



上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

学 生 作 业 报 告

乒乓球比赛模拟与胜率计算程序开发与实现
及赛制改变对比赛结果影响的量化探究

姓名

班级

学号

课程 程序设计思想与方法 (C++)

时间 2021-12-19

摘 要

2001 年国际乒联推行乒乓球赛制改革, 将每局比分由 21 分改为 11 分。为分析赛制改变对比赛结果的影响, 可以建立数学模型, 并使用 C++ 编程语言进行算法实现。本文详细分析了模型的建立过程、程序的架构与功能模块设计, 关键变量的说明, 最终完成了“乒乓球比赛模拟与胜率计算程序”的构建, 实现了赛制改变对不同级别、不同水平乒乓球比赛结果影响的量化计算。

关键词 乒乓球 赛制改革 C++ 量化计算

目 录

1.绪论	2
2.模型构建	2
2.1 选手参数量化	2
2.2 比赛过程模拟	3
3.程序架构设计	3
3.1 程序总体架构	3
3.2 功能模块划分	4
3.3 代码架构	5
4.功能详细设计	5
4.1 用户交互界面	5
4.1.1 cmd_opts 库的关键函数	5
4.1.2 main 函数中用户界面的关键函数	7
4.2 比赛模拟 (ppong 库内容介绍)	13
4.2.1 PlayerLevel 结构体	13
4.2.2 Game 类的数据成员与关键函数	13
5.结果分析	20
5.1 赛制改变对比赛结果的影响	20
5.1.1 二项分布法理论计算	20
5.1.2 程序模拟法统计	21
5.2 总结与启示	23
6.参考文献	23
附录 1 程序运行环境	24
附录 2 全部代码	24
main.cpp 中的代码	24
/lib/cmd_opts.h 中的代码	30
/src/cmd_opts.cpp 中的代码	32
/lib/ppong.h 中的代码	34
/src/ppong.cpp 中的代码	36

1. 绪论

乒乓球是一种世界流行的体育项目，也是中国的“国球”。2001 年 9 月 1 日起，国际乒联推行乒乓球赛制改革，将每局比分由 21 分改为 11 分。时任国际乒联主席沙拉拉声称此项改革的目的是加快比赛节奏、便于电视转播。赛制的改革势必会对参赛选手与比赛结果造成影响。有观点认为，对自从 1988 年起即垄断奥运会金牌的中国队，赛制改变的影响将尤其大。为此，可以构建 C++ 程序，由用户输入比赛与选手的相关参数或使用预置参数，重复模拟比赛进程，在统计学上计算胜率并计算赛制变化前后的胜率变化。

2. 模型构建

2.1 选手参数量化

为模拟比赛进程，需要对双方选手的能力进行量化。与广泛采用的三段统计法不同，本程序采用的模型借鉴并简化了李丕亮等人提出的 10 项指标评估法^[1]，对运动员前四板技术结合进攻、相持、防御 3 方面设立评估指标。分别是：

- (1) 发球直接得分率：指在第一板发球直接得分的概率；
- (2) 进攻时命中率：指进行发球抢攻、接球抢攻或发球反攻时球不出界的概率；
- (3) 非进攻时命中率：除上述情况外球不出界的概率；
- (4) 受进攻时接球成功率：收到对方发球抢攻、接球抢攻或发球反攻且球未出界时接到球的概率；
- (5) 非受进攻时接球成功率：除上述情况外，球未出界时接到球的概率；
- (6) 接球抢攻使用率：接到对手第一板发球后，决定使用接球抢攻战术的概率；
- (7) 发球反攻使用率：接到对手接球抢攻后，决定使用发球反攻战术的概率；
- (8) 发球抢攻使用率：接到对手接发球后，决定使用发球抢攻战术的概率；
- (9) 相持状态能力：一个 0-100 范围的整数，表示进入相持状态后相对能力

以上状态是选手的基础能力数值，为更好模拟比赛的随机性与紧张氛围，以上各命中率与接球成功率的具体值会有所变化，具体实现方法见第 4 节。

2.2 比赛过程模拟

乒乓球比赛分为一场比赛（计大比分）和一局比赛（计小比分）。一局比赛流程简化如下：

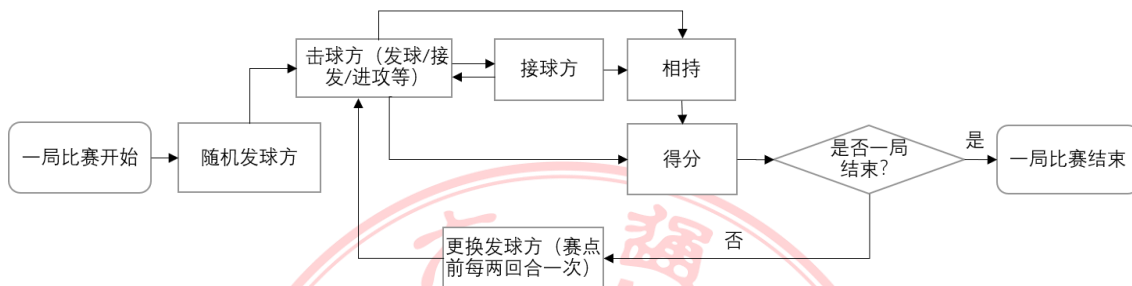


图 2-1 比赛流程简化

将两位选手命名为击球方和接球方，击球动作包括发球、接发球、接球抢攻、发球抢攻和发球反攻等。

当一局比赛开始时，先随机一位选手作为击球方进行第一板发球。发球可能直接得分，若未直接得分则计算是否命中（即是否未出界），若命中，转移到接球方计算是否接住。若接球方成功接住，则判断采取策略，然后接球方变成新的击球方，计算是否命中……击球方和接球方循环往复，直至一方得分，计算比分，若一局比赛未结束则判断是否更换发球方并进入下一次发球。

由于乒乓球战术主要围绕每个球的前三板展开，所以，程序将模拟前三板球的具体过程，当球在场上来回至第四板及以上时，该球将被判断为“进入相持状态”，通过比较双方选手的相持能力判定该球的胜负。

为简化程序设计，默认比赛中不出现犯规的情况。

3. 程序架构设计

3.1 程序总体架构

程序的总体架构如下：



图 3-1 系统架构

3.2 功能模块划分

本程序使用系统 Console 窗口与用户实现交互，实现了自定义参数、预设参数、比赛模拟、结果展示等功能。



图 3-2 功能模块划分

3.3 代码架构

本程序的代码框架如下：

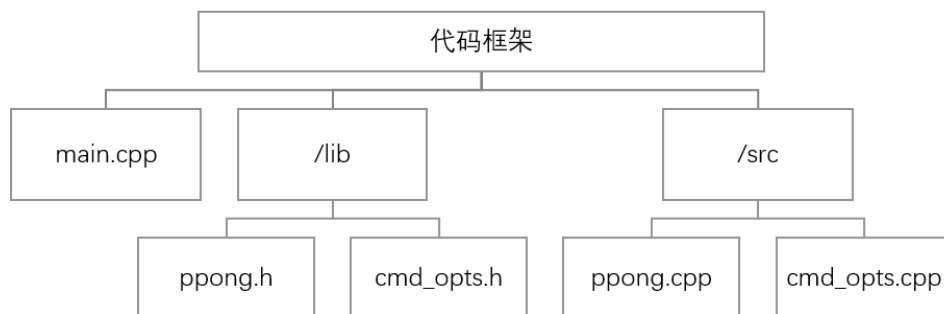


图 3-3 代码框架

程序中包含了两个自定义库：cmd_opts (Console_Operations)库和 ppong (PingPong) 库，它们的头文件和实现文件分开放置于/lib 和/src 子目录下，达到接口与实现过程分开的目的。

cmd_opts 库中包含了简单用户交互界面所需要的一些函数。

ppong 库中包含了储存选手、赛制等参数的类和模拟比赛进程的相关函数。

main.cpp 中包含了与用户交互的大部分代码如功能菜单选择、参数输入和展示。各库中的详细设计和关键函数、类及数据成员将在下节叙述。

4.功能详细设计

4.1 用户交互界面

4.1.1 cmd_opts 库的关键函数

cmd_opts 库基于 windows.h 和 conio.h 库，包含了关于用户界面交互的一些函数，如 Console 窗口的参数设置、输入事件等。

为简化，将标准输入输出设备的句柄简化为全局常量。

```
static const HANDLE __hout = GetStdHandle(STD_OUTPUT_HANDLE);  
static const HANDLE __hin = GetStdHandle(STD_INPUT_HANDLE);
```

cmd_opts 库的全部函数如下（关于函数用途的注释已省略）：

```
void copts_setconsoletitle(const char *title);
void copts_fixconsolesize(void);
void copts_cls(void);
void copts_gotoxy(const int X, const int Y);
void copts_printxy(const int &X,const int &Y,std::string str);
void copts_printspace(const int &X,const int &Y,const int &rp
t);
void copts_printline(const int &X,const int &Y,const int &rpt);
void copts_setconsoleborder(int set_cols, int set_lines, int se
t_buffer_cols, int set_buffer_lines);
void copts_showcursor();
void copts_hidecursor();
```

以下是部分关键函数的功能说明、参数说明、代码及运行结果：

(1) copts_cls 清屏函数

功能：清除屏幕所有内容，用于页面的初始化和切换

```
void copts_cls(){
    system("cls");
}
```

(2) copts_gotoxy 函数

功能：将光标移动到指定位置，以屏幕左上角坐标为(0,0)

参数：const int X: X轴坐标（列）；const int Y: Y轴坐标（行）

```
void copts_gotoxy(const int X, const int Y){
    COORD coord;
    coord.X = X;
    coord.Y = Y;
    SetConsoleCursorPosition(__hout, coord);
}
```

(3) copts_printxy 输出字符串函数

功能：在指定位置输出一个字符串，以屏幕左上角坐标为(0,0)

参数：const int X: X轴坐标（列）；const int Y: Y轴坐标（行）；string str:

要输出的字符串


```
void copts_printxy(const int &X,const int &Y,string str){
    copts_gotoxy(X,Y); cout<<str;
}
```

4.1.2 main 函数中用户界面的关键函数

本程序含有主菜单、参数选择、选手能力展示页、比赛演示等多个页面，采用一系列的函数来切换页面。同时，在允许来回切换的页面，多使用 while(1)语句，使二级界面的函数返回后重新加载一级界面。这样的方法相较于递归，可以防止多次页面切换后递归层数过多的问题。

main 函数中关于用户界面的全部函数如下：

```
void InitializeWindow(); //窗口初始化
void Page_MainMenu(); //主菜单页
void Page_Custom_Paraset(); //自定义计算参数设置
void Page_Custom_Paraset_RMenu(const int index); //自定义计算参数界面右侧菜单（此函数只负责显示）
void Page_Custom_Paraset_Input(PlayerLevel &y,int id); //自定义选手参数输入界面
bool Page_CustomII();
void Page_Demo(); //演示计算页面 I
void Page_DemoII(); //演示计算页面 II
void ShowPlrPara(int i); //显示选手参数（子函数）
void Page_Result(); //显示结果
```

main 中各函数调用的关系图（即用户使用流程图）如下：

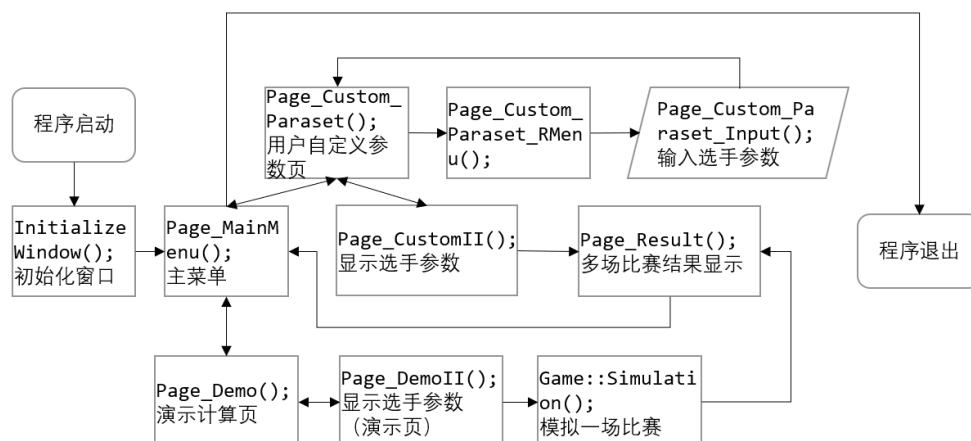


图 4-1 main 中各函数调用关系（用户使用流程图）

以下是部分关键函数的功能说明、代码及运行结果：

(1) 主菜单 Page_MainMenu 函数

功能：程序开始时显示的主菜单页面。

```
void Page_MainMenu(){
    int keyin;
    ..... //显示菜单文字
    while(1){
        keyin=_getch()-48;
        if (keyin==-21) exit(0); //ESC 27-48
        if (keyin==1||keyin==2) break;
    }
    if (keyin==1) Page_Custom_Paraset();
    else if (keyin==2) Page_Demo();
    return;
}
```

运行结果：



图 4-2 主菜单运行结果

运行后，程序使用 getch 函数监听键盘按下按钮的 ASCII 码并判断，直到用户按下数字 1、2 或 ESC 键。按 1 执行 Page_Custom_Paraset 函数并进入自定义参数界面；按 2 执行 Page_Demo 函数并进入演示界面；按 ESC 键使用 exit (0)

命令退出程序。

(2) 自定义参数界面 Page_Custom_Paraset 函数与 Page_Custom_Paraset_R menu 函数

功能：显示自定义参数界面，用户在此界面输入选手参数、选择赛制。

本函数中利用标签和 goto 语句，当下一个页面返回后可以重新加载本页面。

```
void Page_Custom_Paraset(){
    para:
    int keyin,select_index=1;
    .....//显示一级菜单及顶部菜单
    while(1){
        keyin=_getch();
        if (keyin==27) exit(0); //ESC 27
        if (keyin==8) return; //退格 8
        else if(keyin==78||keyin==110){ //N 78 n 110 开始
            usergame.demo=false;
            bool ret=Page_CustomII();
            if(ret) goto para;
            else return;
        }
        else if (keyin==(224)){ //方向键控制左侧列表上下
            int direct;
            direct=_getch();
            switch (direct){
                case 72: ..... //一级菜单上移
                case 80: ..... //一级菜单下移
            }
            .....//显示对应二级菜单
        }
        else if(keyin>=48&&keyin<=57){ //数字键选择二级菜单
            switch(select_index){
                case 1:{
                    switch(keyin){
                        case 49:Page_Custom_Paraset_Input(userga
me.plr[0],0);goto para;break; //输入选手 A 能力数据
                        case 50:Page_Custom_Paraset_Input(userga
me.plr[1],1);goto para;break; //输入选手 B 能力数据
                        case 51:{ .....//使选手 B 和 A 能力数据相同
                    }
                }
            }
        }
    }
}
```

```
        }  
    }  
    case 2:{  
        ..... //等待用户输入数字  
        break;  
    }  
    case 3: {  
        ..... //等待用户输入数字  
        break;  
    }  
}  
}  
}
```

运行结果:



图 4-3 自定义选手参数菜单



图 4-4 自定义赛制参数菜单

用户按上下方向键选择左侧的一级菜单, 按方框中的数字键选择右侧的二级菜单; ESC 键退出程序; 退格键返回上一级 (主菜单); N 键进入下一步。

(3) 选手能力表 Page_Customll 函数、Page_Demo 函数与 ShowPlrPara 函数

功能: 显示选手能力参数

自定义参数模式下执行 Page_Customll 函数, 演示模式下执行 Page_Demo 函数。二者不同之处在于, 演示模式运行 Page_Demo 后将首先随机两位选手的各项参数。ShowPlrPara 为前两者的子函数。

运行结果:

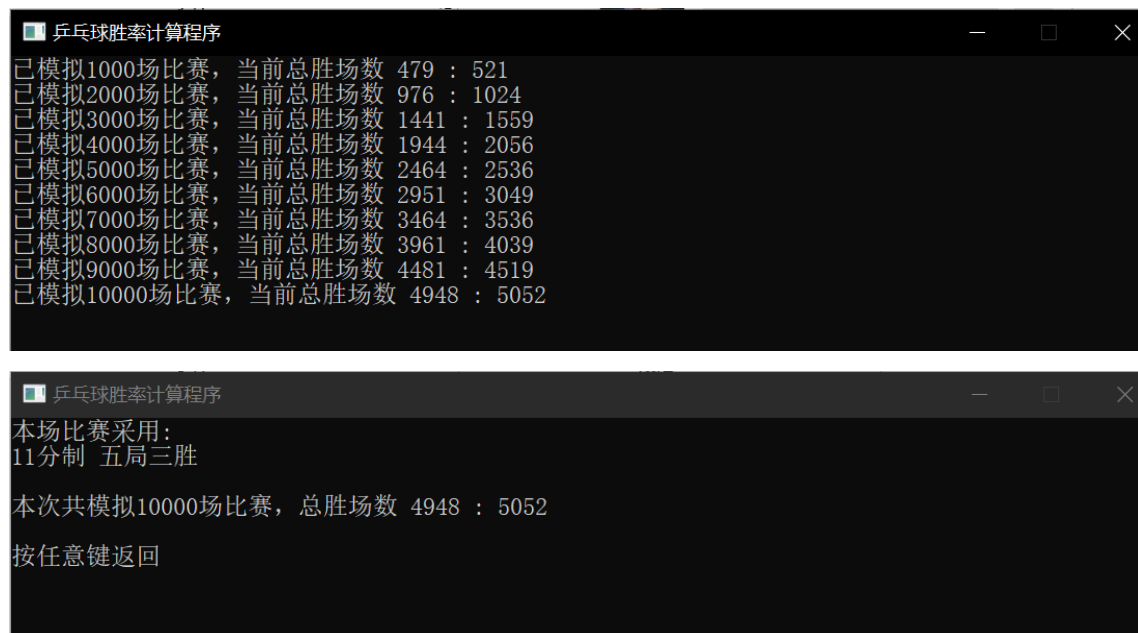


图 4-5 选手能力展示页

(4) 结果展示页 Page_Result 函数

功能：显示多场比赛的模拟结果（此时不显示比赛具体过程），演示模式下额外输出 11 分制和 21 分制的结果比较。

运行结果：



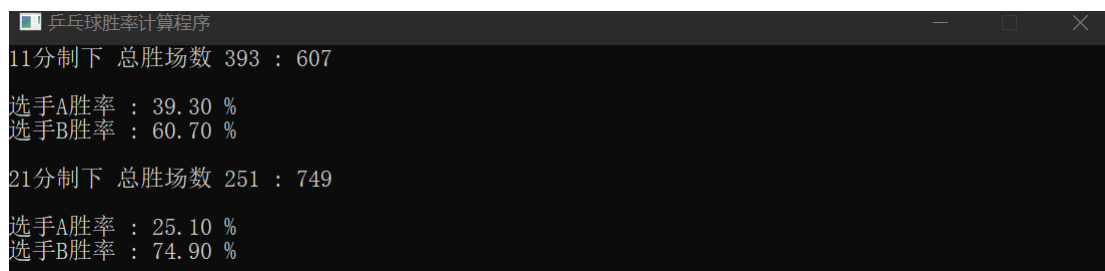


图 4-6 多场比赛模拟统计与对比

4.2 比赛模拟 (ppong 库内容介绍)

4.2.1 PlayerLevel 结构体

本程序使用结构体存储选手能力参数，便于数据管理。各数据含义请见 2.1 节。

```
struct PlayerLevel{
    float first_sev_scr; //发球直接得分率
    float atk_hit; //进攻时命中率（未命中则对手加分，命中则进行对方接球判定）
    float nor_hit; //非进攻时命中率
    float atk_catch; //受进攻时接球成功率
    float nor_catch; //受非进攻时接球成功率
    float sevatk_usg; //发球抢攻使用率（己方第二板）
    float sevctratk_usg; //发球反攻使用率（己方第二板）
    float cthatk_usg; //接球抢攻使用率（对方第一板）
    int stalemate_lv; //相持状态能力（按两方能力计算相持状态胜率）
    int syn_lv; //综合总能力
    float addi_hit=1; //心理状态和随机情况对命中率的加成倍率
    float addi_cth=1; //心理状态和随机情况对接球成功率的加成倍率
};
```

4.2.2 Game 类的数据成员与关键函数

本程序创建了一个类 Game，存放一场比赛的赛制数据、选手数据和比分数据与模拟比赛相关的成员函数。

赛制数据与选手数据作为用户可自定义的选项，为了简化赋值设为公有成员（可以利用友元函数改进）；比分数据作为私有成员，可避免类外函数的修改。

模拟比赛的函数除 Simulation 函数外均为私有函数，类的使用者将无法直接

访问包括击球函数、接球函数在内的具体模拟过程，后述函数都是 Simulation 的子过程。

Game 类的数据成员如下：

```
public:
    int calcmmod;
    int one_game; //1 为 11 分制，2 为 21 分制
    int whole_game; //3456 分别代表三局两胜/五局三胜/七局四胜/仅一局
    bool demo; //是否出于演示模式
    char strindex[2]={'A','B'};
    PlayerLevel plr[2]; //双方球员编号为 0/1
private:
    int og_score[2]; //小比分
    int wg_score[2]; //大比分
    int plr_hold; //持球方 (plr_hold=1-plr_hold 实现球权交换)
    int plr_sev; //发球方 (同上实现球权交换)
    int plr_alrsev; //当前发球方已发球个数
    int plr_firsev; //本局首发球方 (每局开始实现交换，第一次随机)
    双方球员的编号为 0 和 1，在球权交换时采用  $x=1-x$  的方式实现交换。
```

Game 类的全部成员函数如下：

```
private:
    bool Random(float probab); //随机函数
    bool Checkover(); //判断比赛是否结束，函数返回 true 则结束
    void Hit(int phit,int mod); //击球函数
    void Catch(int pcth,int mod); //接球函数
    void Stalemate(); //相持函数
    void MoodandRand(); //处理选手心理状况+随机改变选手状态的函数
public:
    Game(int a=1,int b=4,int cm=1):one_game(a),whole_game(b),calcmmod(cm) //构造函数
    {
        memset(og_score,0,sizeof(og_score));
        memset(wg_score,0,sizeof(wg_score));
        demo=false;
    }
    int Simulation(); //模拟比赛函数
    void CalcsynLevel(PlayerLevel &p); //计算总能力值
};
```


Game 类中各函数调用的关系图（即执行 Simulation 函数后的流程）如下：

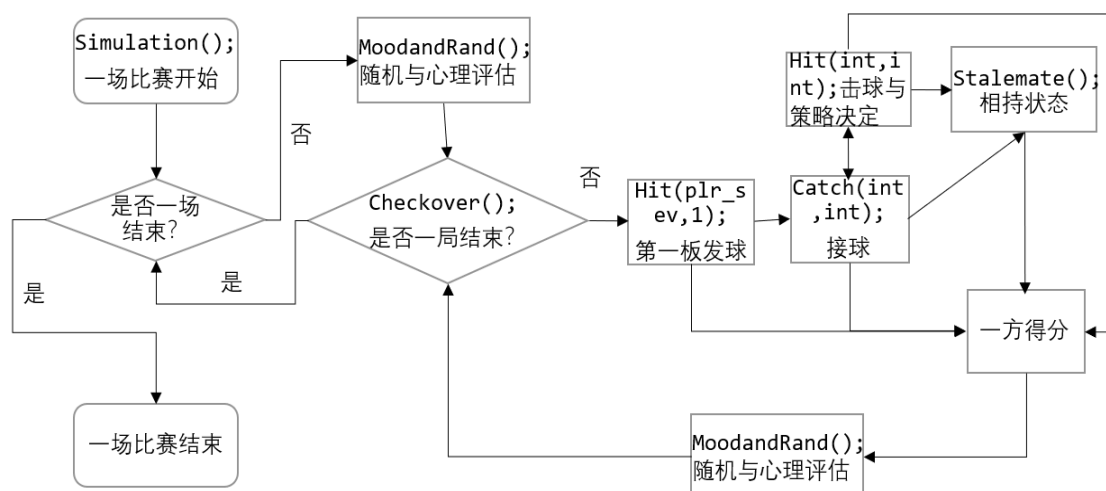


图 4-7 Game 中各函数调用的关系图（即执行 Simulation 函数后的流程图）

以下是部分关键函数的功能说明、代码及运行结果：

(1) Random 随机函数

功能：根据 prob 参数作为真的概率，使用 rand 函数随机并返回真假。（对应 sr and 在 main 函数中执行）

参数：float prob 概率（0-1 之间） 返回值：bool

```

bool Game::Random(float prob){
    float ran;
    ran= rand()/double(RAND_MAX);
    if(ran<=prob) return true;
    else return false;
}
  
```

(2) Simulation 模拟比赛主函数

功能：模拟一场比赛过程的主函数。包括：判断发球方->第一板发球->交换发球方->判断比赛结束。类的使用者只能调用此函数以开始比赛模拟。

返回值：int，为胜利选手的编号。

```

int Game::Simulation(){
    ..... //清空大小比分，根据赛制设定目标
    ..... //比赛开始抽签决定第一局发球方
    while(wg_score[0]<wggoal&&wg_score[1]<wggoal){//一场比赛开始
  
```

```

memset(og_score,0,sizeof(og_score)); //清空小比分
//交换该局首发球方（第一次也交换，由于概率相等不影响）
plr_firsev=1-plr_firsev;
plr_sev=plr_firsev;
//一局比赛开始
MoodandRand();
while(!Checkover()){
    ..... //演示模式输出赛制说明
    Hit(plr_sev,1); //第一板发球
    //计算是否更改发球方
    if(og_score[0]>=oggoal-1&&og_score[1]>=oggoal-1)
        ..... //进入赛点后一分一换
    else ..... //进入赛点后两分一换
        MoodandRand();//随机下一轮心理状态
    } //一局比赛结束
} //一场比赛结束
..... //演示模式输出结果
..... //返回胜负方
}

```

(3) Checkover 函数

功能：根据比分判断一局比赛是否结束

返回值：bool，真/假分别表示是/否结束

```

bool Game::Checkover(){
    int t;
    if(abs(og_score[0]-og_score[1])<=1||
        (og_score[0]>og_score[1]?og_score[0]:og_score[1])<(one_
game==1?11:21))
        return false;
    t=(og_score[0]>og_score[1]?0:1);
    wg_score[t]++;
    if(demo)..... //演示模式下输出
    return true;
}

```

(4) MoodandRand 函数

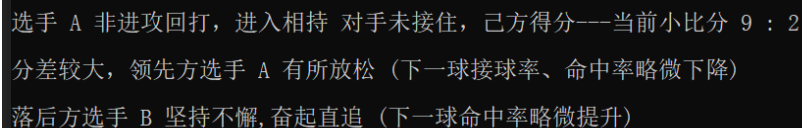
功能：为更好模拟比赛的随机性与紧张氛围，以上各命中率与接球成功率的具体值会有所变化。

每次发球前都将产生介于 0.95-1.1 的数值 1。当二者比分差大于等于 3 时触发条件，领先者有很小概率“有所放松”（下一球接球率、命中率略微下降），落后者有较小概率“坚持不懈，奋起直追”（下一球命中率略微提升），心理影响量化为数值 2。对命中率和接球成功率分别计算各自的数值 1*数值 2，作为额外倍率。额外倍率将和选手开始比赛前输入的基础倍率共同作用。

```
void Game::MoodandRand(){
    ..... //将命中率和接球成功率随机乘 0.95-1.1 倍
    for(int i=0;i<=1;i++){
        ..... //防止概率超过 100%
    }
    ..... //如果该局比赛已经结束则返回（防止演示模式下结束后多输出一
    次心理变化）
    return;
    int plead=(og_score[0]>og_score[1]?0:1);
    if (og_score[plead]-og_score[1-plead]>=3){
        if(Random(0.125)){
            ..... //领先方选手下一球接球率、命中率略微下降
        }
        if(Random(0.333)){
            ..... //落后方选手下一球命中率略微提升
            ..... //防止概率超过 100%
        }
    }
}
```

运行结果：

演示模式下，将会输出分差较大时出现的选手心理变化。



```
选手 A 非进攻回打，进入相持 对手未接住，己方得分——当前小比分 9 : 2
分差较大，领先方选手 A 有所放松（下一球接球率、命中率略微下降）
落后方选手 B 坚持不懈，奋起直追（下一球命中率略微提升）
```

图 4-8 选手心理变化输出

(5) Hit 击球函数

功能：击球函数，包含一局比赛中几乎所有击球动作。

击球时根据击球模式，使用进攻或非进攻下的命中率判断是否命中，命中则

转至接球方的 Catch 函数，未命中则接球方加分并重新发球。

选手进攻（非进攻）命中率=基础进攻（非进攻）命中率*额外倍率 1（由 MoodandRand 函数生成）

参数：int phit 击球者编号（球由选手 phit 被击给选手（1-phit））

Int mod 击球模式 0 为常规非进攻发球,1 为第一板发球,2 为接抢,3 为发球抢,4 为发球反攻,5 为接发球

```
void Game::Hit(int phit,int mod){
    if(demo)..... //演示模式输出

    if (mod==1){ //第一板发球
        if(Random(plr[phit].first_sev_scr))
            ..... //讨论是否直接得分，得分则己方加分
    }
    //讨论是否命中，若未命中
    if(mod>=2&&mod<=4){
        if(!Random(plr[phit].atk_hit*plr[phit].addi_hit))
            ..... //未命中，对手加分
    }
    else if(!Random(plr[phit].nor_hit*plr[phit].addi_hit))
        ..... //未命中，对手加分
    //命中后转到对手的 Catch 接球函数
    Catch(1-phit,mod);
    return;
}
```

运行结果：

演示模式下，将会输出具体的击球与接球过程。

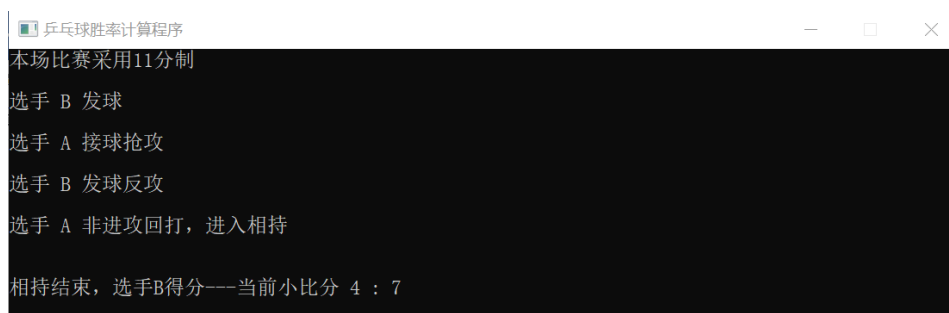




图 4-9 击球与接球过程

(6) Catch 接球函数

功能：接球函数，当击球函数中击球方命中后焦点转到接球方的接球函数。

接球时先判断是否成功接球，**选手进攻（非进攻）接球成功率=基础进攻（非进攻）接球成功率*额外倍率 2（由 MoodandRand 函数生成）。**

若接球失败，则击球方得分。

若接球成功，则**根据对应击球模式决定对策**。如击球方是第一板发球，根据接球者的接抢使用率随机使用接抢或接发球。然后接球方作为新的击球方并**再次调用 Hit 击球函数**；如击球方接球抢攻，则接球者随机使用发球反攻或非进攻击球；如击球方接发球，则接球者随机使用发球抢攻或非进攻击球。

当接到发球抢攻、发球反攻或决定非进攻击球后，下一回合进入相持。**调用 Stalemate 相持函数**，由双方选手的相持能力产生概率，随机该球的得分方。

参数：int pcth 接球者编号

Int mod 击球者的击球模式 0 为常规非进攻发球,1 为第一板发球,2 为接抢,3 为发抢,4 为发球反攻,5 为接发球

```
void Game::Catch(int pcth,int mod){
    //是否接住，先讨论未接住
    if(mod>=2&&mod<=4){
        if(!Random(plr[pcth].atk_catch*plr[pcth].addi_cth))
            ..... //未接住，击球方得分
    }
    else if(!Random(plr[pcth].nor_catch*plr[pcth].addi_cth))
        ..... //未接住，击球方得分
    //接住后判断对策
    if(demo) printf("\n");
    switch (mod){
        case 0:{Stalemate(); return;} //直接进入相持
```

```

    case 1:{ //接到对方发球
        if(Random(plr[pcth].cthatk_usg)) Hit(pcth,2); //接抢
        else Hit(pcth,5); //接发球
        break;
    }
    case 2:{ //接到对方接抢
        if(Random(plr[pcth].sevctratk_usg)) Hit(pcth,4); //
发球反攻
        else Hit(pcth,0); //非进攻回发, 下一回合进入相持
        break;
    }
    case 3: Hit(pcth,0); break;
    case 4: Hit(pcth,0); break;
    case 5:{ //接到对方接发球
        if(Random(plr[pcth].sevatk_usg)) Hit(pcth,3); //发球
抢攻
        else Hit(pcth,0); //非进攻回发, 下一回合进入相持
        break;
    }
}
}
}

```

运行结果：演示模式下，将会输出具体的击球与接球过程（见图 4-9）

5.结果分析

5.1 赛制改变对比赛结果的影响

5.1.1 二项分布法理论计算

在使用程序模拟之前，可以首先使用二项分布原理简单计算一局比赛的胜率。假设有两位乒乓球运动员，A 每球得分的概率为 p ($0 < p < 1$)，B 每球得分的概率为 $1-p$ 。在 21 分制下，A 获胜的可能为 21:0, 21:1, 21:2, ……，21:19, 21:20（为简化计算，到达 21:20 时即算 A 获胜）。

设 A 以 21:0 的情况获胜的概率为 P_1 ，A 以 21:1 的情况获胜的概率为 P_2 ……A 以 21:20 获胜的概率为 P_{21} 。以 $p=0.55$ 计算，各概率如下表^[3]：

P_1	p^{21}	$(0.55)^{21}$ (这一列都是 $p=0.55$ 时的概率)
P_2	$[C_{21}^{20} \cdot p^{20} \cdot (1-p)] \cdot p =$ $C_{21}^{20} \cdot (1-p) \cdot p^{21}$	$[C_{21}^{20} \times 0.55^{20} \times (1-0.55)] \times 0.55 =$ $C_{21}^{20} \times (1-0.55) \times 0.55^{21}$
P_3	$[C_{21}^{20} \cdot p^{20} \cdot (1-p)^2] \cdot p =$ $C_{21}^{20} \cdot (1-p)^2 \cdot p^{21}$	$[C_{21}^{20} \times 0.55^{20} \times (1-0.55)^2] \times 0.55 =$ $C_{21}^{20} \times (1-0.55)^2 \times 0.55^{21}$
...
P_{21}	$[C_{21}^{20} \cdot p^{20} \cdot (1-p)^{20}] \cdot p =$ $C_{21}^{20} \cdot (1-p)^{20} \cdot p^{21}$	$[C_{21}^{20} \times 0.55^{20} \times (1-0.55)^{20}] \times 0.55 =$ $C_{21}^{20} \times (1-0.55)^{20} \times 0.55^{21}$

表 5-1 二项分布计算 21 分制下 A 选手获胜概率

21 分制下 A 的总获胜概率 $P=P_1+P_2+\cdots+P_{21}$, 当 $p=0.55$ 时 $P \approx 0.74$ 。

同理, 考虑 11 分制, 当 $P=P_1+P_2+\cdots+P_{11}$, 当 $p=0.55$ 时 $P \approx 0.68$ 。

假设每球得分率高的选手能力高, 将 21 分制和 11 分制下的比赛结果进行比较, 发现赛制改变后能力高的选手胜率仍然较高, 但数值有所下降。多次改变 p 的数值并计算, 可以得到相同的结果。

5.1.2 程序模拟法统计

接下来我们采用“乒乓球比赛模拟与胜率计算程序”计算。首先使用演示计算功能随机选手 A 和选手 B 的能力参数如下:



图 5-1 随机选手能力参数

经过 1000 场比赛模拟，程序输出 11 分制和 21 分制下（按奥运会单人淘汰赛标准，11 分制下选择七局四胜制，21 分制下选择三局两胜制）选手胜率如下图所示：

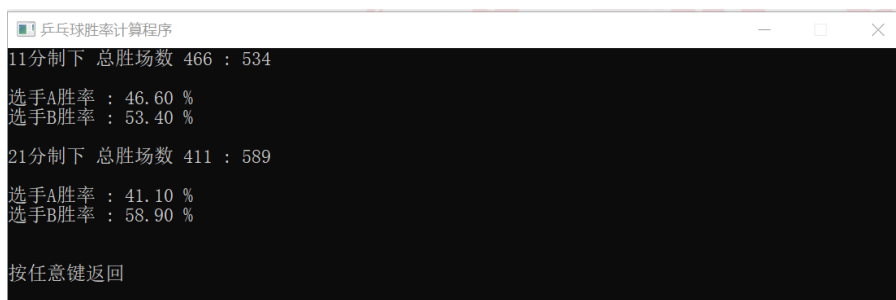


图 5-2 随机结果

假设每球得分率高的选手能力高，此次模拟中可视作选手 B 能力较强。从 21 分制变换为 11 分制时，领先选手 B 的胜率下降了 5.5%，可以看出和二项分布计算得到的结果大致相同——**能力高的选手胜率有所下降**。

为探究 21 分制胜率与赛制变化后胜率变化率的关系，多次运用演示计算功能随机选手参数并模拟比赛，得到领先方 21 分制下的胜率和 11 分制下的胜率，将得到的数据作图如下：

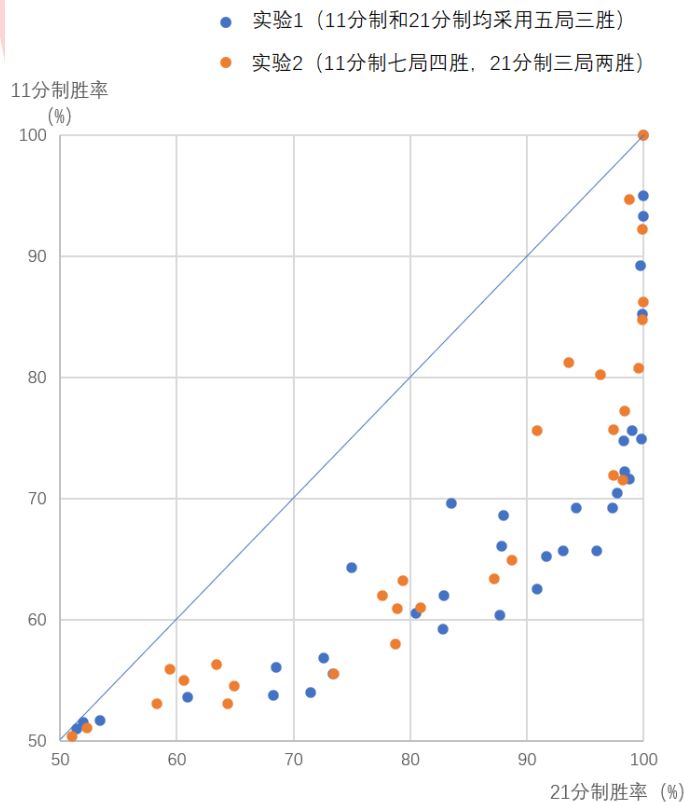


图 5-3 数据统计散点图

从上图可以看出,假设 21 分制下胜率越高的选手能力越强,则:

在一定范围内,双方能力差距越大,赛制改变对领先者胜率的降低作用越显著。即原先 21 分制下胜率较高一方胜率越高,改至 11 分制后该选手胜率仍处于较高一方,但数值降低得越多。

双方能力差距极大时,赛制改变的胜率降低作用将迅速下降,此时可以视为两位选手的水平不在一个数量级。

与二项分布理论计算不同,通过程序模拟比赛,赛制改变后胜率的差距更大。

因选手能力各项参数的权重有所不同,图中点的分布有一定分散性,但经过多次模拟实验,上图的大致趋势已经足以证明上述结论。

5.2 总结与启示

赛制改变会降低对战双方胜率的差距。双方能力越悬殊,赛制改变对领先者胜率的降低作用就越显著。缓解了乒乓球比赛中“强者恒强”的情况,增加了比赛结果的随机性与比赛的可观看性。

中国乒乓球队的实力领先外国选手较多,因此,赛制从 21 分制变为 11 分制,会降低中国队选手的胜率,缩小领先地位。中国队选手应该探索新赛制下的新战术,继续磨练自身水平,延续辉煌。

6.参考文献

- [1] 李今亮,苏丕仁.对部分世界优秀男子乒乓球进攻型选的评估——兼谈十项指标评估法的建立[J].北京:北京体育大学学报,1998.
- [2] 栗磊.对中国优秀选手马龙的技战术特征分析[D].北京:北京体育大学,2010.
- [3] 罗远华.用二项分布原理分析“每局 11 分制”对中国乒乓球队的影响[J].北京:高中数理化,2021.

附录 1 程序运行环境

程序在 Windows 平台下运行，需额外使用的动态链接库包含：

libgcc_s_seh-1.dll

libstdc++-6.dll

libwinpthread-1.dll

附录 2 全部代码

main.cpp 中的代码

```
#include "cmd_opts.h" //控制台相关操作
#include "ppong.h" //模拟比赛
#include <ctime>
#include <cmath>
#include <cstdio>
#include <iostream>
#include <iomanip>
/* 一些关于窗口位置的参数
    窗口大小 X=90,Y=30
    左侧列 X=10 选择符号 X=8
    右侧菜单列 X=40 (参数详细设置) 选择符号 X=38
    最右侧列 X=65 (退格和下一步)
    顶行 Y=3
*/
void InitializeWindow(); //窗口初始化
void Page_MainMenu(); //主菜单页
void Page_Custom_Paraset(); //自定义计算参数设置
void Page_Custom_Paraset_RMenu(const int index); //自定义计算参数界面右侧菜单 (此函数只负责显示)
void Page_Custom_Paraset_Input(PlayerLevel &y,int id); //自定义选手参数输入
bool Page_CustomII();
void Page_Demo(); //演示计算 I
void Page_DemoII(); //演示计算 II
void ShowPlrPara(int i); //显示选手参数 (子函数)
void Page_Result(); //显示结果
string to_string2(float x); //保留两位小数的将 0-1 之间的小数转为两位

Game usergame; //用户定义赛制

int main()
{
    InitializeWindow();
    srand(time(NULL));
    while(1){
        Page_MainMenu();
    }
    system("pause");
    return 0;
}
```

```

}

void InitializeWindow(){
    copts_setconsoleborder(90, 30, 90, 30);
    copts_setconsoletitle("乒乓球胜率计算程序");
    copts_fixconsolesize();
    copts_hidecursor();
}

void Page_MainMenu(){
    int keyin;
    copts_cls();
    copts_printxy(10,5,"乒乓球胜率计算程序");
    copts_printxy(10,7,"Developed by UNikeEN");
    copts_printxy(10,9,"-----");
    copts_printxy(10,11,"[1] 开始新的自定义计算");
    copts_printxy(10,13,"[2] 预设计算演示");
    copts_printxy(10,15,"[ESC] 退出本程序");
    copts_printxy(65,29,"Released_1.2.1_20211217");
    while(1){
        keyin=_getch()-48;
        if (keyin==21) exit(0); //ESC 27-48
        if (keyin==1||keyin==2) break;
    }
    if (keyin==1) Page_Custom_Paraset();
    else if (keyin==2) Page_Demo();
    return;
}

void Page_Custom_Paraset(){
    para:
    int keyin,select_index=1;
    copts_cls();
    copts_printxy(10,3,"[Backspace] 上一级菜单");
    copts_printxy(65,3,"[ESC] 退出本程序");
    copts_printline(10,5,70);
    copts_printxy(10,7,"在这里修改有关选手、比赛、计算模型的相关参数");
    copts_printxy(10,9,"(按方向键选择一级菜单,按数字键选择二级菜单)");
    copts_printxy(65,27,"[N] 下一步");
    string Paralists[5]=
        {"","I.选手参数","II.赛制参数","III.计算模型"};
    for(int i=1;i<=3;i++){
        copts_printxy(10,9+2*i,Paralists[i]);
    }
    copts_printxy(8,9+2*select_index,"> ");
    Page_Custom_Paraset_RMenu(select_index);
    while(1){
        keyin=_getch();
        if (keyin==27) exit(0); //ESC 27
        if (keyin==8) return; //退格 8
        else if(keyin==78||keyin==110){ //N 78 n 110 开始
            usergame.demo=false;
            bool ret;
            ret=Page_CustomII();
            if(ret) goto para;
            else return;
        }
    }
}

```

```

else if (keyin==(224)){ //方向键控制左侧列表上下
    int dirct;
    dirct=_getch();
    switch (dirct){
    case 72:{
        if(select_index>1) select_index--;
        break;
    }
    case 80:{
        if(select_index<3) select_index++;
        break;
    }
    }
    for(int i=1;i<=3;i++) copts_printxy(8,9+2*i," ");
    copts_printxy(8,9+2*select_index,"> ");
    Page_Custom_Paraset_RMenu(select_index);
}
else if(keyin>=48&&keyin<=57){ //数字键选择二级菜单
    switch(select_index){
    case 1:{
        switch(keyin){
            case 49:Page_Custom_Paraset_Input(usergame.plr[0],0);goto para;break;
            case 50:Page_Custom_Paraset_Input(usergame.plr[1],1);goto para;break;
            case 51:{
                usergame.plr[1]=usergame.plr[0];
                goto para;
                break;
            }
        }
    }
    case 2:{
        if(keyin>=49&&keyin<=50){
            for(int i=1;i<=2;i++) copts_printxy(38,9+2*i," ");
            copts_printxy(38,9+2*(keyin-48),"> ");
            usergame.one_game=keyin-48;
        }
        else if(keyin>=51&&keyin<=54){
            for(int i=3;i<=6;i++) copts_printxy(38,9+2*i," ");
            copts_printxy(38,9+2*(keyin-48),"> ");
            usergame.whole_game=keyin-48;
        }
        break;
    }
    case 3:break;
    }
}
}

void Page_Custom_Paraset_RMenu(const int index){
    for(int i=11;i<25;i++) copts_printspace(40,i,52);
    for(int i=11;i<25;i++) copts_printxy(38,i," ");
    switch (index){
    case 1:{
        copts_printxy(40,11,"[1] 修改选手 A 能力参数");
        copts_printxy(40,13,"[2] 修改选手 B 能力参数");
        copts_printxy(40,15,"[3] 使选手 B 能力参数与 A 一致");
    }
    }
}

```

```

        break;
    }
    case 2:{
        copts_printxy(40,11,"[1] 11 分制");
        copts_printxy(40,13,"[2] 21 分制");
        copts_printxy(40,15,"[3] 三局两胜制");
        copts_printxy(40,17,"[4] 五局三胜制");
        copts_printxy(40,19,"[5] 七局四胜制");
        copts_printxy(40,21,"[6] 一局定胜负");
        copts_printxy(38,9+2*usergame.one_game,"> ");
        copts_printxy(38,9+2*usergame.whole_game,"> ");
        break;
    }
    case 3:{
        copts_printxy(40,11,"[1] 仿真比赛");
        copts_printxy(38,9+2*usergame.calcmod,"> ");
        break;
    }
}
}

void Page_Custom_Paraset_Input(PlayerLevel &y,int id){
    PlayerLevel *p=&y;
    copts_showcursor();
    copts_cls();
    printf("请输入选手%c 的参数(请保证数据合法, 本程序不做判断)\n",usergame.strindex[id]);
    printf("请输入发球直接得分率(0<x<1):\n");
    scanf("%f",&p->first_sev_scr);
    printf("请输入进攻时命中率(0<x<1):\n");
    scanf("%f",&p->atk_hit);
    printf("请输入非进攻时命中率(0<x<1):\n");
    scanf("%f",&p->nor_hit);
    printf("请输入受进攻接球成功率(0<x<1):\n");
    scanf("%f",&p->atk_catch);
    printf("请输入非受进攻接球成功率(0<x<1):\n");
    scanf("%f",&p->nor_catch);
    printf("请输入发球抢攻使用率(0<x<1):\n");
    scanf("%f",&p->sevatk_usg);
    printf("请输入发球反攻使用率(0<x<1):\n");
    scanf("%f",&p->sevctratk_usg);
    printf("请输入接球抢攻使用率(0<x<1):\n");
    scanf("%f",&p->cthatk_usg);
    printf("请输入相持状态能力(0<x<100):\n");
    scanf("%d",&p->stalemate_lv);
    int keyin;
    printf("\n 输入完成, 按任意键继续\n");
    keyin=_getch();
    copts_hidecursor();
    return;
}

bool Page_CustomII(){
    int keyin;
    copts_cls();
    copts_printxy(10,3,"[Backspace] 上一级菜单");
    copts_printxy(65,3,"[ESC] 退出本程序");
    copts_printxy(65,27,"[N] 开始计算");
    copts_printline(10,5,70);

```

```

copts_printxy(10,7,"选手 A");
copts_printxy(40,7,"选手 B");
//随机生成两位选手数值
PlayerLevel *p;
for(int i=0;i<=1;i++){
    p=&usergame.plr[i];
    ShowPlrPara(i);
}
while(1){
    keyin=_getch();
    if (keyin==27) exit(0); //ESC 27
    if (keyin==8) return 1; //退格 8
    else if(keyin==78||keyin==110){ //N 78 n 110 开始
        usergame.demo=false;
        Page_Result();
        return 0;
    }
}
}
void Page_Demo(){
    int keyin;
    copts_cls();
    copts_printxy(10,3,"[Backspace] 上一级菜单");
    copts_printxy(65,3,"[ESC] 退出本程序");
    copts_printxy(65,27,"[N] 开始计算");
    copts_printline(10,5,70);
    copts_printxy(10,7,"第一步 随机生成选手能力 请稍后");
    Sleep(500);
    copts_printxy(10,7,"选手 A");
    copts_printxy(40,7,"选手 B");
    //随机生成两位选手数值
    PlayerLevel *p;
    for(int i=0;i<=1;i++){
        p=&usergame.plr[i];
        p->first_sev_scr=RandomVal(0.05,0.16);
        p->atk_hit=RandomVal(0.65,0.9);
        p->nor_hit=RandomVal(0.9,0.98);
        p->atk_catch=RandomVal(0.65,0.75);
        p->nor_catch=RandomVal(0.9,0.95);
        p->sevatk_usg=RandomVal(0.3,0.55);
        p->sevctratk_usg=RandomVal(0.3,0.55);
        p->cthatk_usg=RandomVal(0.3,0.45);
        p->stalemate_lv=(int)(RandomVal(0.45,0.55)*100);
        ShowPlrPara(i);
    }
    while(1){
        keyin=_getch();
        if (keyin==27) exit(0); //ESC 27
        if (keyin==8) return; //退格 8
        else if(keyin==78||keyin==110){ //N 78 n 110 开始
            //Page_DemoII();
            usergame.demo=true;
            Page_DemoII();
            return;
        }
    }
}
}

```



```

void ShowPlrPara(int i){
    PlayerLevel *p=&usergame.plr[i];
    copts_printxy(10+30*i,9,"发球得分率:"+to_string2(p->first_sev_scr));
    copts_printxy(10+30*i,10,"进攻命中率:"+to_string2(p->atk_hit));
    copts_printxy(10+30*i,11,"非进攻命中率:"+to_string2(p->nor_hit));
    copts_printxy(10+30*i,12,"受进攻接球率:"+to_string2(p->atk_catch));
    copts_printxy(10+30*i,13,"非受进攻接球率:"+to_string2(p->nor_catch));
    copts_printxy(10+30*i,14,"发抢使用率:"+to_string2(p->sevatk_usg));
    copts_printxy(10+30*i,15,"发球反攻使用率:"+to_string2(p->sevctratk_usg));
    copts_printxy(10+30*i,16,"接抢使用率:"+to_string2(p->cthatk_usg));
    copts_printxy(10+30*i,17,"相持得分能力:"+to_string(p->stalemate_lv));
    copts_printxy(10+30*i,19,"综合能力评分:"+to_string(p->syn_lv));
    usergame.CalcsynLevel(*p);
    copts_printxy(10+30*i,19,"综合能力评分:"+to_string(p->syn_lv));
}

void Page_DemoII(){
    int keyin;
    copts_cls();
    copts_printxy(10,7,"第二步 模拟一场比赛（一局定胜制）\n");
    Sleep(1000);
    usergame.demo=true;
    usergame.one_game=1;
    usergame.whole_game=6;
    int tmp=usergame.Simulation();
    printf("\n 按任意键进入下一页\n");
    keyin=_getch();
    Page_Result();
    return;
}

void Page_Result(){
    int keyin;
    copts_cls();
    copts_hidecursor();
    if(usergame.demo){
        usergame.one_game=1;
        usergame.whole_game=5;
        copts_printxy(10,7,"第三步 模拟一千场比赛(11分制,七局四胜)");
        Sleep(1000);
        copts_cls();
        copts_gotoxy(0,0);
        usergame.demo=0;
        int tot[2]={0};
        for(int i=1;i<=1000;i++){
            int tmp=usergame.Simulation();
            tot[tmp]++;
            printf("已模拟%d场比赛,当前总胜场数 %d : %d \n",i,tot[0],tot[1]);
            Sleep(5);
        }
        printf("\n 按任意键进入下一页\n");
        keyin=_getch();
        usergame.one_game=2;
        usergame.whole_game=3;
        copts_cls();
        copts_printxy(10,7,"第四步 模拟一千场比赛(21分制,三局两胜)");
        Sleep(1000);
        copts_cls();
        usergame.demo=0;
    }
}

```

```

int tot2[2]={0};
for(int i=1;i<=1000;i++){
    int tmp=usergame.Simulation();
    tot2[tmp]++;
    printf("已模拟%d 场比赛, 当前总胜场数 %d : %d \n",i,tot2[0],tot2[1]);
    Sleep(5);
}
printf("\n 按任意键进入下一页\n");
keyin=_getch();
copts_cls();
printf("11 分制 (七局四胜) 下 总胜场数 %d : %d \n\n",tot[0],tot[1]);
printf("选手 A 胜率 : %.21f %\n", (100*(double)tot[0]/(tot[0]+tot[1])));
printf("选手 B 胜率 : %.21f %\n\n", (100*(double)tot[1]/(tot[0]+tot[1])));
printf("21 分制 (三局两胜) 下 总胜场数 %d : %d \n\n",tot2[0],tot2[1]);
printf("选手 A 胜率 : %.21f %\n", (100*(double)tot2[0]/(tot2[0]+tot2[1])));
printf("选手 B 胜率 : %.21f %\n\n", (100*(double)tot2[1]/(tot2[0]+tot2[1])));
printf("\n 按任意键返回\n");
keyin=_getch();
return;
}
else{
    int tot[2]={0};
    for(int i=1;i<=10000;i++){
        int tmp=usergame.Simulation();
        tot[tmp]++;
        if(i%1000==0) printf("已模拟%d 场比赛, 当前总胜场数 %d : %d \n",i,tot[0],tot[1]);
    }
    Sleep(500);
    copts_cls();
    if(usergame.one_game==1)printf("本场比赛采用:\n11 分制 ");
    else printf("本场比赛采用:\n21 分制 ");
    switch (usergame.whole_game){
        case 3:printf("三局两胜\n\n");break;
        case 4:printf("五局三胜\n\n");break;
        case 5:printf("七局四胜\n\n");break;
        case 6:printf("一局定胜\n\n");break;
    }
    printf("本次共模拟%d 场比赛, 总胜场数 %d : %d \n",10000,tot[0],tot[1]);
    printf("\n 选手 A 胜率 : %.21f %\n", (100*(double)tot[0]/(tot[0]+tot[1])));
    printf("选手 B 胜率 : %.21f %\n", (100*(double)tot[1]/(tot[0]+tot[1])));
    printf("\n 按任意键返回\n");
    keyin=_getch();
    return;
}
}

string to_string2(float x){
    if(x<0||x>1) return("ERROR");
    if(x==1) return("1.00");
    int a=(int)(x*100);
    if(a<10) return ("0.0"+to_string(a));
    return ("0."+to_string(a));
}

```

/lib/cmd_opts.h 中的代码

```

/*关于控制台的相关操作*/
#pragma once
#include <string>
#include <conio.h>
#include <windows.h>
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;
/*****
    函数名称: copts_setconsoletitle
    功    能: 设置 cmd 窗口标题
    *****/
void copts_setconsoletitle(const char *title);

/*****
    函数名称: copts_fixedconsolesize
    功    能: 隐藏标题栏最大化按钮, 禁止边框拖动改变窗口大小
    *****/
void copts_fixconsolesize(void);

/*****
    函数名称: copts_cls
    功    能: 清屏
    *****/
void copts_cls(void);

/*****
    函数名称: copts_gotoxy
    功    能: 将光标移动到指定位置
    输入参数: const int X      : X 轴坐标 (列)
               const int Y      : Y 轴坐标 (行)
    说    明: 屏幕左上角坐标为(0,0), 在 cmd 窗口的大小未被调整的情况下, Win10 为:
               横向 x 轴, 对应列(0-119)
               纵向 y 轴, 对应行(0-29)
    *****/
void copts_gotoxy(const int X, const int Y);

/*****
    函数名称: copts_printxy
    功    能: 在指定位置输出指定颜色字符串
    输入参数: const int &X, const int &Y 行列
               const string &str 输出字符串
    *****/
void copts_printxy(const int &X, const int &Y, std::string str);

/*****
    函数名称: copts_printspace
    功    能: 在指定位置输出空格
    输入参数: const int &X, const int &Y 行列
               const int &rpt 输出次数 (长度)
    *****/
void copts_printspace(const int &X, const int &Y, const int &rpt);

/*****
    函数名称: copts_printline
    功    能: 在指定位置输出一条线

```

```

    输入参数: const int &X, const int &Y 行列
               const int &rpt 输出次数 (长度)
    *****/
void copts_printline(const int &X,const int &Y,const int &rpt);

/*****
    函数名称: copts_setconsoleborder
    功    能: 改变 cmd 窗口的大小及缓冲区的大小
    输入参数: const int cols      : 新的列数
               const int lines     : 新的行数
               const int buffer_cols : 新的缓冲区列数
               const int buffer_lines : 新的缓冲区行数
    说    明: 必须先设置缓冲区, 再设置窗口大小,
    *****/
void copts_setconsoleborder(int set_cols, int set_lines, int set_buffer_cols, int set_buffer_lines);

/*****
    函数名称: copts_hidecursor
    功    能: 隐藏光标
    *****/
void copts_hidecursor();

/*****
    函数名称: copts_hidecursor
    功    能: 显示光标 (横线)
    *****/
void copts_showcursor();

```

/src/cmd_opts.cpp 中的代码

```

#include "cmd_opts.h"
using namespace std;

static const HANDLE __hout = GetStdHandle(STD_OUTPUT_HANDLE); //取标准输出设备对应的句柄
static const HANDLE __hin  = GetStdHandle(STD_INPUT_HANDLE);  //取标准输入设备对应的句柄

void copts_setconsoletitle(const char *title)
{
    SetConsoleTitleA(title);
}

void copts_cls(void){
    system("cls");
}

void copts_fixconsolesize(void){
    SetWindowLongPtrA(
        GetConsoleWindow(),
        GWL_STYLE,
        GetWindowLongPtrA(GetConsoleWindow(),GWL_STYLE)
        & ~WS_SIZEBOX & ~WS_MAXIMIZEBOX );
}

void copts_gotoxy(const int X, const int Y)
{
    COORD coord;
    coord.X = X;
    coord.Y = Y;
}

```

```

        SetConsoleCursorPosition(__hout, coord);
    }
    void copts_printxy(const int &X,const int &Y,string str)
    {
        copts_gotoxy(X,Y);
        cout<<str;
    }
    void copts_printspace(const int &X,const int &Y,const int &rpt)
    {
        copts_gotoxy(X,Y);
        for(int i=1;i<=rpt;i++) cout<<' ';
    }
    void copts_printline(const int &X,const int &Y,const int &rpt)
    {
        copts_gotoxy(X,Y);
        for(int i=1;i<=rpt;i++) cout<<'-' ;
    }
    void copts_setconsoleborder(int set_cols, int set_lines, int set_buffer_cols, int set_buffer_lines)
    {
        /* 取当前系统允许的窗口的行列最大值 */
        COORD max_coord;
        max_coord = GetLargestConsoleWindowSize(__hout); /* .X 和 .Y 分别是窗口的列和行的最大值 */
        /* 设置窗口的行列大小 (从 0 开始, 0 ~ lines-1, 0 ~ cols-1) */
        SMALL_RECT rect;
        rect.Top = 0;
        rect.Bottom = set_lines - 1;
        rect.Left = 0;
        rect.Right = set_cols - 1;
        /* 设置缓冲区的行列大小(缺省或小于窗口值则与窗口值一样) */
        COORD cr;
        cr.X = (set_buffer_cols == -1 || set_buffer_cols < set_cols) ? set_cols : set_buffer_cols;
        //未给出或给出的值小于 set_cols 则用 set_cols, 未控制上限
        cr.Y = (set_buffer_lines == -1 || set_buffer_lines < set_lines) ? set_lines : set_buffer_lines;
        //未给出或给出的值小于 set_lines 则用 set_lines, 未控制上限
        /* 取当前窗口及缓冲区的大小 */
        int cur_cols, cur_lines, cur_buffer_cols, cur_buffer_lines;
        CONSOLE_SCREEN_BUFFER_INFO binfo;
        GetConsoleScreenBufferInfo(__hout, &binfo);

        cur_cols = binfo.srWindow.Right - binfo.srWindow.Left + 1; //可见窗口的列数
        cur_lines = binfo.srWindow.Bottom - binfo.srWindow.Top + 1; //可见窗口的行数
        cur_buffer_cols = binfo.dwSize.X; //缓冲区的列数
        cur_buffer_lines = binfo.dwSize.Y; //缓冲区的行数

        copts_cls();
        /* 设置窗口大小时, 现缓冲区的列值要确保窗口值 */
        if (cr.X <= cur_buffer_cols) {
            if (cr.Y <= cur_buffer_lines) {
                SetConsoleWindowInfo(__hout, true, &rect); //设置窗口
                SetConsoleScreenBufferSize(__hout, cr); //设置缓冲区
            }
            else {
                COORD tmpcr;
                tmpcr.X = cur_buffer_cols;
                tmpcr.Y = cr.Y;
                SetConsoleScreenBufferSize(__hout, tmpcr); //设置缓冲区
            }
        }
    }

```

```

        SetConsoleWindowInfo(__hout, true, &rect); //设置窗口
        SetConsoleScreenBufferSize(__hout, cr); //设置缓冲区
    }
}
else {
    if (cr.Y >= cur_buffer_lines) {
        SetConsoleScreenBufferSize(__hout, cr); //设置缓冲区
        SetConsoleWindowInfo(__hout, true, &rect); //设置窗口
    }
    else {
        COORD tmpcr;
        tmpcr.X = cr.X;
        tmpcr.Y = cur_buffer_lines;
        SetConsoleScreenBufferSize(__hout, tmpcr); //设置缓冲区
        SetConsoleWindowInfo(__hout, true, &rect); //设置窗口
        SetConsoleScreenBufferSize(__hout, cr); //设置缓冲区
    }
}
return;
}
void copts_hidecursor(){
    CONSOLE_CURSOR_INFO tmp;
    tmp.bVisible = 0;
    tmp.dwSize = 1;
    SetConsoleCursorInfo(__hout, &tmp);
}
void copts_showcursor(){
    CONSOLE_CURSOR_INFO tmp;
    tmp.bVisible = 1;
    tmp.dwSize = 25;
    SetConsoleCursorInfo(__hout, &tmp);
}

```

/lib/ppong.h 中的代码

```

#pragma once
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <iostream>
#include "cmd_opts.h"
using namespace std;
struct PlayerLevel{
    float first_sev_scr; //发球直接得分率
    float atk_hit; //进攻时命中率（未命中则对手加分，命中则进行对方接球判定）
    float nor_hit; //非进攻时命中率
    float atk_catch; //受进攻时接球成功率
    float nor_catch; //受非进攻时接球成功率
    float sevatk_usg; //发球抢攻使用率（己方第二板）
    float sevctratk_usg; //发球反攻使用率（己方第二板）
    float cthatk_usg; //接球抢攻使用率（对方第一板）
    int stalemate_lv; //相持状态能力（按两方能力计算相持状态胜率）
    int syn_lv; //综合总能力
    float addi_hit=1; //心理状态和随机情况对命中率的加成本率
    float addi_cth=1; //心理状态和随机情况对接球成功率的加成本率
}

```

```

    int power=100; //体能剩余
};

//比赛类,数据成员小写开头, 成员函数大写开头
class Game{
public:
    int calcmob;
    int one_game;//1 为 11 分制, 2 为 21 分制
    int whole_game;//3456 代表三局两胜/五局三胜/七局四胜/仅一局
    bool demo; //是否演示每一局具体过程
    char strindex[2]={'A','B'};
    PlayerLevel plr[2]; //双方球员编号为 0/1

private:
    int og_score[2]; //小比分
    int wg_score[2]; //大比分
    int plr_hold; //持球方 (plr_hold=1-plr_hold 实现球权交换)
    int plr_sev; //发球方 (同上实现球权交换)
    int plr_alrsev; //当前发球方已发球个数
    int plr_firsev; //本局首发球方 (每局开始同上实现交换, 第一次随机)

    //随机函数, 根据 prob 参数作为真的概率, 随机并返回真假
    bool Random(float prob);

    //判断比赛是否结束, 函数返回 true 则结束
    bool Checkover();

    //发球函数, phit->(1-phit)
    //mod=0 为常规非进攻发球, 1 为第一板发球, 2 为接抢, 3 为发抢, 4 为发球反攻, 5 为接发球
    void Hit(int phit, int mod);

    //接球函数, 球员接到球后的思考过程, 决定策略后调用 Hit (是否接到球在上一个 Hit 中计算)
    //mod=0 为常规非进攻接球, 1 为接到对方发球, 2 为接到对方接抢, 3 为接到对方发抢, 4 为接到对方发球反攻
    void Catch(int pcth, int mod);

    //相持函数
    void Stalemate();

    //处理选手心理状况+
    //随机改变选手状态的函数
    void MoodandRand();
    /*心理影响
    当单内分差达到临界值 (设为 4) 时, 进行随机
    领先方有 1/8 概率放松 (接球率、命中率下降)
    失败方有 1/2 概率反追 (命中率提升) */

public:
    Game(int a=1, int b=4, int cm=1):one_game(a),whole_game(b),calcmob(cm)
    {
        memset(og_score, 0, sizeof(og_score));
        memset(wg_score, 0, sizeof(wg_score));
        demo=false;
    }
    //模拟比赛
    int Simulation();

    //计算总能力值

```



```

void CalcsynLevel(PlayerLevel &p);

};

float RandomVal(float l,float r); //随机数生成, 产生一个 l-r 的数( $0<l<r<1$ )

```

/src/ppong.cpp 中的代码

```

#include "ppong.h"

bool Game::Random(float prob){
    float ran;
    ran= rand()/double(RAND_MAX);
    if(ran<=prob) return true;
    else return false;
}

int Game::Simulation(){
    memset(og_score,0,sizeof(og_score));
    memset(wg_score,0,sizeof(wg_score));
    int wggoal; //wggoal 为一场比赛结束所需赢场次数
    int oggoal; //oggoal 为一局比赛结束所需赢小分数
    switch (whole_game){
        case 3:wggoal=2;break;
        case 4:wggoal=3;break;
        case 5:wggoal=4;break;
        case 6:wggoal=1;break;
    }
    switch (one_game){
        case 1:oggoal=11;break;
        case 2:wggoal=21;break;
    }
    //比赛开始抽签决定第一局发球方
    Random(0.5)?plr_sev=0:plr_sev=1;
    plr_alrsev=0;
    plr_firsev=plr_sev;
    //一场比赛开始
    while(wg_score[0]<wggoal&&wg_score[1]<wggoal)
    {
        //清空小比分
        memset(og_score,0,sizeof(og_score));
        //交换该局首发球方 (第一次页交换, 由于概率相等不影响)
        plr_firsev=1-plr_firsev;
        plr_sev=plr_firsev;
        //一局比赛开始
        MoodandRand();
        while(!Checkover()){
            if(demo){
                copts_cls();
                printf("本场比赛采用 11 分制\n");
                Sleep(500);
            }
            Hit(plr_sev,1); //发球
            //计算是否更改发球方
            if(og_score[0]>=oggoal-1 && og_score[1]>=oggoal-1) plr_sev=1-plr_sev;
            else{

```

```

        plr_alrsev++;
        if(plr_alrsev==2){
            plr_sev=1-plr_sev;
            plr_alrsev=0;
        }
    }
    MoodandRand();
}
//一局比赛结束
}
//一场比赛结束
if(demo){
    copts_cls();
    printf("比赛结束 小比分 %d : %d\n\n",og_score[0],og_score[1]);
}
if (wg_score[0]==wggoal)
    {if (demo)printf("最终胜者为:选手 A\n");
    return 0;}
else
    {if(demo)printf("最终胜者为:选手 B\n");
    return 1;}
}

void Game::Hit(int phit,int mod){
    //demo 模式输出
    if(demo){
        switch(mod){
            case 0:{printf("\n 选手 %c 非进攻回打, 进入相持 ",strindex[phit]);Sleep(300);break;}
            case 1:{printf("\n 选手 %c 发球 ",strindex[phit]);Sleep(300);break;}
            case 2:{printf("\n 选手 %c 接球抢攻 ",strindex[phit]);Sleep(300);break;}
            case 3:{printf("\n 选手 %c 发球抢攻 ",strindex[phit]);Sleep(300);break;}
            case 4:{printf("\n 选手 %c 发球反攻 ",strindex[phit]);Sleep(300);break;}
            case 5:{printf("\n 选手 %c 接发球 ",strindex[phit]);Sleep(300);break;}
        }
    }

    if (mod==1){ //第一板发球
        if(Random(plr[phit].first_sev_scr))
            {og_score[phit]++;
            if(demo){printf("直接得分---当前小比分 %d : %d \n",og_score[0],og_score[1]);Sleep(110
0);}

            return;}}

    }
    //是否命中,先讨论未命中
    if(mod>=2&&mod<=4){
        if(!Random(plr[phit].atk_hit*plr[phit].addi_hit))
            {og_score[1-phit]++;
            if(demo){printf("出界, 对方得分---当前小比分 %d : %d \n",og_score[0],og_score[1]);Sleep(1
100);}

            return;}}

    }
    else if(!Random(plr[phit].nor_hit*plr[phit].addi_hit))
        {og_score[1-phit]++;
        if(demo){printf("出界, 对方得分---当前小比分 %d : %d \n",og_score[0],og_score[1]);Sleep(1
100);}

        return;}}

    //命中后转到对手的 Catch

```

```

    Catch(1-phit,mod);
    return;
}

void Game::Catch(int pcth,int mod){
    //是否接住,先讨论未接住
    if(mod>=2&&mod<=4){
        if(!Random(plr[pcth].atk_catch*plr[pcth].addi_cth))
        {og_score[1-pcth]++;
        if(demo){printf("对手未接住,己方得分---当前小比分 %d : %d \n",og_score[0],og_score[1]);Sleep(1100);}
        return;}
    }
    else if(!Random(plr[pcth].nor_catch*plr[pcth].addi_cth))
    {og_score[1-pcth]++;
    if(demo){printf("对手未接住,己方得分---当前小比分 %d : %d \n",og_score[0],og_score[1]);Sleep(1100);}
    return;}
    //接住后判断对策
    if(demo) printf("\n");
    switch (mod){
        case 0:{Stalemate(); return;}
        case 1:{ //接到对方发球
            if(Random(plr[pcth].cthatk_usg)) Hit(pcth,2);//接抢
            else Hit(pcth,5); //接发球
            break;
        }
        case 2:{ //接到对方接抢
            if(Random(plr[pcth].sevctratk_usg)) Hit(pcth,4);//发球反攻
            else Hit(pcth,0); //进入相持
            break;
        }
        case 3: Hit(pcth,0); break;
        case 4: Hit(pcth,0); break;
        case 5:{ //接到对方接发球
            if(Random(plr[pcth].sevatk_usg)) Hit(pcth,3);//发球抢攻
            else Hit(pcth,0); //进入相持
            break;
        }
    }
}

void Game::Stalemate(){
    int a=plr[0].stalemate_lv,b=plr[1].stalemate_lv;
    if(Random(a/(float)(a+b)))
    {og_score[0]++;
    if(demo){printf("\n\n相持结束,选手 A 得分---当前小比分 %d : %d \n",og_score[0],og_score[1]);Sleep(1100);}
    }
    else
    {og_score[1]++;
    if(demo){printf("\n\n相持结束,选手 B 得分---当前小比分 %d : %d \n",og_score[0],og_score[1]);Sleep(1100);}
    }
    return;
}

```

```

bool Game::Checkover(){
    int t;
    if(abs(og_score[0]-og_score[1])<=1||
        (og_score[0]>og_score[1]?og_score[0]:og_score[1])<(one_game==1?11:21))
        return false;
    t=(og_score[0]>og_score[1]?0:1);
    wg_score[t]++;
    if(demo){
        printf("\n 本局结束 小比分 %d : %d\n\n 本局胜者为:选手%c\n",
            ,og_score[0],og_score[1],strindex[t]);
        Sleep(1500);
    }
    return true;
}

void Game::MoodandRand(){
    plr[0].addi_hit=RandomVal(0.95,1.1);
    plr[1].addi_hit=RandomVal(0.95,1.1);
    plr[0].addi_cth=RandomVal(0.95,1.1);
    plr[1].addi_cth=RandomVal(0.95,1.1);
    for(int i=0;i<=1;i++){
        if(plr[i].addi_hit*plr[i].atk_hit>=1||plr[i].addi_hit*plr[i].nor_hit>=1)
            plr[i].addi_hit=1;
        if(plr[i].addi_cth*plr[i].atk_catch>=1||plr[i].addi_cth*plr[i].nor_catch>=1)
            plr[i].addi_cth=1;
    }
    if (!(abs(og_score[0]-og_score[1])<=1||
        (og_score[0]>og_score[1]?og_score[0]:og_score[1])<(one_game==1?11:21)))
        return;
    int plead=(og_score[0]>og_score[1]?0:1);
    if (og_score[plead]-og_score[1-plead]>=3){
        if(Random(0.125)){
            if(demo){printf("\n 分差较大, 领先方选手 %c 有所放松 (下一球接球率、命中率略微下降)\n",st
                rindex[plead]); Sleep(1500);}
            plr[plead].addi_hit-=RandomVal(0,0.25);
            plr[plead].addi_cth-=RandomVal(0,0.25);
        }
        if(Random(0.333)){
            if(demo) {printf("\n 落后方选手 %c 坚持不懈,奋起直追 (下一球命中率略微提升)\n",strindex
                [1-plead]); Sleep(1500);}
            plr[1-plead].addi_hit+=RandomVal(0,0.2);
            PlayerLevel &backw=plr[1-plead];
            if(backw.addi_hit*backw.atk_hit>=1||backw.addi_hit*backw.nor_hit>=1)
                backw.addi_hit=0.99/(backw.nor_hit>backw.atk_hit?backw.nor_hit:backw.atk_hit);
        }
    }
}

void Game::CalcsynLevel(PlayerLevel &p){
    int tot=0;
    tot+=p.first_sev_scr*100;
    tot+=p.nor_hit*150;
    tot+=(p.atk_catch*360+p.nor_catch*240);
    tot+=(p.sevatk_usg*p.atk_hit*150);
    tot+=(p.sevctratk_usg*p.atk_hit*150);
    tot+=(p.cthatk_usg*p.atk_hit*150);
    tot+=p.stalemate_lv*2;
    p.syn_lv=tot;
}

```

```
    return;  
}  
  
float RandomVal(float l,float r){  
    int ran;  
    int L=(int)(l*1000),R=(int)(r*1000);  
    ran=rand()%(R-L+1)+L;  
    return (ran/1000.00);  
}
```

