入游戏界面后点击左上方 About 按钮点击 Usage 也可以查看玩法说明。



3.2.4 模式选择

在主界面点击开始游戏按钮后，程序会进入一个选择游戏模式或选择读档的界面，在这个界面，玩家可以选择单人简单模式、双人简单模式、双人九球模式、双人十六球模式，点击按钮即可进入对应模式进行游戏。



3.2.5 游戏窗口信息记录提示

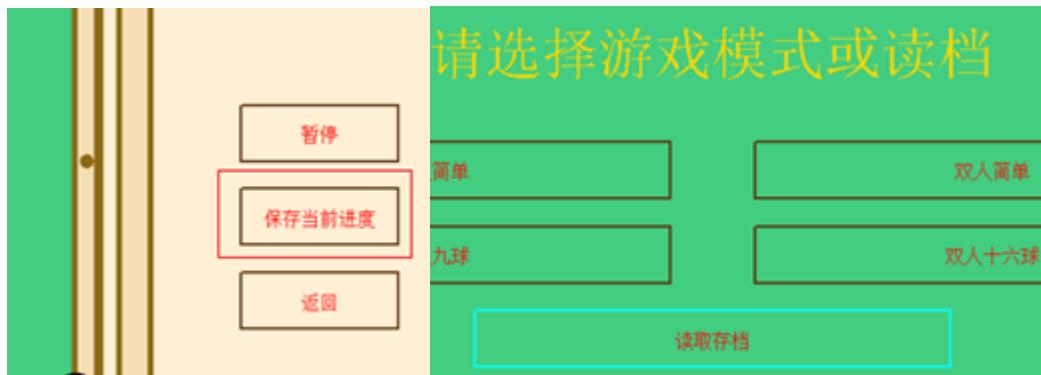
从模式选择或读取存档进入游戏界面后，游戏界面会有许多信息提示：游戏界面有“查看规则”的按钮，点击后可查看当前模式的规则；在球桌上方有击球方、上一轮进球和犯规信息、轮次信息、玩家目标球信息，在程序窗口底部还有游戏时间、当前模式的信息提示。



3.2.6 游戏过程保存读取

在任何模式下进行游戏后，可以在游戏界面右侧看到“保存当前进度”按钮，点击即可将当前游戏进度保存到二进制文件和文本文件。当重新进入程序后点击

开始游戏按钮进入游戏模式选择或读档界面，点击读取存档即可进入上一次保存的游戏进度和相关信息。



3.3 数据结构设计

本程序主要使用了数组和链表进行数据组织，程序根据数据自身特点选择组织形式。例如，球槽使用链表保存，而台球不使用链表存储，因为碰撞等操作经常需要随机读取，不太适合链表存储，而球槽则时刻需要记录台球的进洞和出洞重置，适合链表的储存形式。除此之外，程序将模式、计时器、轮次结果等有限数据定义为枚举类型，将球、杆、玩家上一轮的结果_犯规次数等存储为结构体类型。

在 Main 函数中，本程序进行了窗口信息、全局控制变量初始化，注册了时间响应函数、键盘回调函数、鼠标回调函数，利用定时器调用 display 函数实现画面的刷新。在 compute_judge.c 中定义了许多数学计算的函数，程序主要利用了数学方法如勾股定理等数学公式进行计算和判断。在 control.c 中，定义了一些控制函数，如回合控制函数，程序利用了一些全局变量来判定回合的开始和结束；利用布尔表达式来作为判定条件。在 draw.c 中，程序通过调用 graphics 库函数或者自定义一些库中缺少的函数来绘制图形，其中通过根据一些全局变量、利用向量运算等来进行绘制，利用链表知识存储进球信息等。

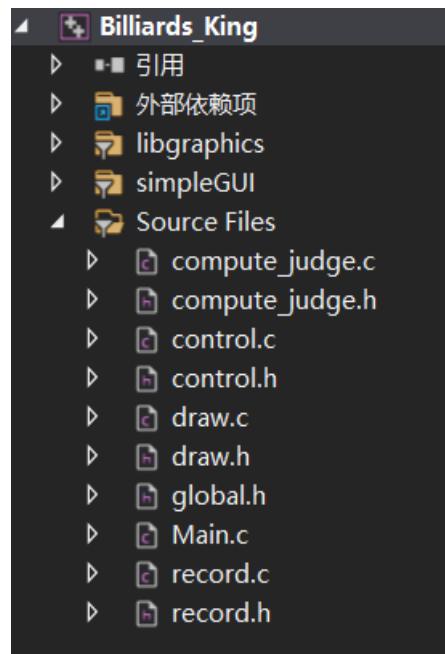
本程序主要数据存储对象是一些决定游戏进程的全局变量，按文件存储类型主要分为二进制存储和文本文件存储，分别存储在 binaryfiles.txt 和 txtfiles.txt 中。

这一部分函数主要在 record.c 中，在存档时，程序定义文件指针，先调用 delete_data 函数将已有存档删除，使用 fopen 函数新建/打开 txtfiles.txt 文件后，使用函数 fprintf 将 int、double、bool 型变量写入文本文件中，在读档时使用 fscanf 函数从文本文件中读取这些变量，最后使用 fclose 函数关闭文件；使用 fopen 函数新建/打开 binaryfiles.txt 文件用于存放二进制文件，使用函数 fwrite 将数组、链表等数据二进制写入文件，在读档时使用 fread 函数从二进制文件 binaryfiles.txt 中读取数据，最后使用 fclose 函数关闭文件。

3.4 源代码文件组织设计

3.4.1 文件函数结构

3.4.1.1 文件结构：



文件结构如上图，包含了 libgraphics 和 simpleGUI 库，源文件中包含了 10 个文件，包括：

- 主函数 Main.c
- 定义结构体、枚举类型、声明全局变量的 global.h

- 实现基本向量运算和逻辑判断的 compute_judge.h 和 compute_judge.c
- 实现绘制、显示功能的 draw.h 和 draw.c
- 实现游戏进程控制的 control.h 和 control.c
- 实现存档、读档的 record.h 和 record.c

3.4.1.2 函数结构:

主函数: 定义了 Main() 函数 (还定义了控制游戏进程的全局变量)

global.h: 未声明函数 (只起声明全局变量、定义结构体)

compute_judge.c: 定义了如下 11 个函数, 主要实现计算和判断功能

compute_judge.h: 声明了如下 11 个函数

```
compute_judge.c
└── b_s_h_distance(ball)
└── b_s_v_distance(ball)
└── bb_distance(ball, ball)
└── get_min_ball()
└── getcos(coordinate)
└── getmod(coordinate)
└── getsin(coordinate)
└── is_allball_still(int)
└── is_in_hole(ball)
└── is_overlap(int, coordinate)
└── is_target(int, int)
```

draw.c: 定义了如下 17 个函数, 以及定义关于规则和玩法的静态字符串

draw.h: 声明了这 17 个函数

```

draw.c
├── about_text
└── define_my_colors()
    └── display(int)
        └── display_game()
            └── draw_background()
                └── draw_balls()
                    └── draw_dot_line(double, double)
                        └── draw_groove()
                            └── draw_indicator()
                                └── draw_one_ball(ball)
                                    └── draw_stick()
                                        └── draw_table()
                                            └── draw_text()
                                                └── draw_win_text(int)
                                                    └── drawBegin()
                                                        └── DrawCircle(coordinate, double, string)
                                                            └── drawMenu()
                                                                └── drawNewGame()
                                                                    └── rule_text
                                                                        └── usage_text

```

control.c: 定义了如下 13 个函数，主要用于控制游戏进程

control.h: 声明了这 13 个函数

```

control.c
└── balls_hit(ball *, ball *)
    └── correct_pos()
        └── delete_list(List)
            └── edge_hit(ball *)
                └── event_control(EVENT)
                    └── if_win()
                        └── Init_game_data()
                            └── move_ball()
                                └── myKeyboardEvent(int, int)
                                    └── myMouseEvent(int, int, int, int)
                                        └── set_balls()
                                            └── turn_control(int)
                                                └── win_and_pause()

```

record.c: 定义了如下三个函数，实现文件的保存、读取、删除功能

record.h: 声明了这三个函数

```

record.c
└── delete_data()
    └── read_data()
        └── save_data()

```

3.4.2 多文件构成机制

3.4.2.1 文件构成

Main.c: 包含了外部库文件和 5 个源文件中的头文件： global.h 、 compute_judge.h 、 draw.h 、 control.h 、 record.h 。

global.h: 包含外部库文件（ libgraphics 等）

compute_judge.h: 包含了 global.h

control.h: 包含了 global.h 、 compute_judge.h

draw.h: 包含了 control.h 、 global.h

record.h: 包含了 global.h

除 Main 外的另外 5 个 .c 文件：包含对应的 .h 头文件

3.4.2.2 头文件保护

每个头文件都采用了预处理的方法防止重定义。

#ifndef 文件名

#define 文件名

#endif

```
#ifndef _GLOBAL_H
#define _GLOBAL_H
```

3.4.2.3 外部变量

在 global 中用 extern 关键字声明了 Main 中定义的控制游戏进程的全局变量（也是文件读写的对象）。这些全局变量在函数中起到了至关重要的作用。（因此该程序全局变量较多，是项目的一个缺点）

3.5 函数设计描述

<此部分内容较多，共 45 个函数，其中重点函数（划线）是 3.5.2 中的 3 、 4 、 7 、 9 以及 3.5.3 中的 8 和 9 >

3.5.0: Main 函数：主函数

函数原型: Main()

功能描述; 作为主函数, 是程序的开始。定义了控制游戏进程的所有全局变量, 初始化窗口, 设置窗口基本信息, 自定义颜色, 初始化游戏数据(全局变量), 注册响应函数, 开启定时器调用事件回调函数。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 无特别算法

3.5.1: compute_judge.c: 计算、判断函数

3.5.1.1:

函数原型: double getmod(coordinate vec)

功能描述; 获取二元向量 vec 的模长

参数描述: vec 是一个二元坐标, 可以一个向量

返回值描述: 返回模长

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 直接返回两坐标平方和的根号即可

3.5.1.2:

函数原型: double getcos(coordinate vec)

功能描述; 返回向量 vec 与水平向右方向夹角(逆时针)的余弦值

参数描述: vec 是向量

返回值描述: 夹角余弦值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 利用数学公式即可

3.5.1.3

函数原型: double getcsin(coordinate vec)

功能描述; 返回向量 vec 与水平向右方向夹角(逆时针)的正弦值

参数描述: vec 是向量

返回值描述: 夹角余弦值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 利用数学公式即可

3.5.1.4:

函数原型: double b_s_v_distance(ball ball_to_hit);

功能描述; 求出自球沿杆方向与待击球之间的垂直距离, 画指示器时要用

参数描述: ball_to_git 是当前击球方向下, 将击到的第一颗球

返回值描述: 返回待击球与杆的垂直距离

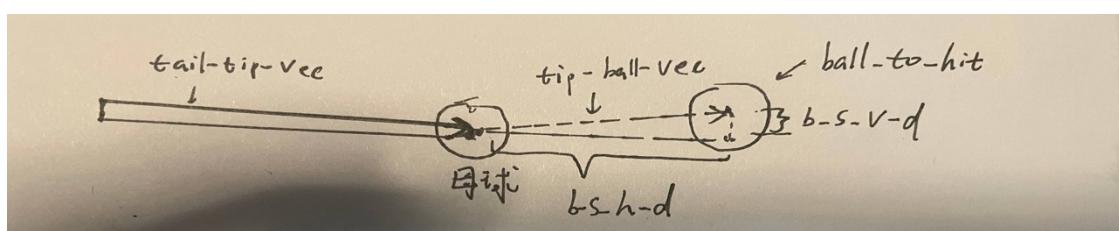
重要局部变量定义: coordinate tail_tip_vec: 杆尾到杆尖的向量,
coordinate tip_ball_vec 杆尖到待击球球心的向量,

double dot_product: 上面这两个向量的点积

proj_vec: tip_ball_vec 在 tail_tip_vec 方向上的投影向量

重要局部变量用途描述: 为了利用勾股定理求出垂直距离做铺垫

函数算法描述: 利用勾股定理(见下图)



3.5.1.5:

函数原型: double b_s_h_distance(ball ball_to_hit)

功能描述; 求出自球沿杆方向与待击球之间的水平距离, 画指示器时要用

参数描述: 同上

返回值描述: 返回待击球与杆的水平距离

重要局部变量定义： tail_tip_vec, tip_ball_vec 定义同上，

另外定义： tail_tip_unit_vec 表示杆方向的单位向量

重要局部变量用途描述： 利用数学原理求出水平距离

函数算法描述： 用 tail_tip_unit_vec 与 tip_ball_vec 点乘即可

3. 5. 1. 6:

函数原型： double bb_distance(ball b1, ball b2);

功能描述： 计算球心距离，以便判定碰撞

参数描述： b1、b2 是两个球

返回值描述： 返回这两个球球心的距离

重要局部变量定义： 无

重要局部变量用途描述： 无

函数算法描述： 利用距离公式

3. 5. 1. 7:

函数原型： bool is_allball_still(int start);

功能描述： 从第 start 颗球开始，判断是否所有往后的球都停止运动了

参数描述： start 是判定起点，一般只取 0 或 1，取 1 代表不判断母球（白球）是否停止运动，用于母球进洞等特殊情况下

返回值描述： 全部静止返回 TRUE，否则返回 FALSE

重要局部变量定义： 无

重要局部变量用途描述： 无

函数算法描述： 遍历球数组，遇到非静止球直接返回 FALSE

3. 5. 1. 8:

函数原型： bool is_in_hole(ball b);

功能描述： 判断球 b 是否进洞

参数描述： b 是要判定的台球

返回值描述： 进洞返回 TRUE，否则返回 FALSE

重要局部变量定义： double in_range。

重要局部变量用途描述：：当球和洞的间距小鱼 in_range 时，球才进洞。其值取决于宏定义的 ACCURACY_NEED(决定游戏难度) R*HOLE_BALL_RATIO 是洞口的半径，当 ACCURACY_NEED 取 1 时，球洞相切就可以进洞，实际取 0.8 左右比较合理。

```
double in_range = ACCURACY_NEED * (R + R * HOLE_BALL_RATIO);
```

函数算法描述：将球逐个与 6 个洞判定距离是否小于 in_range，是则返回 TRUE

3.5.1.9:

函数原型：bool is_overlap(int id, coordinate pos)

功能描述； 判定是否能以 pos 为圆心放 id 号球，而不与球、洞、墙重叠。
主要在放置自由球时调用，放置鼠标控制的自由球随意移动发生重叠。

参数描述： id 要放的球（通常是 0，母球为自由球）。pos：要放置的位置

返回值描述： 能放置返回 TRUE，否则返回 FALSE

重要局部变量定义： ball ball_temp

重要局部变量用途描述：ball_temp 是虚拟的暂时球，用于作为 bb_distance 快速分析是否会重叠

函数算法描述：计算球与其他球、洞、墙的距离

3.5.1.10:

函数原型：bool is_target(int m_turn, int id)

功能描述； 判断如果当前轮次为 m_turn (-1 代表 A, 1 代表 B)，id 号球是否击球者的目标球

参数描述： m_turn 表示轮次，id 表示要判断的球的编号。

返回值描述：是返回 TRUE，不是返回 FALSE

重要局部变量定义： 无

重要局部变量用途描述：无

函数算法描述：根据模式，单人简单、双人简单恒返回 TRUE，双人九球返回是否是最小编号的球。双人十六球模式，根据全局变量 g_target_A,

g_target_B 判断是否是目标球；

3.5.1.11:

函数原型: int get_min_ball()

功能描述; 返回在场的编号最小的球（从 1 号球开始），用于双人九球模式下判断目标球

参数描述: 无

返回值描述: 返回最小球编号，若场上无球（母球除外），返回-1

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 遍历，判断 present 是否为 1。

3.5.2: control.c 中的函数

3.5.2.1:

函数原型: void set_balls();

功能描述: 根据模式，将球放在初始位置上

参数描述: 无

返回值描述: 无

重要局部变量定义: double r

重要局部变量用途描述: R 的某一个倍数（如 1.25），为初始放置时，相邻球心之间间距的一半（虚拟半径）。放置时并未让球相切，二是留出一定距离，防止一瞬间碰撞太过频繁引起 bug

函数算法描述: 根据模式，初始化全部变量：球数组 g_balls。present 全部设为 1（九球模式下 10~15 号球除外）。just_hit 设为-5，代表未与任何单位碰撞

3.5.2.2:

函数原型: void Init_game_data();

功能描述: 重置所有游戏中的数据，主要在新建游戏时调用

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：先调用 `set_balls()` 初始化球的位置，然后初始化各个控制游戏进程全局变量。

3.5.2.3:

函数原型：`void event_control(EVENT eve);`

功能描述：这个函数不是一直调用的，而是在 `turn_control` 中有条件地调用，即只在一个轮次结束后调用（更具体来说，是一方击球，到所有球都停止，算是一个轮次结束）。函数功能主要是：给轮次数加一、纠正重叠球的位置、根据回合事件更新玩家的上一轮击球结果和连续犯规次数、重置提前击进的球（并在球槽链表中删除）、调用 `if_win()`，判断是否有人获胜。

参数描述：`eve` 是 `turn_control` 中传进来的回合事件，即一方击球后的结果。

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述无：

函数算法描述：根据传进的事件更新玩家的上一轮结果。对于提前击进的球（比如击进母球、双人九球模式下犯规击进 9 号球），重置球的位置，并将其从球槽链表中剔除（链表的节点删除）。

3.5.2.4:

函数原型：`void turn_control(int matter);`（最复杂的一个函数）

功能描述：回合控制函数：接口众多，不断被 `display_game()` 调用，另外 `moveball` 中进球也会调用。调用时传参不同，用于输入事件。

参数描述：`matter`：传入的事件，用宏定义表示。

```
// 击球状况
#define MAT_NULL 0 // 无
#define MAT_COLOREDBALLIN 1 // 进彩球
#define MAT_NOTARGETBALL 2 // 击中非目标球
#define MAT_TARGETBALL 3 // 击中目标球
#define MAT_WHITEBALLIN 4 // 白球进洞
#define MAT_BLACKBALLIN 5 // 黑8提前进洞（十六球模式下）
```

其中 0(MAT_NULL) 不断被 display_game 调用传参, 1、2、3、4、5 被 move_ball 中的进球判定调用传参(只有事件发生时才会传参)。

返回值描述： 无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：此函数算法最为复杂，将在下面详细介绍。

```

if (g_prev_if_all_ball_still == 0 && is_allball_still(0) == 1) // 开始新的回合
{
    if (RES_BLACKBALLIN == g_result) // 十六球模式下黑8提前进洞
        event_control(BLACK_8_IN_ADVANCE);
    if (RES_MISS == g_result) // 空杆
        event_control(EMPTY);
    else if (RES_WHITEBALLIN == g_result) // 进白球
        event_control(WHITE_IN);
    else if (RES_NOTHIT == g_result || RES_HITNOT == g_result) // 没进白球，但没打到目标球
        event_control(TARGET_MISS);
    else if (RES_HITANDIN == g_result) // 没进白球，击打目标球并进了彩球
        event_control(BALL_IN);
    else if (RES_HITANDOUT == g_result) // 击打目标球但没进球
        event_control(NO_BALL_IN);
    g_result = RES_MISS; // result回归原始状态
}
g_prev_if_all_ball_still = is_allball_still(0); // 此语句可以原来结合上面的if语句判定一个击球轮次何时开始或结束

```

首先是回合结束的判定：利用全局变量 `g_prev_if_all_ball_still` 和函数 `is_all_ball_still` 来判定回合开始与结束。一个回合开始前，两者均为 1，刚开始时，为 $(1, 0)$ ，击球后，两者一直为 $(1, 1)$ ，球全部停止后（回合结束）两者一瞬间为 $(0, 1)$ ，而后马上变为 $(0, 0)$ 。这个判断语句就是抓住了这个 $(0, 1)$ 的时机，实现回合结束的判定。回合结束后马上根据 `result` 给 `event_control` 传参。

另外，说明一下 result 的更新：result 有以下几种：

```
// 击球结果，用于控制回合和犯规判定
#define RES_HITANDOUT 1 // 击球成功，球未进
#define RES_HITANDIN 2 // 击球成功，球进
#define RES_HITORNOT 3 // 击球存疑，需进一步判断
#define RES_NOHIT 4 // 击球失败
#define RES_WHITEBALLIN 5 // 白球进洞
#define RES_MISS 6 // 空杆
#define RES_BLACKBALLIN 7 // 提前进黑球（十六球模式下）
```

这些事件有优先级，数字越大的优先级越高，根据传进来的 matter 实时更新，初始的 g_result 是 RES_MISS，如果传进击球的事件，根据击球类型调整为对应的情况。比如如果十六球模式下提前进黑 8，直接升级为 RES_BLACKBALLIN，不再改。进白球的优先级也很高。具体的优先级体现在持续中用了一个只进依次的 while() 来实现，这一部分写的有点生硬。

3. 5. 2. 5:

函数原型: void if_win();

功能描述: 判断回合结束时是否有人获胜，若有，调用 win_and_pause()，只在 event_control 最后调用

参数描述: 无

返回值描述: 无

重要局部变量定义: bool all_in

重要局部变量用途描述: 简单模式下判定是否所有彩球都已进洞

函数算法描述: 根据模式，简单模式下判端是否所有球都进洞即可。双人九球模式，判断是否在不犯规的情况下击入了九号球。双人十六球模式下，判断是否某一方目标球是黑 8 且已经不犯规地击进球。另外只要不是单人简单模式，如果一方犯规次数（全局变量中储存）达到阈值，直接判负。

3. 5. 2. 6:

函数原型: void win_and_pause();

功能描述: 获胜，暂停游戏（中止游戏定时器 TIMER_GAME）。同时修改全局变量

参数描述: 无

返回值描述: 无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：调用 cancelTimer。修改表示是否暂停的全局变量 g_is_pause 为 TRUE，修改表示是否有人获胜的全局变量 g_is_win 为 TRUE。

3.5.2.7:

函数原型：void balls_hit(ball* b1, ball* b2);

功能描述：模拟两球的碰撞过程，交换对心方向的速度，乘以衰减系数，给 turn_control 传进碰撞对应参数，更新 just_hit

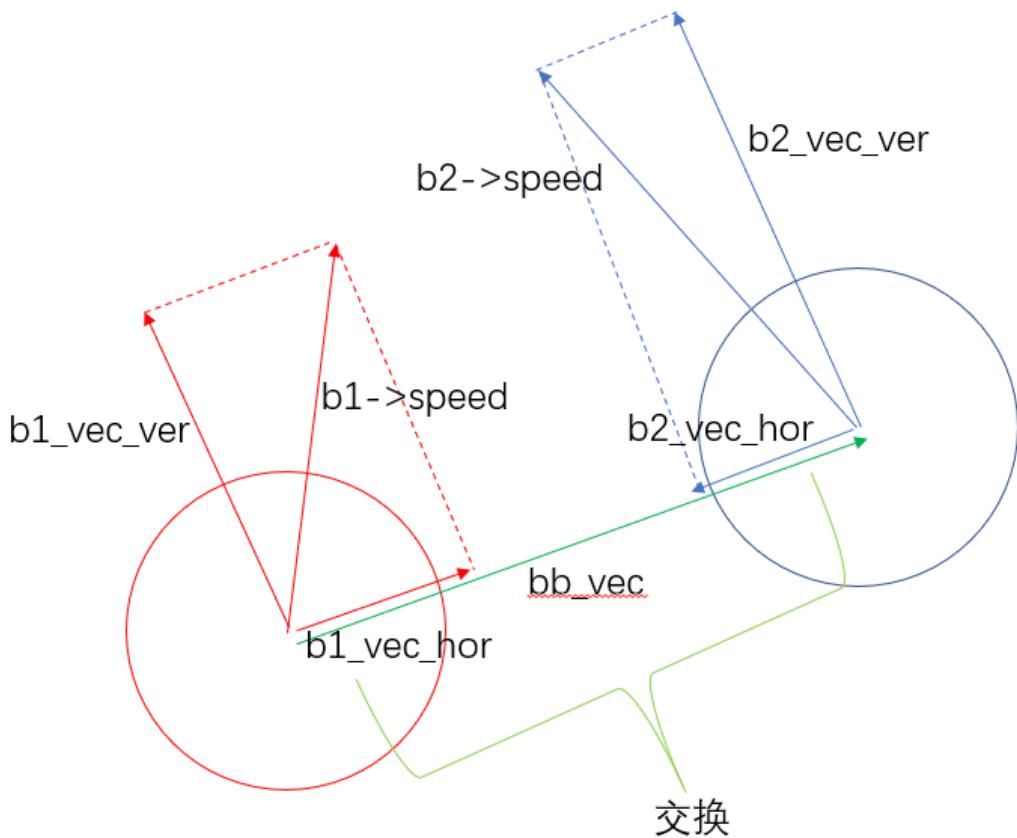
参数描述：b1 和 b2 是两个碰撞的球指针

返回值描述：无

重要局部变量定义：coordinate bb_vec, coordinate b1_vec_hor, coordinate b1_vec_ver, coordinate b2_vec_hor, coordinate b2_vec_ver.

重要局部变量用途描述：bb_vec 是两碰撞球之间的连线向量，b1_vec_hor 是 b1 的速度向量相对于 bb_vec 的水平分量，_ver 则是垂直分量，b2 同理。这几个向量用于做交换速度的中间变量。

函数算法描述：交换水平方向速度，最终总速度乘上衰减系数 HIT_REMAIN_RATION(本程序中设为 0.975，也可以设为 1，代表完全弹性碰撞)。需要注意的是由于球不是连续移动的，所以实际判定碰撞时两球已经相交，模拟有一定的误差。



3.5.2.8:

函数原型: void edge_hit(ball* b)

功能描述: 模拟台球与桌面边缘碰撞的过程, 垂直于桌面的速度分量反向, 总速度乘以衰减系数。

参数描述: b 是与边缘碰撞的球指针

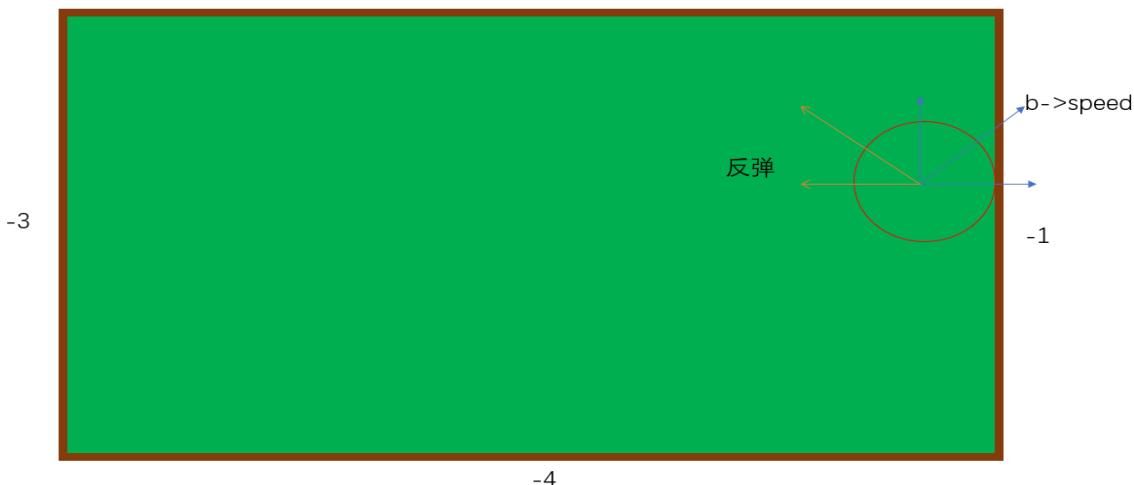
返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 如图, 四面墙用数字代表。

-2



3.5.2.9:

函数原型: void move_ball()

功能描述: 不断在 display_game 中被调用, 内部调用 balls_hit 和 edge_hit, 实现球的移动、碰撞、进洞等基本行为, 更新分数、目标, 调用回合控制函数 turn_control, 是关键函数之一。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 主要用 if 语句。首先逐个判断在场的球是否要进洞, 如果进洞且不是母球, 更改其 present 值为 0、归零速度, 根据模式改变分数, 将其编号存入对应玩家的球槽链表中。如果母球进洞, 传递母球进洞的犯规事件, 将母球速度归零并重置。第二步判断是否有球撞到边缘。如果有, 调用 edge_hit。然后判断是否有两球相撞, 若有, 调用 balls_hit (此处决定此程序的复杂度, 双循环遍历台球数组)。最后控制在场的球的移动和匀减速。

3.5.2.10:

函数原型: void correct_pos()

功能描述: event_control 中都调用一次, 在一个轮次结束后, 纠正重叠球

的位置，防止意外情况发生。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：遍历判断两球是否重叠，如若重叠，沿两球球心连线方向错开它们的位置。母球出桌也纠正回来。

3. 5. 2. 11:

函数原型：void myMouseEvent(int x, int y, int button, int event)

功能描述：鼠标回调函数，主要实现击球时的方向选择与蓄力。

参数描述：x、y 分别是鼠标的横纵坐标，button、event 是鼠标传进来的事件。

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：主要修改全局变量。根据是否按下鼠标决定 if_fixed 的值（1 代表杆固定方向、蓄力，0 代表未固定、选择方向）。记录 g_press_pos，来表示蓄力时刚开始的鼠标坐标。更新 mouse_out 的值以判断鼠标是否出了窗口，防止鼠标出窗口时系统误认为玩家一直不松手。

（具体击球实现请参见下面 draw.c 中 draw_stick 的图示）

3. 5. 2. 12:

函数原型：void myKeyboardEvent(int key, int event)

功能描述：键盘回调函数，负责在可以发自由球的时候控制母球的置球位置

参数描述：key 和 event 是键盘传过来的事件，分别代表键位和按下或者松开

返回值描述：无

重要局部变量定义：char move_c[4], double move_speed

重要局部变量用途描述：move_c 代表控制移动的四个键 WASD. move_speed 代表按一次键母球移动的量（移动速度）

函数算法描述：无特别算法，控制母球移动即可。注意要保证不发生重叠（overlap）、保证杆尖时刻跟随母球球心。

3. 5. 2. 13:

函数原型：void delete_list(List head)

功能描述：删除头指针 head 所指的链表，主要初始化清空球槽时（Init_game_data 和 read_data）中调用。

参数描述：head 是要删除的链表的头指针

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：遍历节点，临时储存下一个节点，释放掉当前节点，后移。

3. 5. 3: draw.c 中的函数

3. 5. 3. 1:

函数原型：void define_my_colors();

功能描述：由于系统的定义颜色不够使用，我们自定义了五种颜色：Table Green 桌布绿，Beige 米色，Light Beige 淡米色，Dark Yellow 深黄色，Edge Brown 边框棕色

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：直接调用 graphics 库中的 DefineColor 函数，根据资料配置所需颜色的 RGB 值即可。

3. 5. 3. 2:

函数原型: void draw_dot_line(double dx, double dy)

功能描述: 实现画虚线的函数(与 DrawLine 基本类似, 只不过画的是虚线), 在指示器绘制中被调用。

参数描述: dx 为 x 坐标的偏离方向, dy 为 y 坐标的偏离方向。

返回值描述: 无

重要局部变量定义: int n

重要局部变量用途描述: 用于判断 for 循环何时结束, 其赋值为点坐标的二元向量

函数算法描述: 与 DrawLine() 函数类似, 主要依靠 for 循环, 每间隔 1 个点就调用一次 DrawLine() 函数, 在此时的坐标上画一条短线, 然后调用 MovePen() 函数移动坐标到下一个位置

3. 5. 3. 3:

函数原型: void DrawCircle(coordinate centre, double r, string color)

功能描述: 实现画实心圆的函数, 画笔最终在圆心。

参数描述: centre 为圆心的坐标, r 为圆的半径, color 为圆填充的颜色

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 主要调用 DrawArc() 函数, 通过圆心 centre 的坐标画圆, 并且使用 StartFillRegion() 和 EndFillRegion() 函数填充圆

3. 5. 3. 4:

函数原型: void draw_background()

功能描述: 绘制开始界面背景颜色图的函数。可以绘制出一个纯色背景界面, 同时有两个小球沿着边框做逆时针运动

参数描述: 无

返回值描述:

重要局部变量定义: static double max_x, min_x, max_y, min_y, x, y, speed

重要局部变量用途描述：用于控制双子球的运动，`max_x`、`max_y` 和 `min_x`、`min_y` 四个变量控制小球可以运动的范围，`x` 和 `y` 是小球的实时坐标，`speed` 控制小球运动的速度

函数算法描述：主要通过 `if` 语句，判断小球是否在规定的区域内，如果没有超过指定范围则小球按 `speed` 做匀速直线运动，每次刷新移动 `speed` 的值的距离，超过范围则不做处理

3.5.3.5:

函数原型：`void draw_one_ball(ball d_ball)`

功能描述：实现画一个台球的函数，画出球体、编号、横纹、目标球指示环。

参数描述：`d_ball` 为要画的台球的结构体

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：主要调用 `DrawArc()` 和 `MovePen()` 函数，根据 `d_ball` 的坐标、颜色、编号进行绘制

3.5.3.6:

函数原型：`void draw_balls()`

功能描述：实现画所有的台球的函数，重置进洞的母球

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：主要调用 `draw_one_ball()` 函数，如果白球进洞了则重置白球的位置

3.5.3.7:

函数原型：`void draw_table()`

功能描述：绘制球桌桌面的函数。画一个矩形球桌，显示台球的初始位置和球洞等部分，显示在图形界面上。

参数描述：无

返回值描述：无

重要局部变量定义：double pen_size_loss

重要局部变量用途描述：边框采用加粗的画笔直接 DrawLine 画出，这个量约为画笔宽度的一半。

函数算法描述：主要调用 DrawLine() 和 MovePen() 函数，绘制球桌的内外框、球洞、置球线以及美化桌面

3. 5. 3. 8:

函数原型：void draw_stick()

功能描述：绘制击球杆，并根据全局变量控制击球功能。

参数描述：无

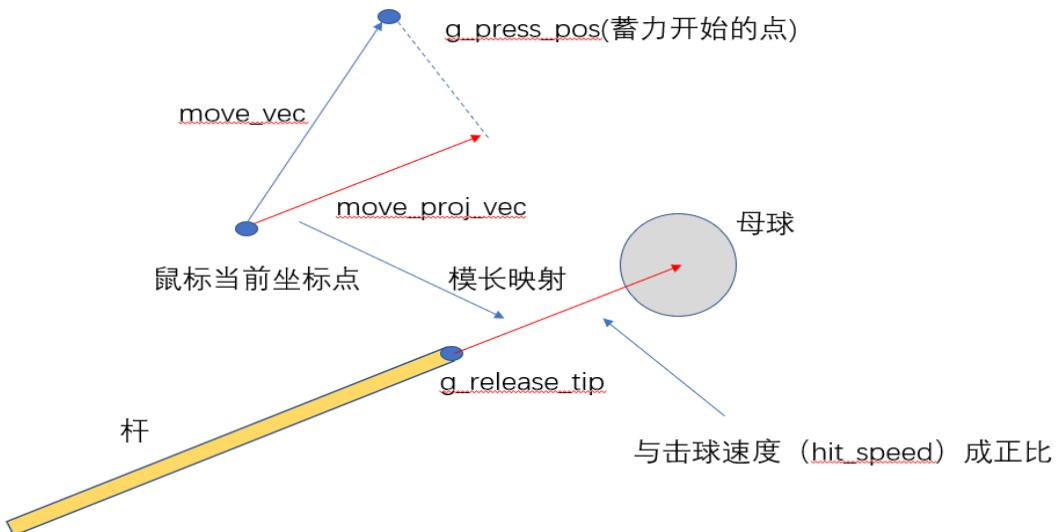
返回值描述：无

重要局部变量定义：coordin stick_vec

重要局部变量用途描述：该变量为杆的方向向量，用于判断杆的蓄力值和所运动的方向

函数算法描述：主要使用向量运算，杆尖跟随白球的坐标，鼠标按下后开始蓄力，鼠标松开后表示蓄力结束，此时计算鼠标两次按下的坐标之间的向量，在杆方向上的投影作为蓄力的值，通过一个 if 条件语句实时记录杆尖的位置。同时需要判断，如果鼠标移到图形界面外部，球杆回到初始位置不进行击球，如果没有超出范围，则根据蓄力值计算球杆击球赋予白球的初始速度和初始方向。注意我们给蓄力值加了一个上限（防止穿透）。

函数内用 if 语句判断杆的四种状态：未固定（可自由移动）、已经固定蓄力、鼠标移出窗口杆复位、正常蓄力后松手击球。最后两段 DrawLine 画出球杆。其中，杆的蓄力程度由全局变量 g_press_pos(myMouseEvent 中决定)，g_release_tip 记录松手前杆尖的位置。几何关系如图。



3. 5. 3. 9:

函数原型: void draw_indicator()

功能描述: 绘制击球轨迹、击球后运动方向和撞墙后反弹方向的指示器

参数描述: 无

返回值描述: 无

重要局部变量定义: int hit_ball_id, coordinate ind_pos

重要局部变量用途描述: hit_ball_id: 用于标识将要的目标球的编号。

ind_pos: 记录被击球的球心坐标

函数算法描述: 主要使用向量运算, 分为如下几部分:

<1>确定被击球: 首先遍历当前球桌上所有球的编号, 算出在击球方向(鼠标决定)正向的、离击球方向垂直距离小于 $1.99*R$ (需要留一点空余)且距离最短的台球的编号。

<2>根据向量运算(有一点复杂)计算运动痕迹, 然后绘制相切的虚拟球、被击球的运行方向、母球的击后运动方向。

<3>如果母球沿当前方向不会击到任何球, 则说明小球会与墙产生碰撞, 此时同样根据向量运算, 绘制与墙相切的虚拟球和撞墙后的反弹路线。

3. 5. 3. 10:

函数原型: void draw_groove()

功能描述: 利用球槽链表, 绘制球槽, 按顺序绘制槽中进洞的台球。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：用基本绘制函数绘制球槽外壳。主要利用链表知识，由 move_ball() 函数中传来的数据，判断进球成功后，记录进球的编号颜色等信息，存储在一个链表里，在球槽中遍历链表绘制进洞的台球。

3. 5. 3. 11:

函数原型：void drawBegin()

功能描述：绘制程序的初始界面

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：主要利用 libgraphics 图形库的函数，绘制菜单和按钮、logo。点击按钮实现对应的功能。

3. 5. 3. 12:

函数原型：void drawNewGame()

功能描述：绘制新建游戏的界面，选择模式后修改关键全局变量。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：主要利用 libgraphics 图形库的函数，绘制新建游戏的菜单，包含四种游戏模式、读取存档、返回、退出游戏按钮。选择一种模式后，会修改 g_mode 和 g_page 的值，设置球的总数，初始化各个游戏中的数据（调用 Init_game_data()），开始计时。如果读档，直接调用 read_data()，这是

read_data 的唯一接口。

3. 5. 3. 13:

函数原型: void drawMenu()

功能描述: 绘制程序的菜单界面, 绘制下拉菜单栏, 设置菜单栏范围和各种按钮, 点击按钮实现相应的功能。同时, 绘制状态栏, 实时更新游戏时间。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 无特别算法, 主要就是通过调用其它函数和修改全局变量的值来实现。

3. 5. 3. 14:

函数原型: void draw_text()

功能描述: 绘制游戏中的各种提示文本。(包括轮次数、击球方、目标球、玩家事件及连续犯规数)

参数描述: 无

返回值描述: 无

重要局部变量定义: string str; string text

重要局部变量用途描述: 便于调用 itoa 和 strcat 函数, 无特别意义。

函数算法描述: 根据全局变量, 绘制游戏状态提示, 通过 if 条件语句根据不同的游戏模式、玩家事件、进球状态等绘制不同的 string。此函数内不修改任何全局变量的值, 起纯粹的文本绘制作用。

3. 5. 3. 15:

函数原型: void draw_win_text(int winner)

功能描述: 只在 drawMenu 中, g_is_win==TRU 时调用, 绘制获胜文本。

参数描述: winner 是获胜方, 0 代表 A, 1 代表 B

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：根据模式和 winner 直接绘制获胜提示文本

3. 5. 3. 16:

函数原型：void display_game()

功能描述：游戏显示函数，随着时间回调函数不断调用，刷新游戏内的界面，同时调用 move_ball。依次移动球、画桌子、画球槽、画台球、画杆、画指示器、画提示文本。其中杆和指示器只在可击球时绘制。调用回合控制函数。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：主要利用时间回调函数刷新，调用了所有的绘制图像的函数和回合控制函数。另外一直判断是否所有球的静止，是的话更新所有球的 just_hit 为 -5（无刚击打对象），画杆和指示器。

3. 5. 3. 17:

函数原型：void display(int timerID)

功能描述：时间回调函数，不断被定时器回调实现页面的刷新

参数描述：timerID 为定时器的编号，本程序中有两个：TIMER_GAME 和 TIMER_MENU

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：非暂停情况下只接受 TIMER_GAME，暂停情况下，TIMER_GAME 已经被删除，只接受 TIMER_MENU。根据 g_page 绘制对应的页面，主要根据定时器的时间间隔进行刷新，在这个函数中也控制了游戏的暂停和开始，如果暂停了，

则取消游戏进行的定时器但是不取消绘制菜单界面的定时器，实现暂停的效果。

3.5.4: record.c 中的函数

3.5.4.1:

函数原型: void save_data()

功能描述: 实现游戏进度保存为文件的函数, 将记录游戏进程的全局变量存入文本文件和二进制文件中, 只在 DrawMenu 中点击保存游戏进度后调用。

参数描述: 无

返回值描述: 无

重要局部变量定义: FILE* fp1, * fp2

重要局部变量用途描述: 指向文件的两个文件指针, 用于读和写操作

函数算法描述: 首先调用 delete_data() 函数, 删除原有的存档。然后将游戏过程中的全局变量分两批, 分别写入文本文件和二进制文件中, 包括小球的数量、游戏时间、小球的坐标等信息。如果球槽中不为空, 则需要写入球槽中的链表信息。

3.5.4.2:

函数原型: void read_data()

功能描述: 实现读取文件中存储的游戏信息的函数, 只在 DrawNewGame 中点击读取存档按钮时调用, 直接进入对应游戏状态。

参数描述: 无

返回值描述: 无

重要局部变量定义: FILE* fp1, * fp2

重要局部变量用途描述: 指向文件的两个文件指针, 用于读和写操作

函数算法描述: 打开 txt 文件, 根据不同的变量类型依次读取变量值, 存储在相应的全局变量中。如果球槽链表不为空, 则读取链表信息还原到图形界面中。

3.5.4.3:

函数原型: void delete_data()

功能描述: 实现删除文件中存储的游戏信息的函数, 只在 save_date() 最开

始被调用。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：主要是调用 `remove()` 函数，将文本文件和二进制文件中的所有数据都删除，防止重复录入。

4 部署运行和使用说明

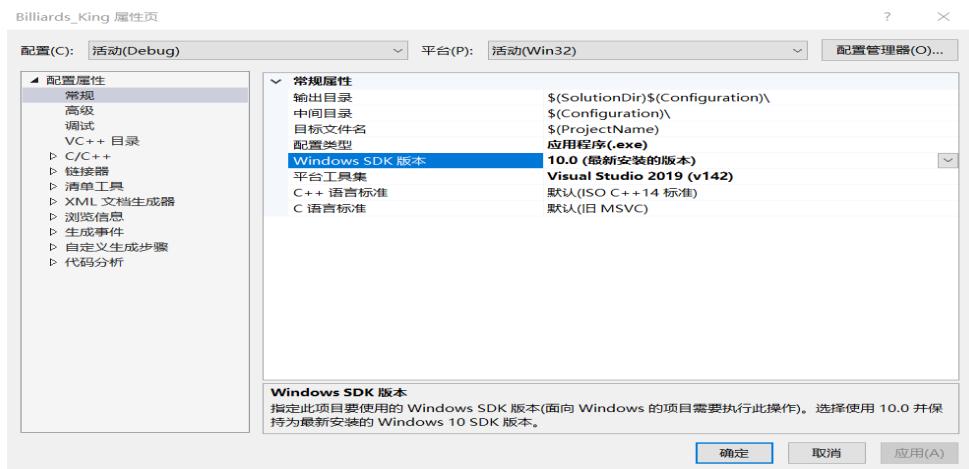
4.1 编译安装

编译环境：Visual Studio 2017

项目文件目录：头文件和相关依赖存储在文件夹 `libgraphics` 和 `simpleGUI` 中，项目文件和源文件存储在文件夹 `code` 中，可执行文件在项目文件夹的 `debug` 文件夹中。

使用方法：

4.1.1（方法一：直接运行源项目文件）点击 `Billiards_King.sln` 文件，打开 Visual Studio 软件，点击菜单栏上“生成 -> 重新生成解决方案”，编译成功后，按快捷键 F5 或者点击菜单栏“调试 -> 开始调试”运行程序。**注意：**如果出现 `stdio.h` 等头文件无法找到的情况，一般原因是 VS 版本和项目版本不匹配，**点击“项目 -> 项目属性 -> 常规”将 Windows SDK 版本改成最新的 10.0.17763.0(或适配具体环境的版本)**，然后修改平台工具集为对应版本 **Visual Studio 2017(v141)**之后，重新生成解决方案即可成功执行。



4.1.2（方法二：运行源代码）打开 Visual Studio 软件，新建一个空的解决方案，新建 libgraphics 和 simpleGUI 两个文件夹，将项目相关库全部导入（右键 -> 添加 -> 现有项），最后导入源文件（方法同上，源文件包括：Main.c, global.h, compute_judge.h, compute_judge.c, draw.h, draw.c, control.h, control.c, record.h, record.c）。**注意：**在生成解决方案前，需要确保“项目 -> 项目属性 -> C/C++ -> SDL 检查”这一栏为否，然后按照第一步操作，正常生成可执行文件。

4.2 运行测试

4.2.1:

球碰撞后振动的 bug： 测试过程中遇到的第一个大问题是实现球与球、球与边缘的碰撞后，两球、球与桌边之间碰撞后，两者会纠缠在一起，不断振动、无法分离。我分析了实现碰撞的代码原理，发现于实际碰撞不同的是；程序中通过球的位置间断性地变化来模拟连续的移动，而碰撞的判据是物体间的距离，碰撞后球马上有一个速度分量反向，但在沿这个反向的速度分量移动一帧后，极有可能不能马上脱离碰撞范围，从而再次进行了碰撞判定、让本来以及相互离开的球对心速度再次方向，如此往复，显示出了振动的效果。针对这个 bug，我们给球的结构体额外设置了 int just_hit 这一成员，用以表示上一个与球碰撞的单位，正数代表碰撞球的编号，-1~-4 代表四个边缘，-5 代表无（初始化的值，在一个轮次结束后也都变为-5）。这样做的原理是：一个球不会连续与同一个单位碰撞两次（其间至少碰了另一个球或者桌边，或者若另一个单位是球，它在此期间必碰

了别的单位）。根据这个原理，即可从逻辑上防止连续碰撞的发生。

```
if (!(g_balls[i].just_hit == j && g_balls[j].just_hit == i) && bb_distance(g_balls[i], g_balls[j]) < 2 * R)
{
    balls_hit(&g_balls[i], &g_balls[j]);
}
```

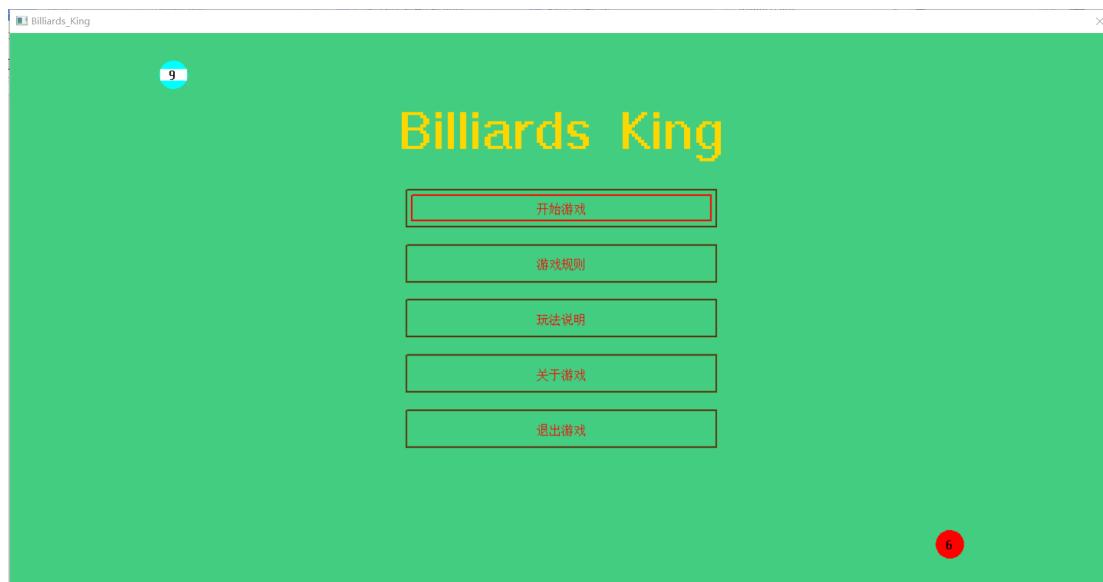
4.2.2:

链表的文件储存终点问题：在 `save_data` 和 `read_data` 中，球槽链表中数据的存取是重要的一部分。链表存储的一个关键是如何判断终点，开始我尝试把 `next` 指针传入，后来发现由于链表是动态分配的，每次的地址都不一样，不能用这个追踪。所以我增设了表示两链表长度的全局变量 `g_grove_A_len` 和 `g_groove_B_len`，将其也存入文件中，到时按这个数量进行读取即可。

4.3 使用操作

本项目的具体使用操作如下流程图所示：

- 开始界面



此界面为运行程序后的第一个界面，包括开始游戏、游戏规则、玩法说明、关于游戏和退出游戏五个按钮，每个按钮提供对应的功能。

① 开始游戏：点击按钮后进入下一个界面，可以选择不同的游戏模式；

② 游戏规则：点击按钮后，会弹出一个系统消息框，可以查看不同模式的游戏规则；

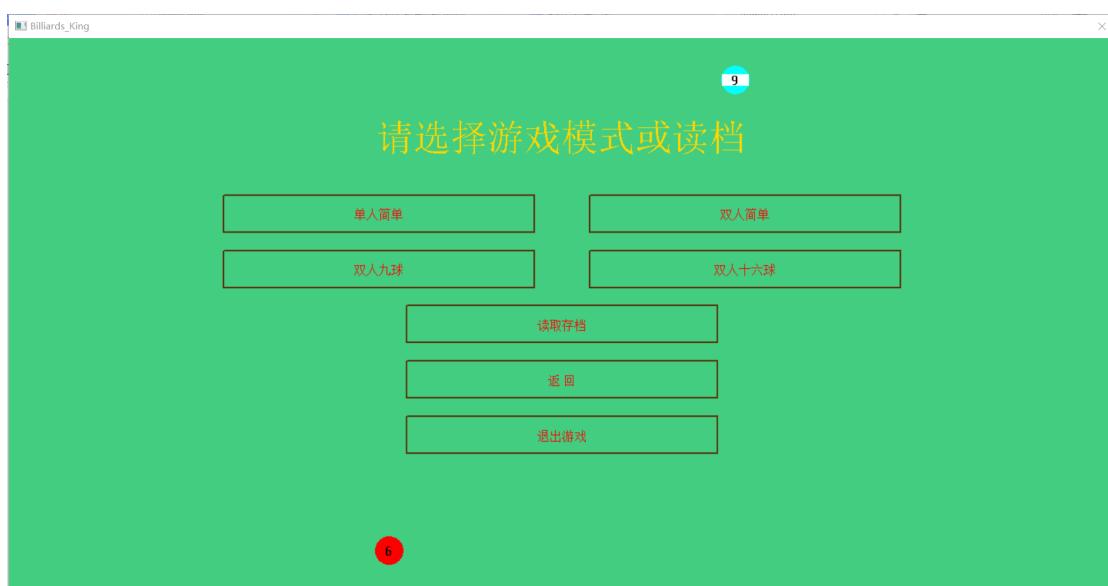
③ 玩法说明：点击按钮后，会弹出一个系统消息框，可以查看游戏整体的操作，包括如何使用球杆，如何击球等方面；



④ 关于游戏：点击按钮后，会弹出一个系统消息框，可以查看该游戏的版本和开发团队；

⑤ 退出游戏：点击按钮后，关闭图形界面窗口。

● 模式选择界面



此界面为点击【开始游戏】按钮后的界面，可以进行不同模式的选择，包括单人简单、双人简单、双人九球和双人十六球等模式，并且可以读取存档和返回上一个界面，每个按钮提供对应的功能。

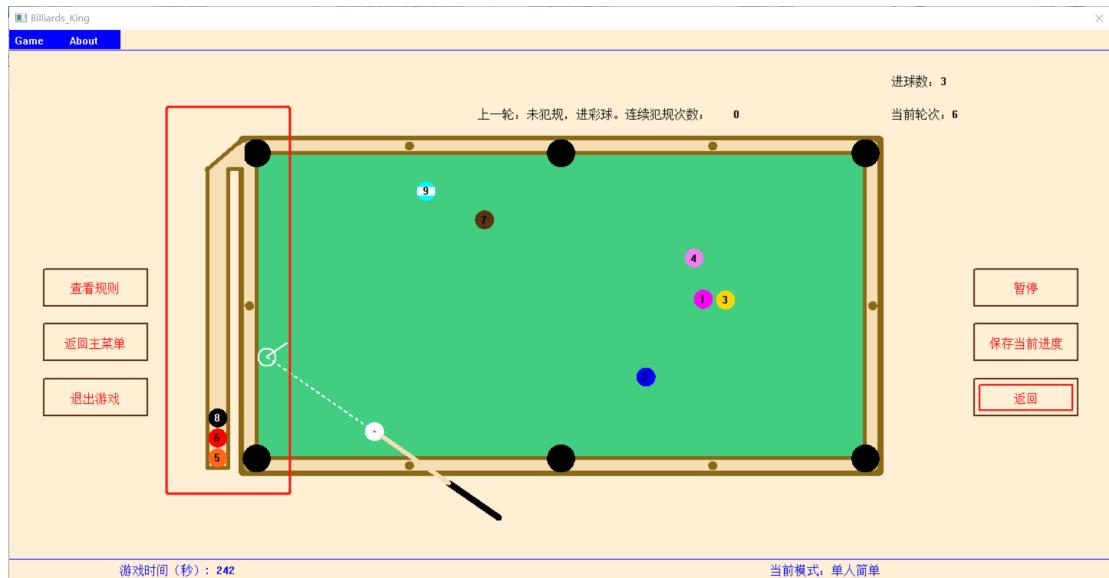
- ① 单人简单、双人简单、双人九球和双人十六球：点击按钮后，根据不同的模式进入到游戏界面；
- ② 读取存档：点击按钮后，系统会自动检索同一文件夹内的二进制文件 binaryfiles.txt 或者文本文件 txtfiles.txt，然后打开游戏界面；如果文件夹不存在相应文件或者是存档文件失效，则点击按钮不会进入游戏。

● 游戏界面



此界面为点击【单人简单】按钮后的界面(以此为例，其他模式界面相似)，这是本项目的主要界面。

中间是游戏的台球球桌，台球的初始位置与游戏的模式有关；球桌左边的三个按钮分别是查看规则、返回主菜单、退出游戏，点击可执行对应功能，但是此时返回主菜单或者退出游戏，均不会保存游戏当前进度；球桌右边的三个按钮是暂停、保存当前进度、返回，点击可执行对应功能，此时的【返回】为返回模式选择界面。球桌上会提示击球状态和当前轮次，界面下方提示游戏时间和当前模式。



上图红框标注的部分会显示当前进洞的球号，进球顺序为从下到上，以提醒玩家。



上图展示的内容为：如果白球进洞后，记一次犯规并且白球回到默认固定的位置。



上图展示的为双人九球模式下的游戏界面，基本功能同单人简单模式，不同的是双人模式增加了“自由球”规则，初始击球方和上一轮对手犯规后，可以出现“自由球”。“自由球”可以在球桌上随意移动，由键盘 WASD 控制，改变的是白球的位置（详细可以见规则）。



上图展示为【暂停】功能，点击暂停按钮后，小球会停止运行并且球杆相应消失，此时可以保存当前进度，保存游戏存档后，该项目文件夹内会生成二进制文件 binaryfiles.txt 和文本文件 txtfiles.txt（如下图）。点击【继续游戏】按钮后，游戏会重新进行。

[binaryfiles.txt](#)

2021/5/30 22:13

文本文档

2 KB

[txtfiles.txt](#)

2021/5/30 22:13

文本文档

1 KB

以上即为该游戏的基本操作介绍，不同的模式下由不同的规则，请玩家自行体验。

3 团队合作

3.1 任务分工

任务	成员	具体描述
组长	管嘉瑞	实现内部基本原理，实现游戏中的球桌、台球、球槽等基本元素的绘制；实现台球的相互碰撞、杆的击球、计分；研究实际的台球规则并利用回合制实现，实现双人九球模式和双人十六球模式。完善论文。
组员	杨锐洁	负责 UI 部分，完成对菜单页面的设计与代码实现，实现程序与玩家的交互。撰写论文。进行程序的测试。建立、调整、压缩项目文件。实现单人简单模式。
组员	王舒杨	利用文本文件和二进制文件的读写知识，实现游戏过程的保存与读档。撰写论文。测试程序并提出改进建议。实现双人简单模式。

3.2 开发计划

时间	里程碑	具体描述
2021.4.29 - 2021.5.11	熟悉项目，确认项目的主题	第一次线下讨论，确定了简易台球游戏为大程选题，并且基本完善游戏实现的思路
2021.5.12 - 2021.5.17	初步完成程序的功能	实现小球运动、碰撞等游戏基本原理，实现菜单功能
2021.5.18	第二次线下讨论	总结并分配下周任务，主要完

		成程序的基本功能，并且整合部分代码
2021.5.19 - 2021.5.24	进一步完善程序的功能	实现游戏暂停、球杆自动索引白球、击球路线显示等功能，实现文件的保存
2021.5.25	第三次线下讨论	整合所有代码，进行多次测试并修复 BUG
2021.5.26 - 2021.5.30	完成程序编写，细化游戏功能	美化界面，对源代码进行分组，编写报告
2021.5.31	项目打包上交，开始互评	

3.3 编码规范

本项目编码规范主要参考课程文件《本学期作业代码自检规范》，具体有以下几个方面：

命名规范：函数名称、变量名称等一律采用小写方式，以下划线进行分割，宏定义一律采用大写，同样以下划线进行分割；

注释规范：一律采用中文注释，模块的注释放在模块前，对单一函数或者变量的注释放在其右侧；

代码规范：程序中出现的有意义的判断常量，均采用宏定义的方式规范来增加代码的可读性。复杂表达式，必须用括号分组，显式规定其优先级。

3.4 合作总结

2021 年 4 月 29 日 - 5 月 10 日：进行线上讨论，提前了解每位同学的编程偏好，并且确定了第一次线下讨论时间和地点。



2021年5月11日：第一次线下讨论，讨论了每个题目的难点和突破点，讨论确定了大作业选题，随后参考了现有的台球小程序，针对项目的每个细节进行了讨论，包括台球桌面、菜单基本模型、碰撞检测、进洞检测等功能。小组成员在一起讨论并编写了代码，基本实现了一些功能后，再各自回去做自己负责的部分。

主题一：排序可视化 简单难突破

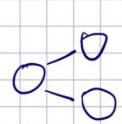
优点：各种排序算法在网络上有许多资料，易于上手
只需对图形界面熟悉。

缺点：目前没想到合适的创新点，以及如何美观地
展示成果

主题二：思维导图 想不明白



优点：大部分基于图形序，主要实现 move 拖拽等操作



缺点：两个 View 之间如何用直线(?)连接，点击后

自动生成？还是用户自己画线连接？连接线如何同
View 绑定在一起？View □ ○ 怎么输入文字。
先画框再定位 Textstring 的输入？

主题三：台球游戏 讨论后应该能实现

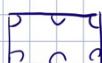


优点：实现不复杂，具有一定难度，思路比较清楚

主要是数学计算，解决 moveball，碰撞检测



可以给每个球定义一个 struct，定义 id, direction
速度，加速度。碰撞就是检测两个球圆心距离
小于直径，再看 x, y 方向的速度，决定球碰撞后
往哪个方向走，更新属性。



缺点：球杆自动牵引？文件保存？游戏规则？

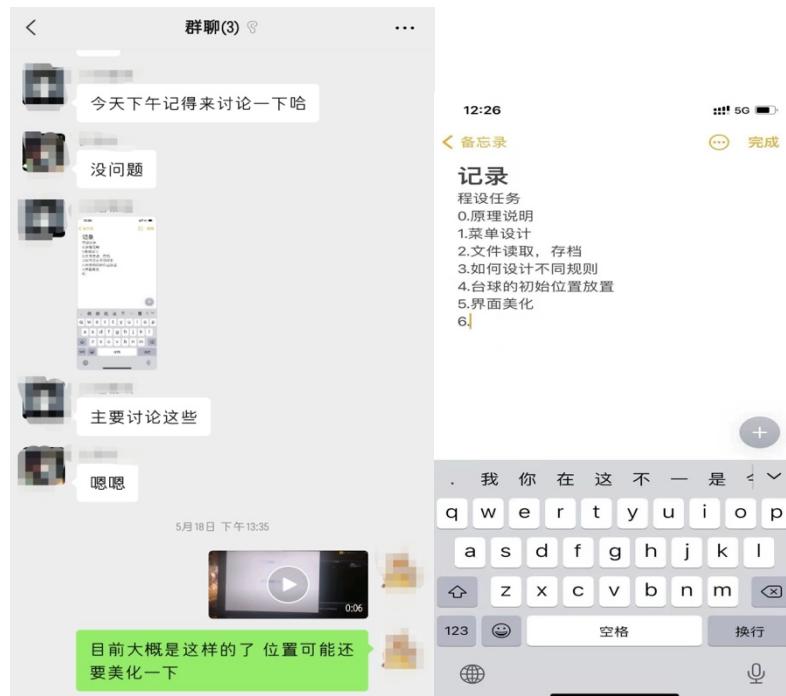
单/双人模式。

13 /
15
▼
④

2021 年 5 月 12 日 - 5 月 17 日：这一星期主要为线上讨论，实现了基本的小球碰撞，进球检测，菜单设计等功能，并且不断讨论细化。



2021年5月18日：第二次线下会议，主要讨论了上一周的成果和本周需要完成的任务，包括详细的菜单设计、文件读取/保存、不同规则/模式的游戏等问题，此时已经能基本完成游戏的功能。目标是增加球杆的自动索引功能和游戏初始菜单、快捷键等的使用，此外也分配了每个人的任务。

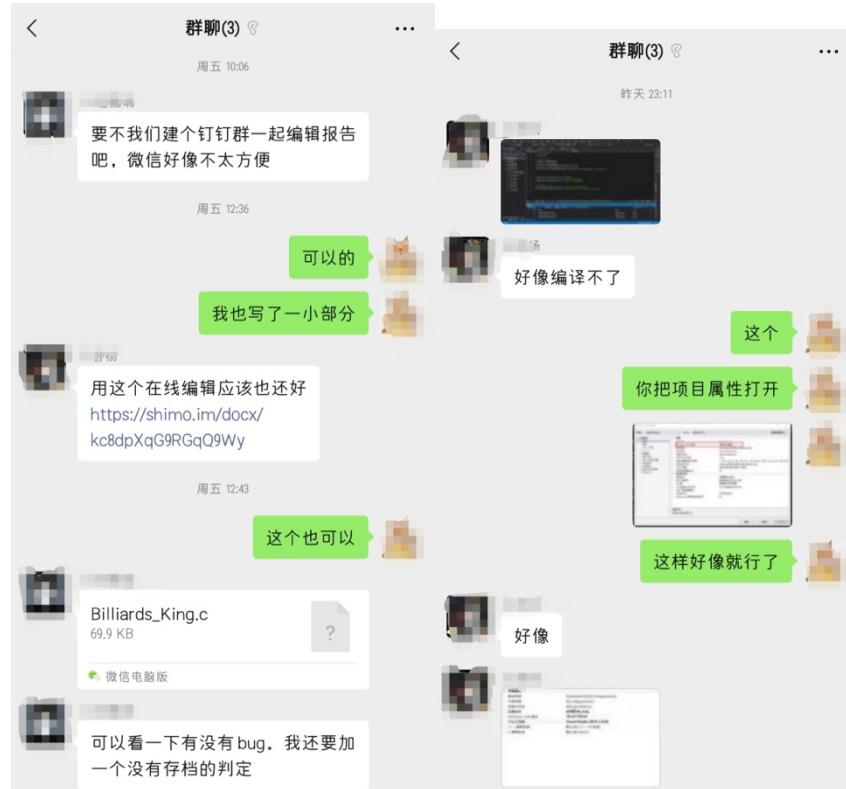


2021年5月19日 - 5月24日：这一周主要是线上讨论，各位成员分别完成了菜单界面、文件和游戏功能模块，但是这周并没有整合完整的系统，成员们互相讨论解决编码中遇到的问题。



2021年5月25日：第三次线下会议，总结了上周任务的完成情况，并且基本上整合了所有成员的代码，所有的功能已经基本完备，在会议过程中，成员间互相交流实现了游戏时间记录、二进制和文本文件的保存等功能。会议最后讨论了本周的任务安排。

2021年5月25日 - 5月30日：这期间为线上讨论，成员们继续对游戏进行细化，并且着手报告的编写，并且交流了不同电脑对项目文件打开情况可能出现的BUG然后进行修复和说明描述。



3.5 收获感言

管嘉瑞：

完成这次大作业是一个收获颇丰的过程。一方面，我加深了对 C 语言的理解，特别是掌握了实际开发项目时的注意事项与规范。在开发的过程中我也巩固了许多知识，比如用链表存储已进的球，期间出过无数的 bug，全都克服后，我对链表的理解更加深刻了。实际开发中会遇到很多具体的问题和困难，最重要的就是如何利用已有知识，利用全局思维，最高效率地解决一个又一个的子问题。另一方面培养了我的团队合作能力，明白了如何在多人项目中与他人分配任务、明确程序接口、讲实现各个功能的模块整合到一起，相信这对我以后的学习、开发都是十分有作用的。做完这个台球游戏还是蛮有成就感的，毕竟是第一次不依赖于现有代码、独立地开发出一个可以玩的、功能相对完整的小游戏，让我对编程的兴趣大大提高了。

杨锐洁：

在这次项目设计的过程中，我深刻感受到团队合作和团队沟通的重要性，有良好的沟通就能加快整体项目的进度，不会耽误进程导致最后项目不能按时完成。小组成员们遇到难题，讨论沟通然后解决难题，这样的团队体验很好，也非常实用。并且，在这次项目中，我主要负责图形界面的设计，也深入了解了 libgraphics 图形库的知识，实践操作更加增强了对 C 语言的理解，也通过小组合作，明白了一些上课没听懂的知识。无疑，这次程序开发过程中带来的体会是一个单纯的结果无法总结的，但是完成这个项目后确实让我重新感受到 C 语言的魅力。

王舒杨：

首先，大程序设计是团队协作的成果，我从组员身上和与组员的交流中学习到了很多珍贵的东西。其次，第一次进行一个小游戏程序的开发，对我而言难度不小，这是一个不断学习的过程，不仅能巩固一些基础知识，还将知识应用在实践中，加深了对 C 语言理论的理解和对一些新知识的了解学习。除此之外，在老师的规范和要求下，也渐渐养成了规范编写代码的习惯。最后，通过这次大程序设计的实践，我切身感受到了 C 语言在实际应用中的强大功能，兴趣也得到了激发，无论是 debug 的枯燥烦恼，还是完成大程序后的兴奋喜悦，都是课程中不可多得的宝贵体验。

4 参考文献资料

- [1] 简单图形库介绍。 (<https://www.docin.com/p-2108779754.html>)
- [2] 童晶、丁海军、金永霞、周小芹《C 语言课程设计与游戏开发实践教程》