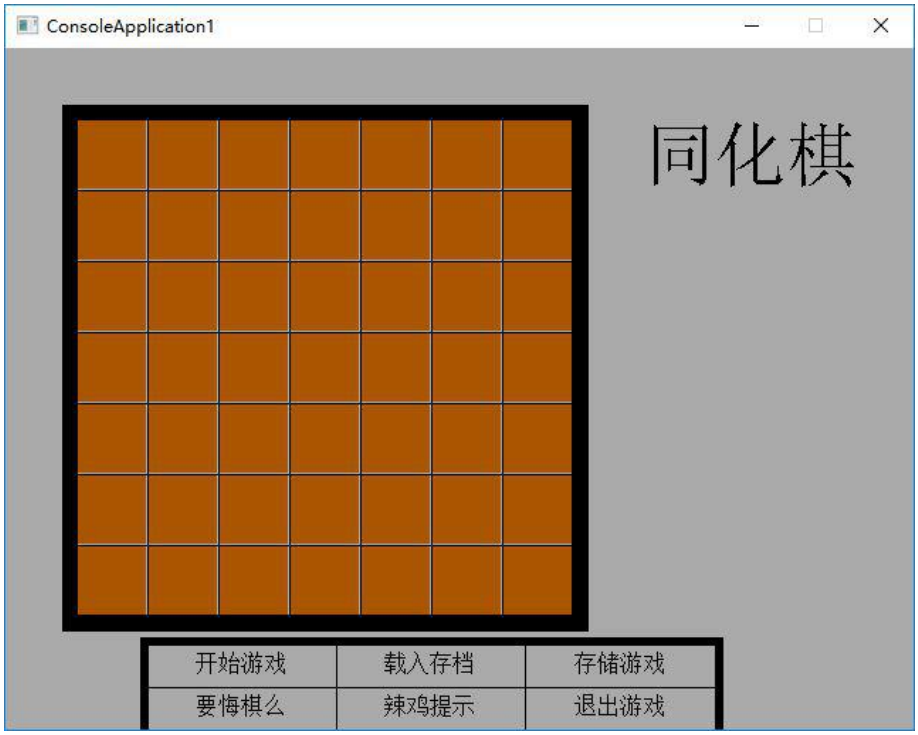
同化棋

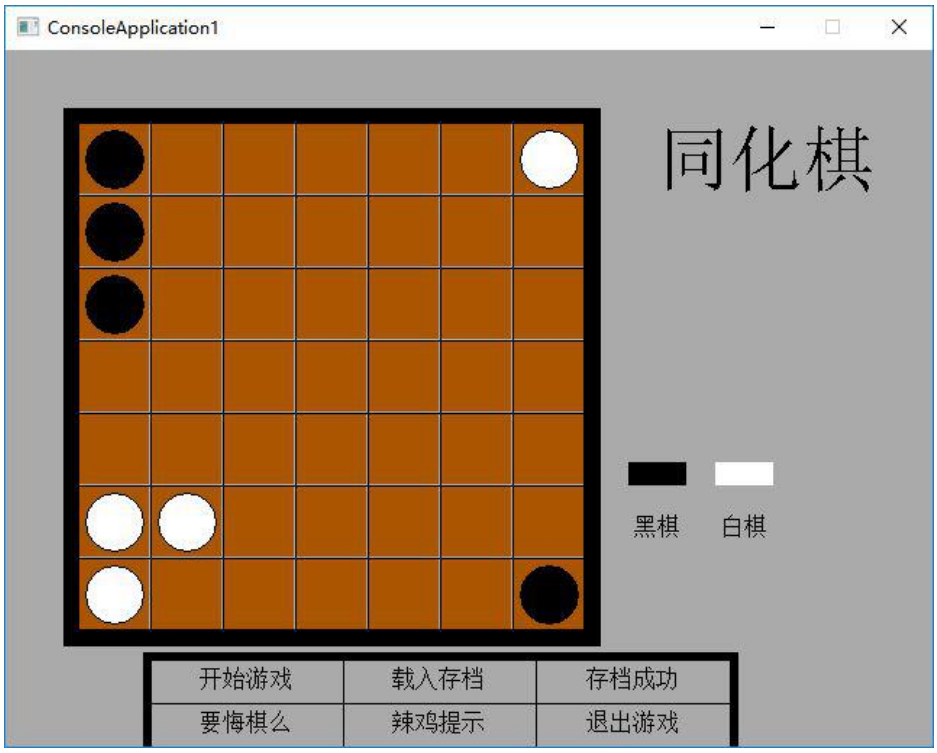
一. 游戏界面及基本操作



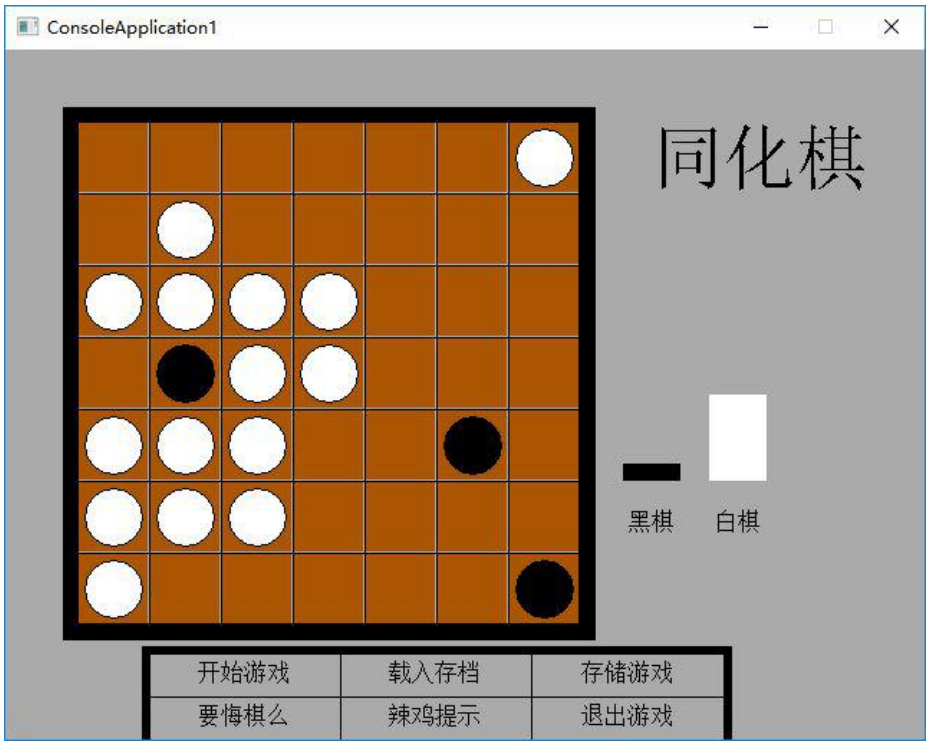
右上角显示游戏标题，下面六个按键可供点击控制游戏。

点击“开始游戏”后，出现黑白双方的棋子，右侧显示双方子数对比。

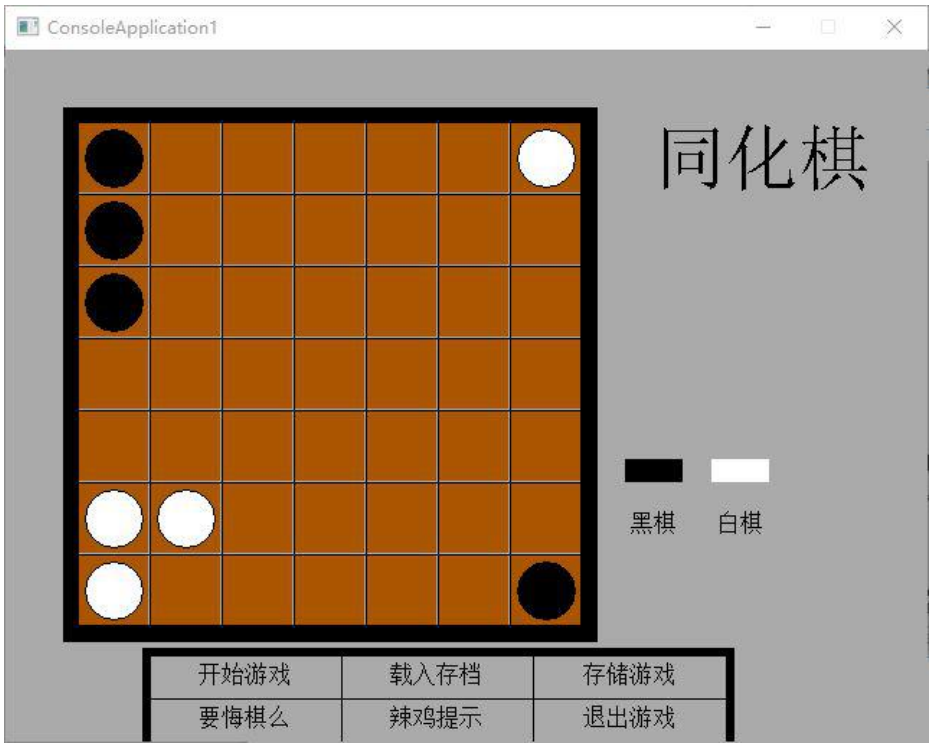
点击左上角的黑子，出现可以落子的区域。



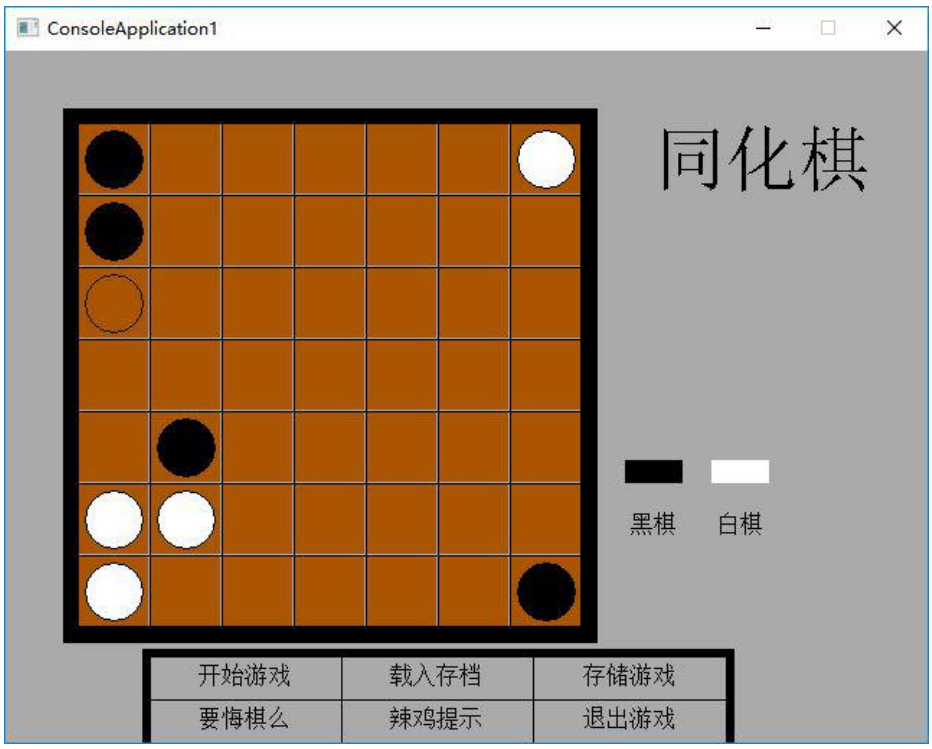
存档，显示存档成功。



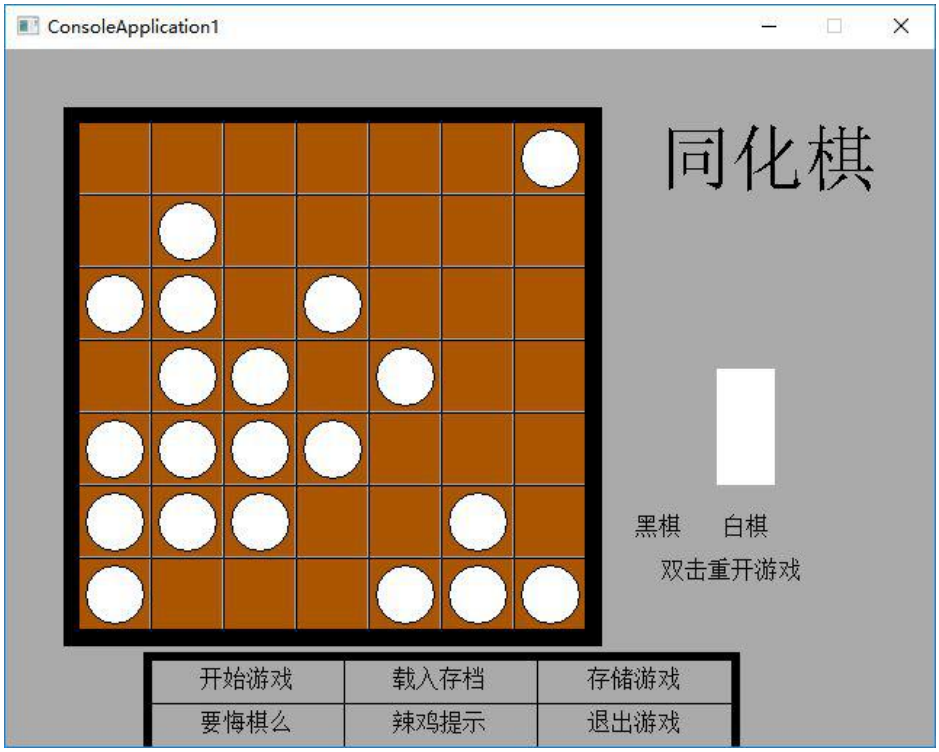
走错棋步后可以读档挽救（↑读档前↑　　↓读档后↓）。



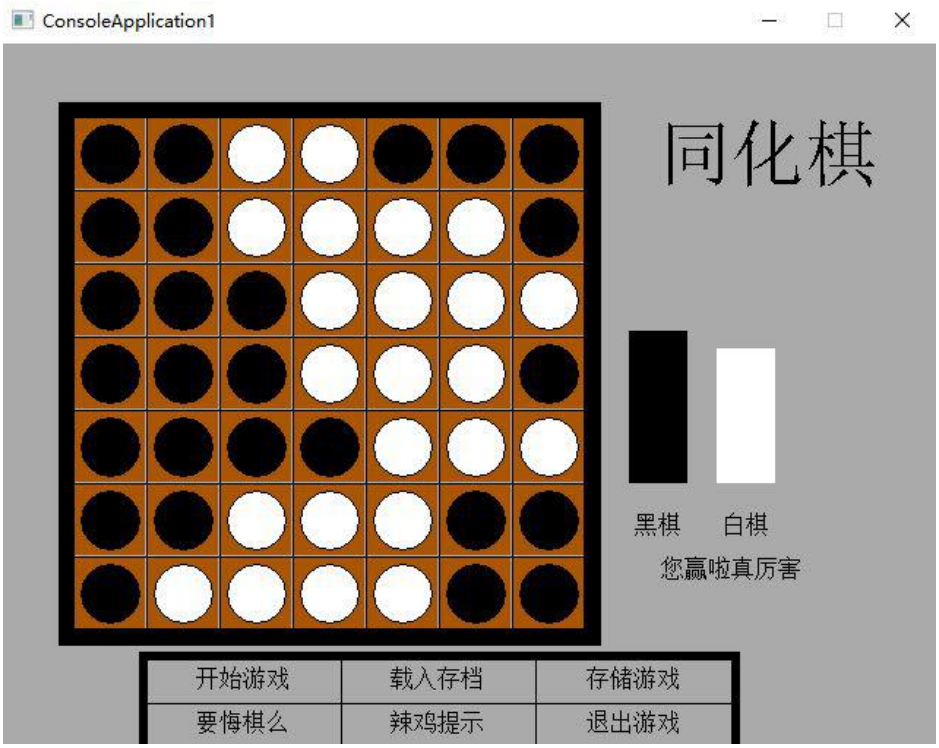
存档放置在文件中，所以退出游戏后重开仍然可以读档。



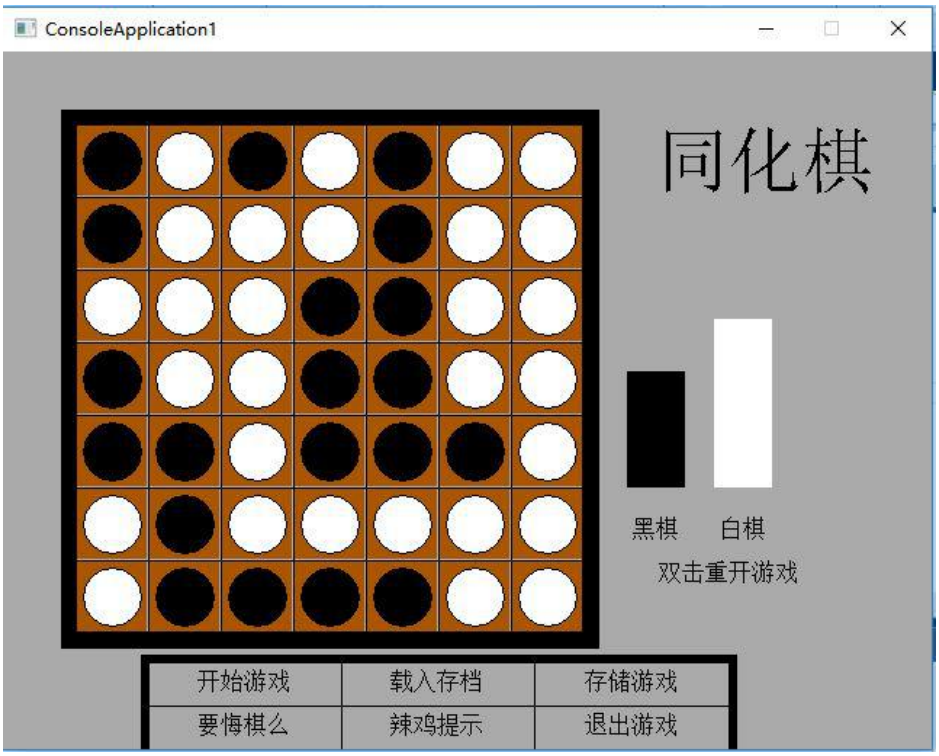
点击“辣鸡提示”后，闪烁提示一种可行的走法



输棋之后右下提示重开游戏，也可以直接退出。



获胜之后的截图



失败之后的截图

此外，点击“要悔棋么”可以退回上一步的棋盘。
白棋落子时会有闪烁显示电脑移动的棋子。
任何时间点击“退出游戏”都可以直接退出。

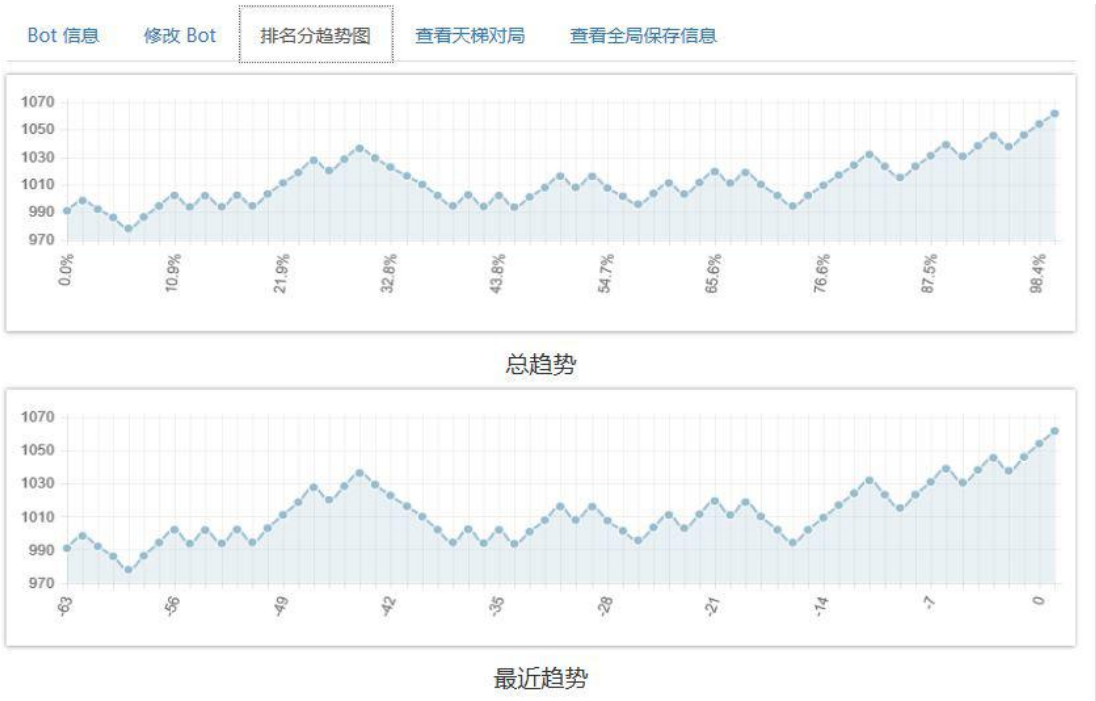
二．Botzone 平台表现

129	贪心的贪吃棋	Adamink	1065.94	二步贪心	9	source.cpp ID
130	q973544b7	Bennot	1062.59	max-min算法	1	source.cpp ID
131	辣鸡bot	DanDoge	1061.63	就是个辣鸡程序	8	source.cpp 进行手动天梯对局 ID
132	ElectroYeast_V4	zh12176	1061.28	TIMES	51	source.cpp ID
133	gakki赛高	GAKKI大好	1059.65	一个continue引发的血案	0	source.cpp ID

Bot 排行：131 / 274

105	Anonymous	怎么又是人工辣鸡	8
106	彩虹在我吗	bot小纯洁	8
107	Hanzo (1600012993)	扶我起来我还能送	8
108	DanDoge (1600017857)	辣鸡bot	8
109	还有比我更菜的吗 (1600012828不管了死就死)	我就是比你更菜的1	8
110	paridos (1600012790)	JustAGame	8

排位赛：108 / 176



分数在 1010 - 1060 间波动

三．代码实现

• 图形界面

由于使用了 EasyX（添加头文件“graphics.h”）可以使用矩形来描述界面。

棋盘的填充通过 map[8][8][8]完成，前两维用来表示 7 * 7 的棋盘，最后一维表示每个方块四个顶点的横纵坐标，以便填充颜色。

```
int map[8][8][8] = {};
RECT startgame = { 100,425,233,455 };
RECT loadgame = { 233, 425, 366, 455 };
RECT savegame = { 366, 425, 500, 455 };
RECT huiqi = { 100, 455,233,475 };
RECT hint = { 233,455,366,475 };
RECT quitgame = { 366,455,500,475 };
RECT help = { 450,350,550,400 };
RECT title = { 425, 50, 625, 100 };
//下面是对它们的初始化
drawtext(_T("开始游戏"), &startgame, DT_CENTER);
drawtext(_T("载入存档"), &loadgame, DT_CENTER);
drawtext(_T("存储游戏"), &savegame, DT_CENTER);
drawtext(_T("辣鸡提示"), &hint, DT_CENTER);
drawtext(_T("要悔棋么"), &huiqi, DT_CENTER);
drawtext(_T("退出游戏"), &quitgame, DT_CENTER);
```

显示双方子数对比通过调用函数得到子数，再填充相应大小的矩形完成。

```
void drawsums() {
    setfillcolor(LIGHTGRAY);
    POINT ini[4] = { {430,300}, {430, 100}, {530, 100}, {530, 300} };
    solidpolygon(ini, 4);

    setfillcolor(BLACK);
    POINT blacksum[4] = { { 430, 300 },{ 430, 300 - getblacknum() * 4 },
                          { 470, 300 - getblacknum() * 4 },{ 470, 300 } };
    solidpolygon(blacksum, 4);
    setfillcolor(WHITE);
    POINT whitesum[4] = { { 490, 300 },{ 490, 300 - getwhitenum() * 4 },
                          { 530, 300 - getwhitenum() * 4 },{ 530, 300 } };
    solidpolygon(whitesum, 4);
}
```

- **main 函数主要代码**

```
if (m.mkLButton == true) {
    //悔棋
    if (m.x < 233 && m.x > 100 && m.y > 455 && m.y < 475) {
        gethuiqi();
    }
    //存读档
    if (m.x < 366 && m.x > 233 && m.y > 425 && m.y < 455) {
        saveloadgame(1);
    }

    if (m.x < 500 && m.x > 366 && m.y > 425 && m.y < 455) {
        saveloadgame(-1);
    }
    //玩家落子
    if (m.x < 400 && m.x > 50 && m.y > 50 && m.y < 400) {
        if (getpixel(m.x, m.y) == BLACK && m.y % 50 && m.x % 50) {
            checklegal(m);
        }
        else if (getpixel(m.x, m.y) == GREEN) {
            savemap();//先存储当前棋盘，电脑再落子
            move(m.x, m.y);//玩家落子
            drawsums();//更新玩家吃掉的子数
            decide();//电脑决策
            drawsums();//更新电脑吃掉的子数
        }
    }
```

每次玩家点下鼠标后，考虑所有可能点击的位置，分别作出反应。
在落子时需要多次更新黑白子的个数，来保证时效性。

```
if (checkwinlose() == 1) {
    drawtext(_T("您赢啦真厉害"), &help, DT_CENTER);
    Sleep(1000);
    drawtext(_T("双击重开游戏"), &help, DT_CENTER);
    break;
}
else if (checkwinlose() == -1) {
    drawtext(_T("很抱歉您输了"), &help, DT_CENTER);
    Sleep(1000);
    drawtext(_T("双击重开游戏"), &help, DT_CENTER);
    break;
```

在游戏结束时，显示游戏结果，并提醒重开游戏或者直接退出。

- **存读档**

```
void saveloadgame(int mode) {
    if (mode == 1) {
        FILE *fp = fopen("save.in", "r");
        if (fp) {
            for (int i = 1; i <= 7; i += 1) {
                for (int j = 1; j <= 7; j += 1) {
                    int num = 0;
                    fscanf(fp, "%d", &num);
                    setfillcolor(BROWN);
                    solidpolygon((POINT*)map[i][j], 4);
                    switch (num) {
                    case 1: setfillcolor(BLACK);
                        fillcircle(i * 50 + 25, j * 50 + 25, 20);
                        break;
                    case -1:setfillcolor(WHITE);
                        fillcircle(i * 50 + 25, j * 50 + 25, 20);
                        break;
                    }
                }
            }
            fclose(fp);
            drawsums();
        }
        else {
            drawtext(_T("没有存档"), &loadgame, DT_CENTER);
            Sleep(500);
            drawtext(_T("SO SAD"), &loadgame, DT_CENTER);
            Sleep(500);
            drawtext(_T("载入存档"), &loadgame, DT_CENTER);
        }
    }
```

从与 exe 文件同根目录的 save.in 文件中读入存档，复原棋盘，并更新黑白子个数
如果读档失败，在“载入存档”处显示“没有存档”以提示用户

```
else {
        FILE *fp = fopen("save.in", "w+");
        if (fp) {
            for (int i = 1; i <= 7; i += 1) {
                for (int j = 1; j <= 7; j += 1) {
                    if (getpixel(i * 50 + 25, j * 50 + 25) == WHITE) {
                        fprintf(fp, "%d ", -1);
                    }
                    else if (getpixel(i * 50 + 25, j * 50 + 25) == BLACK) {
                        fprintf(fp, "%d ", 1);
                    }
                    else {
                        fprintf(fp, "%d ", 0);
                    }
                }
                fprintf(fp, "\n");
            }
            fclose(fp);
            drawtext(_T("存档成功"), &savegame, DT_CENTER);
            Sleep(1000);
            drawtext(_T("存储游戏"), &savegame, DT_CENTER);
        }
        else {
            drawtext(_T("存储失败"), &savegame, DT_CENTER);
            Sleep(500);
            drawtext(_T("SO SAD"), &savegame, DT_CENTER);
            Sleep(500);
            drawtext(_T("存储游戏"), &savegame, DT_CENTER);
        }
    }
```

存档存入同一个文件中，按照棋盘方格的中心点颜色，把棋盘信息转化成数字，写入文件。存档成功后，会提示用户“存档成功”，否则，显示“存储失败”。

- **判断输赢**

一方面，当黑白任意一方没有子时，游戏结束，另一方面，如果玩家没有地方可走，游戏也会结束。

调用获得子数的函数，如果某一方没有子，另一方胜利。

如果玩家的所有子的周围（距离为 2 以内）都不能落子，游戏结束，按照双方子数多少，判定输赢。

```
int checkwinlose() {
    if (getblacknum() == 0) {
        return -1;
    }
    else if (getwhitenum() == 0) {
        return 1;
    }
    else {
        return 0;
    }
```

```
int userlegal() {
    for (int i = 75; i <= 375; i += 50)
        for (int j = 75; j <= 375; j += 50)
            if (getpixel(i, j) == BLACK) {
                for (int p = i - 100; p <= i + 100; p += 50)
                    for (int q = j - 100; q <= j + 100; q += 50)
                        if (getpixel(p, q) == BROWN || getpixel(p, q) == GREEN) {
                            return 1;
                        }
            }
    return -1;
}
```

- **悔棋实现**

玩家每次落子前都会先储存棋盘现状，以供下一步悔棋使用（见 main 函数代码）。复原悔棋后的棋盘过程与读档类似。

```
void savemap() {
    for (int i = 1; i <= 7; i += 1)
        for (int j = 1; j <= 7; j += 1)
            if (getpixel(i * 50 + 25, j * 50 + 25) == BLACK) {
                recentmap[i][j] = 1;
            }
            else if (getpixel(i * 50 + 25, j * 50 + 25) == WHITE) {
                recentmap[i][j] = -1;
            }
            else {
                recentmap[i][j] = 0;
            }
}
```

- **核心算法**

```
if (times == maxtimes) {
        return 0;
    }
    else {
        int nowmap[8][8] = {};
        for (int i = 1; i < 8; i += 1) {
            for (int j = 1; j < 8; j += 1) {
                if (getpixel(i * 50 + 25, j * 50 + 25) == BLACK) {
                    nowmap[i][j] = 1;
                }
                else if (getpixel(i * 50 + 25, j * 50 + 25) == WHITE) {
                    nowmap[i][j] = -1;
                }
            }
        }
        if (abs(x1 - x2) > 1 || abs(y1 - y2) > 1) {
            nowmap[x1][y1] = 0;
        }
        nowmap[x2][y2] = color;
        for (int i = max(1, x2 - 1); i <= min(7, x2 + 1); i += 1) {
            for (int j = max(1, y2 - 1); j <= min(7, y2 + 1); j += 1) {
                if (nowmap[i][j] + nowmap[x2][y2] == 0) {
                    nowmap[i][j] = nowmap[x2][y2];
                }
            }
        }
```

首先新建一个数组来模拟棋盘，并完成一步落子，以及落子之后的同化过程。

```
for (int i = 1; i < 8; i += 1)
    for (int j = 1; j < 8; j += 1)
        if (nowmap[i][j] == color) {//找到了一个可以走的棋子
            for (int k = max(1, i - 2); k <= min(7, i + 2); k += 1)
                for (int l = max(1, j - 2); l <= min(7, j + 2); l += 1)
                    if (nowmap[k][l] == 0) {//找到了一个可以走的地方
                        int tempcnt = 0;
                        for (int p = max(1, k - 1); p <= min(7, k + 1); p += 1)
                            for (int q = max(1, l - 1); q <= min(7, l + 1); q += 1)
                                if (nowmap[p][q] == -color) {//看能同化多少个棋子
                                    tempcnt += (5 - times);
                                }
                        if (max(abs(i - k), abs(j - l)) > 1)
                            tempcnt -= 5;//考虑到一次走两格相当于少了一个棋子
                        //递归，计算这样走会丢掉多少棋子
                        tempcnt -= getgain(i, j, k, l, -color, times + 1);
                        maxcnt = max(maxcnt, tempcnt);
                    }
        }
```

通过调用 getgain()函数计算走每一步时可能的最大收益，通过调用自身的过程考虑到后续几步对方落子带来的影响（也用同样的逻辑估计对方的落子）。

```
if (start[0] > 0 && start[1] > 0) {
        for (int i = 0; i < 3; i += 1)
            setfillcolor(BROWN);
            fillcircle(start[0], start[1], 20);
            Sleep(200);
            setfillcolor(WHITE);
            fillcircle(start[0], start[1], 20);
            Sleep(200);
        if (abs(start[0] - end[0]) <= 50 && abs(start[1] - end[1]) <= 50) {
            setfillcolor(WHITE);
            fillcircle(end[0], end[1], 20);
        }
        else {
            setfillcolor(WHITE);
            fillcircle(end[0], end[1], 20);
            setfillcolor(BROWN);
            solidpolygon((POINT*)map[start[0] / 50][start[1] / 50], 4);
        }
        affected(end[0], end[1]);
    }
```

最后得到落子位置之后，闪烁电脑所执的子几次提醒玩家，之后落子。

- **鼠标捕捉**

```
MOUSEMSG m;
    while (true) {
        m = GetMouseMsg();
        //执行操作
        if (...) {
            ...;
            break;
        }
    }
```

通过不断读取鼠标信息捕捉用户的操作，作出相应的判断。

- **函数定义**

```
int getblacknum();
int getwhitenum();
void newinitialize();
void initializegame();
void checklegal(MOUSEMSG m);
void move(int x, int y);
void affect(int x, int y, int mode);
void drawsums();
int checkwinlose();
void decide();
int getgain(int x1, int y1, int x2, int y2, int color, int times);
int userlegal();
void gethint();
void gethuiqi();
void savemap();
void saveloadgame(int mode);
```

其中 newinitialize() 用来初始化最开始的界面，即棋盘、下面的按钮和标题，而initializegame()用来初始化棋盘，显示最开始的四个子，调用获取子数的函数。

Move()用来处理用户的落子，包括转移黑子和后续的同化过程（通过调用 affect()实现）。

四．过程中的困难和解决方法

• 图形界面

由于是第一次接触和界面有关的程序，一开始的大部分时间花在阅读 EasyX 的附带文档上。同时在自己测试代码的情况下，逐渐了解到了一些基本的使用方法。

之后遇到的就是界面的美观性的问题，这里参考了网上的一些棋盘的图片，再考虑到尽量使编程简单，把棋盘方格大小设置成 50 个像素，棋盘周围画了一圈边框以求有一定的层次感。用户的操作按钮随着功能逐渐实现数目逐渐增加，最终设置到了下面。右侧实时显示子数对比也通过阅读附带文档逐渐实现。

• 悔棋功能

悔棋功能要求实时保存之前一步的棋盘信息，又要求在落子之前能读取出来，而不会被新的棋盘信息覆盖掉，这其实对储存的时间和读取的时间设计有一定的要求。通过不断调试不同的位置配置，最终得出把读档设置在最优先的位置，把存档设置在贴近落子的位置是不会出问题的。虽然这种配置现在看来是自然的，但是这个过程中也练习了一些试错的经验。

• 落子算法

落子的逻辑是暴力遍历所有可能的情况，之后取其中的最大值。一开始把这种思路设计成了很多的 for 循环，但是仍然只能解决自己的一步落子，并不能模拟对方的落子，更不用提更深层的搜索了。后来注意到对方的逻辑也可以用自己的算法模拟，而更深层的搜索可以用递归完成，于是就做了这些改进，虽然必须承认的是，改进后思路仍然是十分简单的。