孔明棋 Peg Solitaire

第 3 次大作业报告

国豪工科 7 班

计算机科学与技术

2352197

冉子易

2024 年 6 月 20 日

目录

1 概览

1.1 功能描述

游戏界面如下，点 PLAY 键开始游戏。点左上角按钮退回菜单。点右上角按钮悔棋。

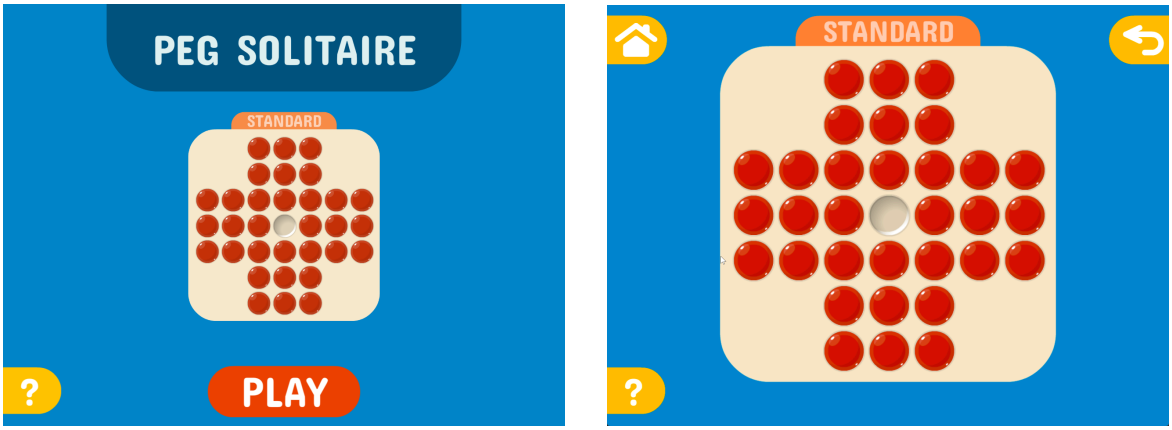

图 1.1: 菜单和标准棋盘

点击左下角按钮可显示帮助。点 OK 键退出。

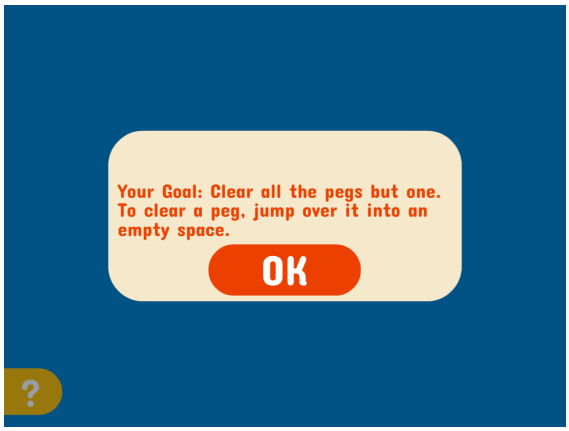

图 1.2: 帮助

残局模式通过隐藏按键（1, 2, 3, 4）进入，比如，可按 1 进入心形局.

无子可落时代表游戏结束。在此基础上若仅剩一子，则游戏胜利。

1.2 设计思路

主要考虑使用 EasyX 图形库设计游戏。easyX 的基本函数（图形库结合 Windows API）包括加载图片 `loadimage`、在某坐标插入图片 `putimage` 和一些其它的缓冲函数（防止闪烁）。

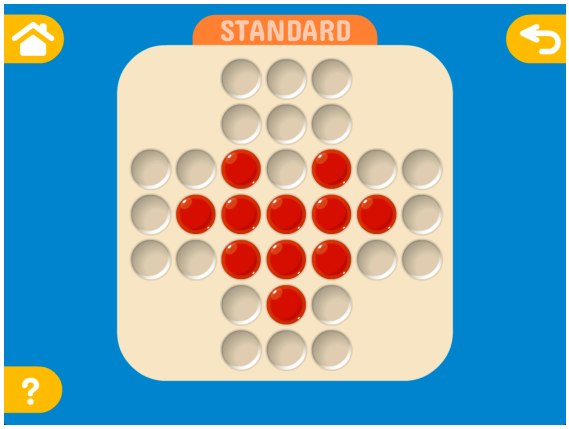

图 1.3: 心形残局

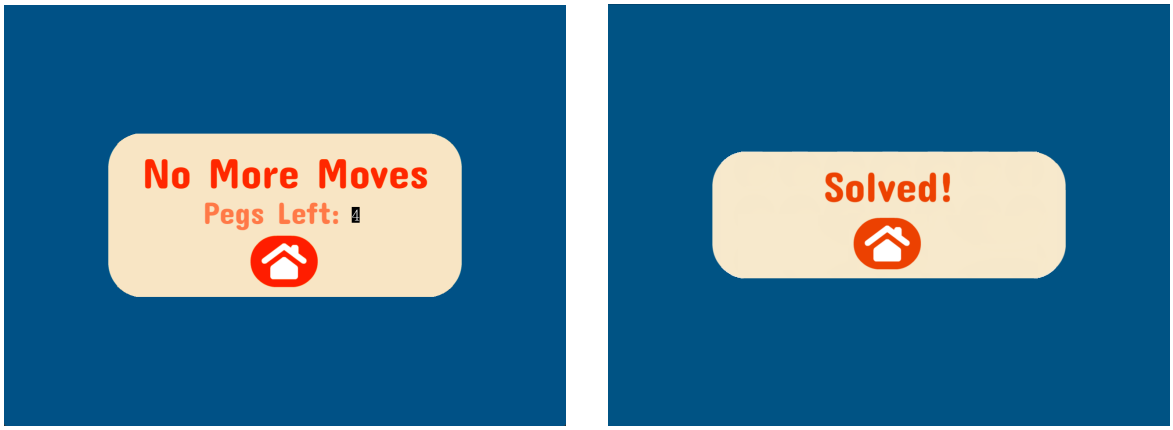

图 1.4: 普通结束和胜利画面

用户交互通过鼠标，鼠标信息记录为 `ExMessage mouse_msg`，其获取函数为 `getmessage`。基本逻辑是通过鼠标指针的位置以及按键信息，判断用户的操作。

游戏的基本逻辑是用数组存储棋盘和空位信息。有时棋盘并非完整的方形，故聪明的做法是将那些非棋盘区域设置为 `-1`。用类 `class standard` 单独存储各种变量：棋盘、悔棋记忆、剩余数量、玩家选择、玩家落点，以及函数：显示空棋盘、显示当前棋子布局、显示选择状态（和周围可走空位）、游戏结束判断。除此之外，可加上棋子移动的动画效果。这些功能在游戏主循环函数中的实现是一目了然的。

动画设计上的细节是，不仅可以使棋子滑到终点，还可使其大小先增大后减小以模拟一种飞跃感；同时，所消去的棋子也可用一个大小的连续变化来过渡，而非直接消失。

残局模式使用的棋盘仍然是标准棋盘，只是布局上棋子并非 32 颗。因此，只需用一个结构体 `struct other_standard` 存储这些残局情况即可。在实际游戏中，根据玩家所按下的隐藏按键，调用相应棋子布局并进行替换。

对于三角排列的棋盘，数组设置仍然基于方形原则，只是在显示时错位，且一个棋子要考虑的是周围六个方向上的情况（本行是左右两颗，上行则是与本棋相邻两颗，下方同

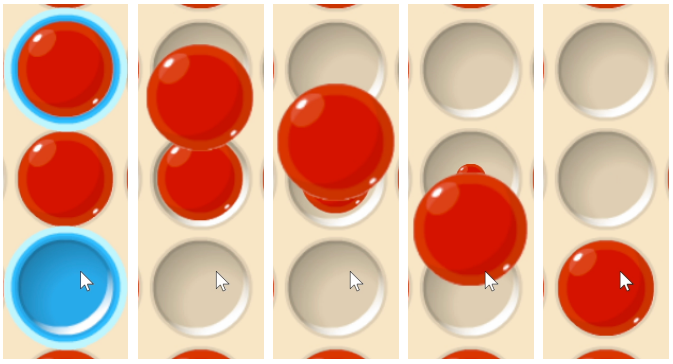

图 1.5: 飞跃过程的 5 次截取。实际帧数为 20，可设置

理)。其余的游戏逻辑有大量地方需要从方形情形修改，因此要用另一个类存储。此外，AI 破解棋局的基础是遍历，但在优化上需要借助路径记忆防止栈溢出。这些功能由于时间因素未完善，但原则上实现逻辑仍然是通俗的。

2 问题及解决方法

2.1 素材

需要花许多时间使用 PS 抠图。比如有如下棋子素材。

图 2.1: 棋子状态

2.2 音效

库和函数都准备好，且音效素材也准备好为 wav 形式，可于项目文件中查看。但在音效播放上出现了未知问题，可能与 AU 剪辑导出音频的设置有关。

2.3 动画

动画实现飞跃时，大小可通过二次函数 `size = a * (n - step) * n` 形式来构造，其中 `step` 表示动画帧数，而函数的其它项根据实际需求确定，比如可加上棋子边长作为常数项。

2.4 透明显示

显示透明颜色需要图像的 alpha 通道，编写一个专门的函数。可见后文。

3 心得体会

3.1 算法设计

主要算法并没有困扰我太多。甚至，由于这次游戏逻辑比较简单，可以脱离 AI 帮助。只是需要花时间查找关于 EasyX 的资料。

3.2 对结构体和类的理解

结构体相较于数组的优势在于其可以存储不同类型的数据，且调用语法更直观。而类在此基础上还支持公私之分，可用于将其对应的函数封装起来，可供外界随时调用。当然，底层变量则封锁起来。

3.3 游戏攻略

据说可以采取一种螺旋式的走棋方向。

3.4 所思所想

为了实现前端，工作量实在太肝，包括抠图、动画制作、坐标 debug，甚至还有则此没有实现的颜色深浅控制。然而结果却又比较赏心悦目，这何尝不是一种补偿性自虐。

4 部分源代码

Listing 1: 透明显示

```
inline void putimage_alpha(int x, int y, IMAGE* img) {
        int w = img->getwidth();
        int h = img->getheight();
        AlphaBlend(GetImageHDC(NULL), x, y, w, h,
                GetImageHDC(img), 0, 0, w, h, { AC_SRC_OVER,0,255,AC_SRC_ALPHA });
}
```

Listing 2: 飞跃动画

```
void move_animation(int x, int y, int to_x, int to_y, int step) {
    for (int n = 0; n < step; ++n) {
        show_board();
        show_pieces();

        float fading_size = 0.1 * n * (step - n) + 92 - (92 * n / step);
        IMAGE fading_piece;
        loadimage(&fading_piece, _T("image/piece.png"), fading_size, fading_size);
        putimage_alpha(x + (to_x - x) / 2 - (fading_size - 92) / 2, y + (to_y - y) / 2 - (fading_size -
             92) / 2, &fading_piece);

        float flying_size = 0.4 * n * (step - n) + 92;
        IMAGE flying_piece;
        loadimage(&flying_piece, _T("image/piece.png"), flying_size, flying_size);
        putimage_alpha(x + n * (to_x - x) / step - (flying_size - 92) / 2, y + n * (to_y - y) / step -
            (flying_size - 92) / 2, &flying_piece);
        FlushBatchDraw();
    }
}
```

动画函数存储在类 `class Game game` 里。

然后是一些游戏的主干架构，同样存储在类中。本程序的结构体用于存储棋盘数据。

Listing 3: 游戏主循环

```
void game() {
        save_memory();
        while (1) {
                show_board();
                show_pieces();

                getmessage(&mouse_msg);
```

```
8
9               //通过隐藏按键进入残局
10              if (mouse_msg.message == WM_KEYDOWN) {
11                      if (mouse_msg.vkcode == '1') {
12                              remaining = 11;
13                              for (int i = 0; i < 7; ++i) {
14                                      for (int j = 0; j < 7; ++j) {
15                                              board[i][j] = canju.board_1[i][j];
16                                      }
17                              }
18                      }
19                      if (mouse_msg.vkcode == '2') {
20                              remaining = 8;
21                              for (int i = 0; i < 7; ++i) {
22                                      for (int j = 0; j < 7; ++j) {
23                                              board[i][j] = canju.board_2[i][j];
24                                      }
25                              }
26                      }
27                      if (mouse_msg.vkcode == '3') {
28                              remaining = 8;
29                              for (int i = 0; i < 7; ++i) {
30                                      for (int j = 0; j < 7; ++j) {
31                                              board[i][j] = canju.board_3[i][j];
32                                      }
33                              }
34                      }
35                      if (mouse_msg.vkcode == '4') {
36                              remaining = 6;
37                              for (int i = 0; i < 7; ++i) {
38                                      for (int j = 0; j < 7; ++j) {
39                                              board[i][j] = canju.board_4[i][j];
40                                      }
41                              }
42                      }
43              }
44
45              if (mouse_msg.message == WM_LBUTTONDOWN) {
46                      //悔棋
47                      if (mouse_msg.x >= 1143 && mouse_msg.x <= 1280 && mouse_msg.y >= 25 &&
                            mouse_msg.y <= 156) {
48                              if (memory_count > 0) {
49                                      memory_count--;
50                                      for (int i = 0; i < 7; ++i) {
```

```
51                                      for (int j = 0; j < 7; ++j) {
52                                              board[i][j] = memory[i][j][memory_count];
53                                      }
54                              }
55                      }
56                      X = Y = to_X = to_Y = -1; //不显示
57              }
58              //退出
59              else if (mouse_msg.x >= 0 && mouse_msg.x <= 137 && mouse_msg.y >= 25 &&
                    mouse_msg.y <= 156) break;
60
61              //棋盘内
62              if (mouse_msg.x >= 288 && mouse_msg.x <= 288 + 7 * 103 && mouse_msg.y >= 125 &&
                     mouse_msg.y <= 125 + 7 * 103) {
63                      int state = board[(mouse_msg.y - 125) / 103][(mouse_msg.x - 288) /
                            103]; //此处状态
64                      //棋子
65                      if (state == 1) {
66                              X = (mouse_msg.x - 288) / 103;
67                              Y = (mouse_msg.y - 125) / 103;
68                      }
69                      //空位
70                      else if (state == 0) {
71                              to_X = (mouse_msg.x - 288) / 103;
72                              to_Y = (mouse_msg.y - 125) / 103;
73                      }
74              }
75              //帮助
76              else if (mouse_msg.x >= 0 && mouse_msg.x <= 131 && mouse_msg.y >= 824 &&
                    mouse_msg.y <= 939) {
77                      help();
78              }
79      }
80
81      show_selected(); //显示选中棋子&可走空位
82
83      if (can_move()) {
84              remaining--;
85              move_animation(288 + X * 103, 125 + Y * 103, 288 + to_X * 103, 125 + to_Y *
                    103, 15); //移动动画
86              board[to_Y][to_X] = 1;
87              memory_count++;
88              save_memory();
89      }
```

```
90
91              //是否通关
92              is_end = true;
93              for (int i = 0; i < 7; ++i) {
94                      for (int j = 0; j < 7; ++j) {
95                              if (board[i][j] == 1) {
96                                      if ((i + 2 < 7 && board[i + 1][j] == 1 && board[i + 2][j] == 0)
97                                              || (i - 2 >= 0 && board[i - 1][j] == 1 && board[i - 2][
                                                    j] == 0)
98                                              || (j + 2 < 7 && board[i][j + 1] == 1 && board[i][j +
                                                    2] == 0)
99                                              || (j - 2 >= 0 && board[i][j - 1] == 1 && board[i][j -
                                                    2] == 0))
100                                             is_end = false;
101                             }
102                     }
103             }
104
105             if (is_end) {
106                     if (remaining == 1) {
107                             IMAGE solved;
108                             loadimage(&solved, _T("image/solved.png"));
109                             putimage_alpha(0, 0, &solved);
110
111                     }
112                     else if (remaining != 1) {
113                             IMAGE end;
114                             loadimage(&end, _T("image/end.png"));
115                             putimage_alpha(0, 0, &end);
116
117                             settextstyle(16, 0, _T("Consolas"));
118                             LOGFONT f;
119                             gettextstyle(&f);
120                             f.lfHeight = 36;
121                             _tcscpy_s(f.lfFaceName, _T("华文行楷"));
122                             f.lfQuality = ANTIALIASED_QUALITY;
123                             settextstyle(&f);
124                             settextcolor(WHITE);
125                             TCHAR str[100] = {};
126                             _stprintf_s(str, _T("%d"), remaining);
127                             outtextxy(792, 461, str);
128                     }
129                     if (mouse_msg.message == WM_LBUTTONDOWN) {
130                             if (mouse_msg.x >= 562 && mouse_msg.x <= 714 && mouse_msg.y >= 525 &&
```

```
                                mouse_msg.y <= 641) {
131                                 break;
132                             }
133                         }
134                 }
135
136                 FlushBatchDraw();
137         }
138 }
```

Listing 4: 棋子显示

```
1  void show_pieces() {
2      IMAGE piece;
3      loadimage(&piece, _T("image/piece.png"));
4      for (int i = 0; i < 7; ++i) {
5          for (int j = 0; j < 7; ++j) {
6              if (board[i][j] == 1) {
7                  putimage_alpha(288 + j * 103, 125 + i * 103, &piece);
8              }
9          }
10     }
11 }
```

最后是主函数。

Listing 5: 主函数

```
1  int main() {
2          initgraph(1280, 960);
3          BeginBatchDraw();
4          //菜单循环
5          while (1) {
6                  menu();
7                  getmessage(&mouse_msg);
8
9                  //点击
10                 if (mouse_msg.message == WM_LBUTTONDOWN) {
11                         //开始游戏
12                         if (mouse_msg.x >= 466 && mouse_msg.x <= 811 && mouse_msg.y >= 824 && mouse_msg
                               .y <= 939) {
13                             standard Game;
14                             Game.game();
15                         }
16                         //帮助
```

```
17                          else if (mouse_msg.x >= 0 && mouse_msg.x <= 137 && mouse_msg.y >= 824 &&
                                mouse_msg.y <= 939) {
18                              help();
19                          }
20                  }
21
22                  FlushBatchDraw();
23          }
24          EndBatchDraw();
25          return 0;
26  }
```