C语言项目报告

项目名称：\_化学工厂\_

项目成员：\_\_\_\_宋源博 杨竞翔 张晔\_\_\_\_\_\_\_

填写日期：\_\_2020.9.9\_

C语言期末项目报告——化学工厂

宋源博 杨竞翔 张晔

吉林大学化学学院

**摘要** 化学工厂是一个由本小组成员原创的回合制卡牌类小游戏。其设计灵感最初来源于麻将和三国杀，但机制较为新颖。游戏与化学反应方程式结合在一起，在进行过程中通过抓牌与弃牌凑出方程式而得分，可以起到带领玩家巩固基础化学方程式的作用，体会化学之美。同时游戏具备对战数据记录回放以及分数排名的功能，使玩家具有更好的游戏体验。

**关键词** 卡牌类游戏；化学方程式；回放功能；排名功能

1.项目背景

化学学科学习的难点不仅仅在于繁复的规则，物质作用而产生的大量化学反应方程式也可能成为学习者的记忆负担。化学工厂小游戏整合了高中化学大部分的反应方程式，以新颖的游戏形式呈现出来，旨在帮助学生巩固关于化学方程式的记忆。

2.组内分工

本小组组长宋源博同学在项目中主要负责游戏大体框架的搭建、战斗总体流程的编写以及规则的实现、对代码进行最后的错误修改，以及进行卡牌游戏背景界面处理；贡献占总代码量的40%。

组员杨竞翔同学在项目中主要负责游戏回放系统部分的代码编写、图片制作，以及游戏规则的提出，贡献占总代码量的35%。

组员张晔同学在项目中负责游戏规则的提出、成绩记录与排名功能，最后共同进行错误修改，贡献占总代码量的25%。

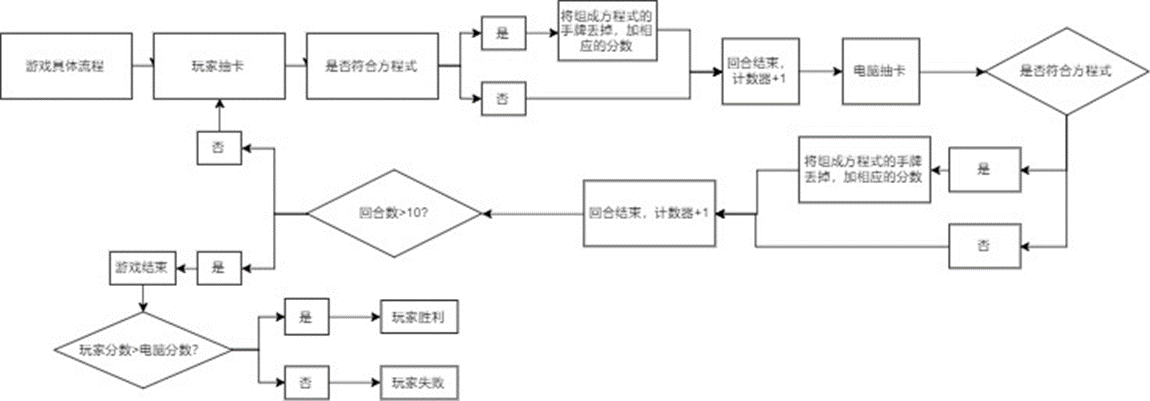
3.项目框架分析

化学工厂的游戏规则简单易懂。游戏中有三种牌型，第一种为系数（数字）牌，代表方程式中元素前的系数；第二种为反应条件牌；第三种为元素牌。系数牌包括2，3，4，5，6，8共40张，为白色牌面。其中系数1由于在方程式中存在过于普遍，游戏中默认系数“1”作为不显示出来的隐藏系数；仅设置6、8的原因也是在方程式中有较高的出现频率。条件牌包括加热、高温、催化剂、点燃、高压、通电、放电共50张，为白色牌面。元素牌包括氢、碳、氮、氧、氟、钠、镁、铝、硅、磷、硫、氯、钾、钙、锰、铁、铜、锌、溴、碘、铅，共90张，为各个元素特征颜色的牌面。

回合开始时己方分别抓取一张系数牌，条件牌，元素牌，可以选择保留或弃牌，敌方（电脑）也是如此。然而当手牌的数量即将超过9张时，由于窗口显示的范围限制，游戏将强制玩家弃牌。当手中的三种牌可以凑出化学方程式时，系统将自动判定方程式成立并撤掉组成方程式的手牌，将方程式相应的分数计入玩家分数。根据元素牌的稀有程度以及方程式的复杂程度，方程式的分数从1分至4分不等。例如，方程式：为1分的方程式，而则为4分。每局游戏进行10个回合，游戏结束后根据敌我双方分数的高低判断胜负，若分数相同则根据手牌数的多少判定，手牌较少的一方获胜。

主题函数的设计思路也是如此。首先通过参数Overvalue判断是否开始游戏，开始游戏后进行界面初始化，己方通过三个抓牌函数先抓三种牌，再进入到回合函数Round\_P1中（对方回合为Round\_P2）。回合函数内包括弃牌函数和分数判定函数（P1\_jgmt），判断现阶段手牌中是否有完整的方程式并打出后，己方回合结束，回合数加一。对方回合开始，流程与己方回合基本相同。在游戏到达10回合判定为结束后，程序跳出循环，执行回放函数，回放函数后为分数排名函数。

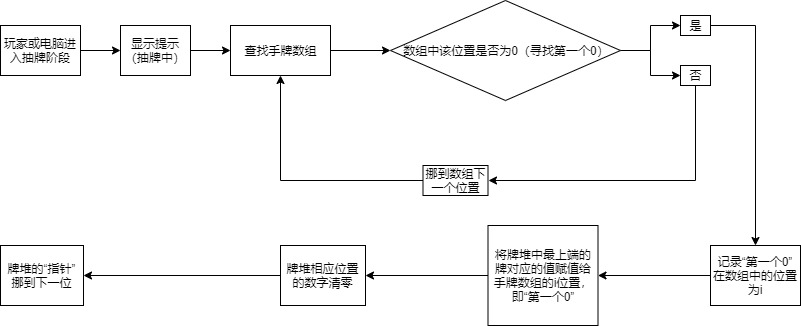




4.功能设计

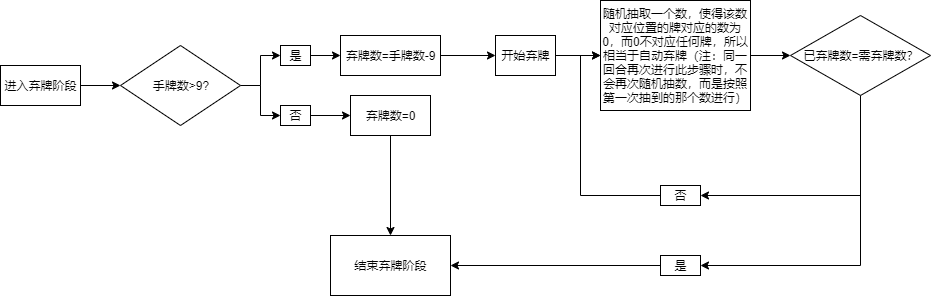
(1).抓牌函数

抓牌函数drawele、drawcond、drawcons分别为抓取元素牌、条件牌、系数牌的函数。首先函数会在界面上显示：“我方摸元素/条件/系数牌中！”，再通过一个for循环找到手牌的牌尾（等于0的数组元素），将牌堆中第一个代码的值赋给牌尾元素。最后将手牌的元素（条件/系数）牌数加一。



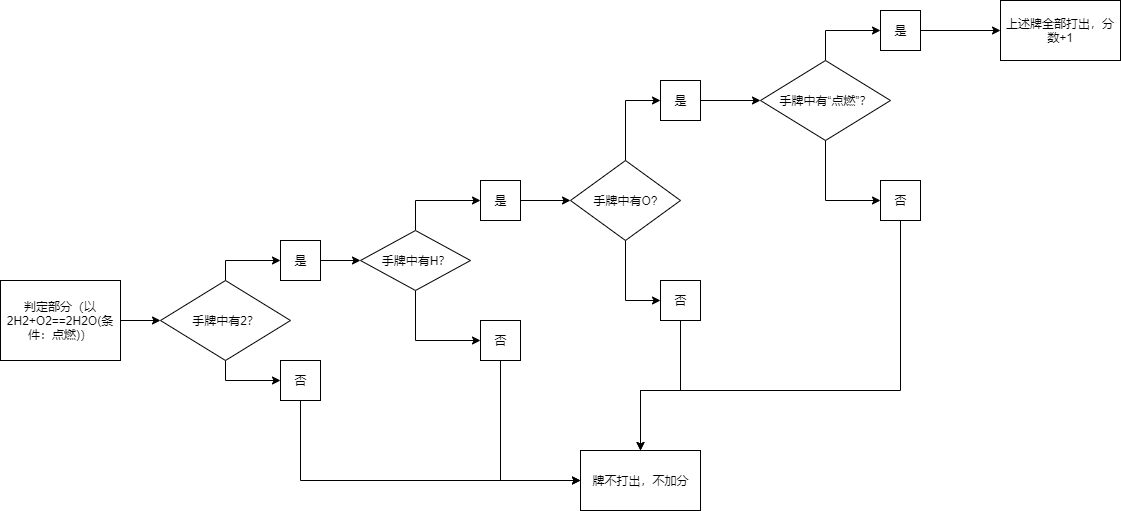
(2).弃牌函数

弃牌函数P1\_discard（P2\_discard）首先判断手牌数是否超过9张，若超过则进行弃牌（弃牌数P1\_AbandonNum为现牌数减9），弃牌时由于未完成手动弃牌模块故选择了随机弃牌。当弃牌数达到使现有手牌数恢复9张的程度，弃牌停止。



(3).分数判定函数

判定函数是由方程式集合中的所有方程式判定的结合体，代码量在全部模块中是最为庞大的。以水燃烧反应为例.程序依次判断手中是否有“2”、“氢”、“氧”、“点燃”，若有则打



出，分数加1；若不全则不打出，不加分。

(4).回放函数

回放函数(review)实际上是在基于EasyX创造的弹窗中输出一系列记录的数据以及数据对照表。首先在代码的前部定义两个结构体数组r1[5]、r2[5]，分别代表双方每回合的信息，其中包括回合手牌card1[13]、card2[13]以及分数score1、score2。包含弃牌和分数判定的Round\_P1（Round\_P2）函数中加入一个for循环，使代表手牌的P1[13]（P2[13]）中的数据转入card1[13]（card2[13]）中，完成记录。另外使游戏中的分数P1\_score（P2\_score）等于score1（score2），完成第二项记录。

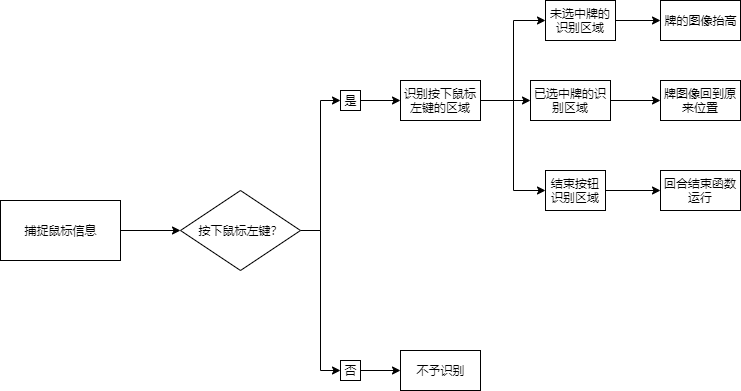
接下来review函数需要做的就是通过for循环将记录的手牌信息，分数以及打印出来即可。

(5).分数排名函数

设置一个包含人物姓名和人物分数的结构体，通过比较函数实现对结构体按成绩进行排序（按成绩进行排序），所有玩家的信息已经存储在abc.txt这个文件中，利用指针打开这个文件，对其进行读操作，读出所有玩家的信息，再按照分数由高到低排序，最后用fun函数输出已经排好序的人物信息，排名则得到展示。

(6).鼠标交互弃牌函数（待用）

此函数被编写出来本应用于手动弃牌模块，但由于一些原因手动弃牌模块未能被实现，现将手动弃牌函数进行展示。该函数识别鼠标左键并将选中的牌图像抬高，再次左键单击则牌图像缩回。同时界面上显示确定按钮，表示确认弃去哪些手牌。



5.代码实现

(1).抓牌函数

void P1\_drawele()

{

putimage(PromptBox\_x, PromptBox\_y, &img\_Promptbox);

setbkmode(TRANSPARENT);

settextstyle(30, 0, \_T("微软雅黑"));

outtextxy(PromptBox\_x + 20, PromptBox\_y, \_T("我方摸元素牌中！"));

FlushBatchDraw();

Sleep(1000);

int t;

int i, j;

for (i = 0; P1[i] != 0; i++)

{

}//找到牌尾

t = i;

for (j = t; j < t + 1; j++)

{

P1[j] = eleset[elenum];

eleset[elenum] = 0; //玩家抽两张牌，抽完牌堆处的数就清零

elenum++;

}

P1\_cardnum++;//加一

}

此处展示的为己方抓取元素牌的函数，其余函数大致与之形式相同。

(2).弃牌函数

void P1\_discard()

{

int a, t = 0;

if (disnum == 0)

{

if (P1\_cardnum > 9)

{

P1\_AbandonNum = P1\_cardnum - 9;

}

else

{

P1\_AbandonNum = 0;

}

P1\_AbandonNum\_once = P1\_AbandonNum;

}

if (P1\_cardnum > 9)

{

P1\_AbandonNum = P1\_cardnum - 9;

}

else

{

P1\_AbandonNum = 0;

}

while ((P1\_AbandonNum > 0) && (t < P1\_AbandonNum))

{

srand((unsigned)time(NULL));

a = rand() % 9;

P1[a] = 0;

P1\_set();

P1\_cardnum--;

t++;

}

putimage(PromptBox\_x, PromptBox\_y, &img\_Promptbox);

setbkmode(TRANSPARENT);

settextstyle(30, 0, \_T("微软雅黑"));

outtextxy(PromptBox\_x + 20, PromptBox\_y, \_T("我方弃牌中！"));

FlushBatchDraw();

Sleep(1000);

if (P1\_AbandonNum\_once > 0)

{

isplayer\_discard = 1;

}

if (P1\_AbandonNum\_once == 0)

{

isplayer\_discard = 0;

}

if (disnum >= P1\_AbandonNum\_once)//已经弃牌完成

{

isplayer\_discard = 0;

P1\_out = -2;

}

}

此处展示的为己方的弃牌函数

(3).分数判定函数

for (i = 0; i < P1\_cardnum; i++)

{

if (2 == P1[i])//手牌有2

{

P1[P1\_out] = 2;

P1\_cardnum--;

P1[i] = 0;

P1\_set();

for (j = 0; j < P1\_cardnum; j++)

{

if (13 == P1[j])//手牌有H

{

P1[P1\_out] = 13;

P1\_cardnum--;

P1[j] = 0;

for (k = 0; k < P1\_cardnum; k++)

{

if (14 == P1[k])//手牌有O

{

P1[P1\_out] = 14;

P1\_cardnum--;

P1[j] = 0;

for (l = 0; l < P1\_cardnum; l++)

{

if (8 == P1[l])//“点燃”

{

P1[P1\_out] = 8;

P1\_cardnum--;

P1[l] = 0;

P1\_score += 1;

}

else if (10 == P1[l])//“通电”

{

P1[P1\_out] = 10;

P1\_cardnum--;

P1[l] = 0;

P1\_score += 1;

}

……

此处展示的为氢的燃烧与水的电解方程式的判定，其他方程式的判定方式相同。

(4).回放函数

struct round1

{

int card1[13];

int discons1;//手动弃牌需要的参数，暂未使用

int discond1;//同上

int disele1;//同上

int score1;

}r1[5];

struct round2 {

int card2[13];

int discons2;

int discond2;

int disele2;

int score2;

}r2[5];//回放功能需要的结构体数组

int a;

for (a = 0; a < 13; a++)

{

r1[roundnum / 2].card1[a] = P1[a];

}

r1[roundnum / 2 ].score1 = P1\_score;//记录

roundnum++;

}

int a,b;

b = roundnum - 1;

for (a = 0; a < 13; a++) {

r2[b/2].card2[a] = P2[a];

}

r2[b / 2].score2 = P2\_score;

roundnum++;

这两部分是分别记录正反方数据的代码。

void review() {

initgraph(Width, Height);

setbkmode(TRANSPARENT);

int i;

char temp1[2][200];

char temp2[2][200];

char temp3[2][200];

char temp4[2][200];

char temp5[2][200];

char temp6[2][200];

char temp7[2][200];

char temp8[2][200];

char temp9[2][200];

char temp10[2][200];

outtextxy(10, 10, \_T("回合数：1"));

outtextxy(10, 30, \_T("己方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp1[0], \_T("%d "), r1[0].card1[i]);

outtextxy(10 + i \* 30, 50, temp1[0]);

}

\_stprintf\_s(temp1[1], \_T("分数：%d"), r1[0].score1);

outtextxy(10, 70, temp1[1]);

outtextxy(10, 90, \_T("回合数：2"));

outtextxy(10, 110, \_T("对方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp2[0], \_T("%d "), r2[0].card2[i]);

outtextxy(10 + i \* 30, 130, temp2[0]);

}

\_stprintf\_s(temp2[1], \_T("分数：%d"), r2[0].score2);

outtextxy(10, 150, temp2[1]);

outtextxy(10, 170, \_T("回合数：3"));

outtextxy(10, 190, \_T("己方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp3[0], \_T("%d "), r1[1].card1[i]);

outtextxy(10 + i \* 30, 210, temp3[0]);

}

\_stprintf\_s(temp3[1], \_T("分数：%d"), r1[1].score1);

outtextxy(10, 230, temp3[1]);

outtextxy(10, 250, \_T("回合数：4"));

outtextxy(10, 270, \_T("对方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp4[0], \_T("%d "), r2[1].card2[i]);

outtextxy(10 + i \* 30, 290, temp4[0]);

}

\_stprintf\_s(temp4[1], \_T("分数：%d"), r2[1].score2);

outtextxy(10, 310, temp4[1]);

outtextxy(10, 330, \_T("回合数：5"));

outtextxy(10, 350, \_T("己方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp5[0], \_T("%d "), r1[2].card1[i]);

outtextxy(10 + i \* 30, 370, temp5[0]);

}

\_stprintf\_s(temp5[1], \_T("分数：%d"), r1[2].score1);

outtextxy(10, 390, temp5[1]);

outtextxy(10, 410, \_T("回合数：6"));

outtextxy(10, 430, \_T("对方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp6[0], \_T("%d "), r2[2].card2[i]);

outtextxy(10 + i \* 30, 450, temp6[0]);

}

\_stprintf\_s(temp6[1], \_T("分数：%d"), r2[2].score2);

outtextxy(10, 470, temp6[1]);

outtextxy(10, 490, \_T("回合数：7"));

outtextxy(10, 510, \_T("己方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp7[0], \_T("%d "), r1[3].card1[i]);

outtextxy(10 + i \* 30, 530, temp7[0]);

}

\_stprintf\_s(temp7[1], \_T("分数：%d"), r1[3].score1);

outtextxy(10, 550, temp7[1]);

outtextxy(10, 570,\_T("回合数：8"));

outtextxy(10, 590, \_T("对方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp8[0], \_T("%d "), r2[3].card2[i]);

outtextxy(10 + i \* 30, 610, temp8[0]);

}

\_stprintf\_s(temp8[1], \_T("分数：%d"), r2[3].score2);

outtextxy(10, 630, temp8[1]);

outtextxy(10, 650, \_T("回合数：9"));

outtextxy(10, 670, \_T("己方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp9[0], \_T("%d "), r1[4].card1[i]);

outtextxy(10 + i \* 30, 690, temp9[0]);

}

\_stprintf\_s(temp9[1], \_T("分数：%d"), r1[4].score1);

outtextxy(10, 710, temp9[1]);

outtextxy(500, 10, \_T("回合数：10"));

outtextxy(500, 30, \_T("对方手牌："));

for (i = 0; i < 13; i++) {

\_stprintf\_s(temp10[0], \_T("%d "), r2[4].card2[i]);

outtextxy(500 + i \* 30, 50, temp10[0]);

}

\_stprintf\_s(temp10[1], \_T("分数：%d"), r2[4].score2);

outtextxy(500, 70, temp10[1]);

outtextxy(500, 130, \_T("数字对照表"));

outtextxy(500, 150, \_T("2--数字2 3--数字3 4--数字4 5--数字5 34--数字6 35--数字8 （奇怪的编码敬请谅解）"));

outtextxy(500,170,\_T("6--加热 7--高温 8--点燃 9--催化剂 10--通电 11--放电 12--高压"));

outtextxy(500, 190, \_T("13--氢 14--氧 15--碳 16--氮 17--钠 18--铝 19--硅 20--硫 21--氯 22--钾"));

outtextxy(500, 210, \_T("23--钙 24--锰 25--铁 26--铜 27--氟 28--镁 29--锌 30--磷 31--溴 32--碘 33--铅"));

getchar();

closegraph();

}

这是最终显示界面的打印部分。

(5).分数排名函数

struct RenWu {//每一个人物玩家

char ch[100];//人物姓名

int ll;//人物对应分数 ，指针指向分数

};

bool cmp(RenWu x, RenWu y) {//排序的方式

return x.ll > y.ll;

}

RenWu a[100];

所需要的结构体和数组。

void fun(RenWu lll[100], int n) {//将排名显示出来

for (int i = 0; i < n; i++)

{

cout << lll[i].ch;

cout << " ";

cout << lll[i].ll;

if (i != n - 1)

cout << endl;

}

return;

}

void ranking()

{

FILE\* p;

p = fopen("abc.txt", "r");

if (p == NULL)

{

cout << "错误" << endl;

}

int i = 0;

while (!feof(p))

{

fscanf(p, "%s", a[i].ch);

fscanf(p, "%d", &a[i].ll);

i++;

}

sort(a, a + i, cmp);

fun(a, i);

}

这里是排名函数。

(6).鼠标交互弃牌函数

void interactive()

{

int i;

MOUSEMSG m;

if (MouseHit()) //有鼠标消息时执行

{

m = GetMouseMsg();

if (m.uMsg == WM\_LBUTTONDOWN)

{

for (i = 0; (i < P1\_cardnum) && (!isselected); i++)

{

if ((m.x >= P1\_x[i] && m.x <= P1\_x[i] + P1\_width) && (m.y >= P1\_y[i] && m.y <= P1\_y[i] + P1\_height))

{

P1\_out = i;

P1\_y[i] -= 20;//突出

isselected = 1;

goto end0;

}

}

if ((isselected) && (m.x >= P1\_x[P1\_out] && m.x <= P1\_x[P1\_out] + P1\_width) && (m.y >= P1\_y[P1\_out] && m.y <= P1\_y[P1\_out] + P1\_height))

{

P1\_y[P1\_out] += 20;//缩回

P1\_out = -1;

isselected = 0;

}

if ((isselected) && (isAbletoPlaycard) && (m.x >= firm\_x && m.x <= firm\_x + firm\_width) && (m.y >= firm\_y && m.y <= firm\_y + firm\_height))

{

isEffective = 1;

P1\_y[P1\_out] += 20;

isselected = 0;

}

if ((m.x >= roundover\_x && m.x <= roundover\_x + roundover\_width) && (m.y >= roundover\_y && m.y <= roundover\_y + roundover\_height))

{

if (isRoundover == 0)

{

isselected = 0;

isRoundover = 1;

//回合结束

}

else if ((isRoundover == 1) && (disnum == 0)) //不想弃牌了

{

isplayer\_discard = 0; //不弃牌，图框消失

isselected = 0;

isRoundover = 0;

}

}

}

end0:;

}

}

6.效果测试

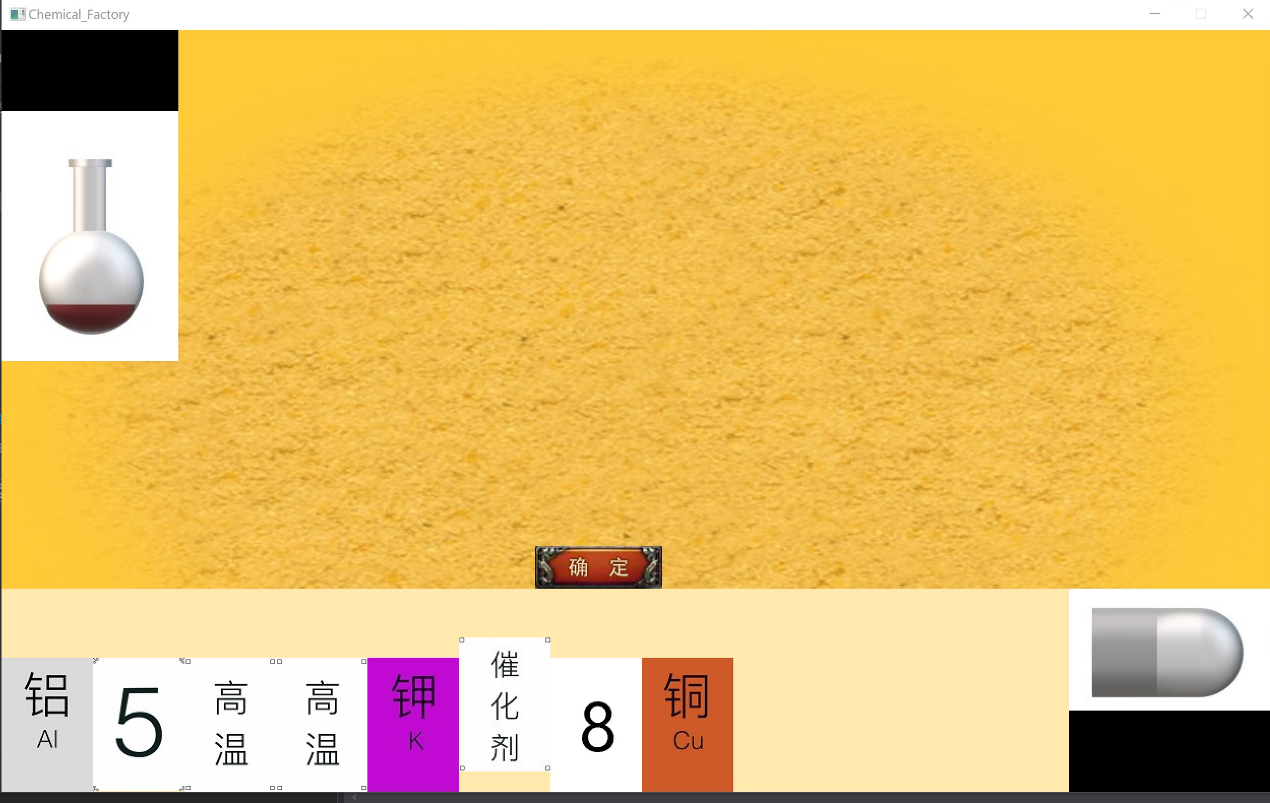
初始界面



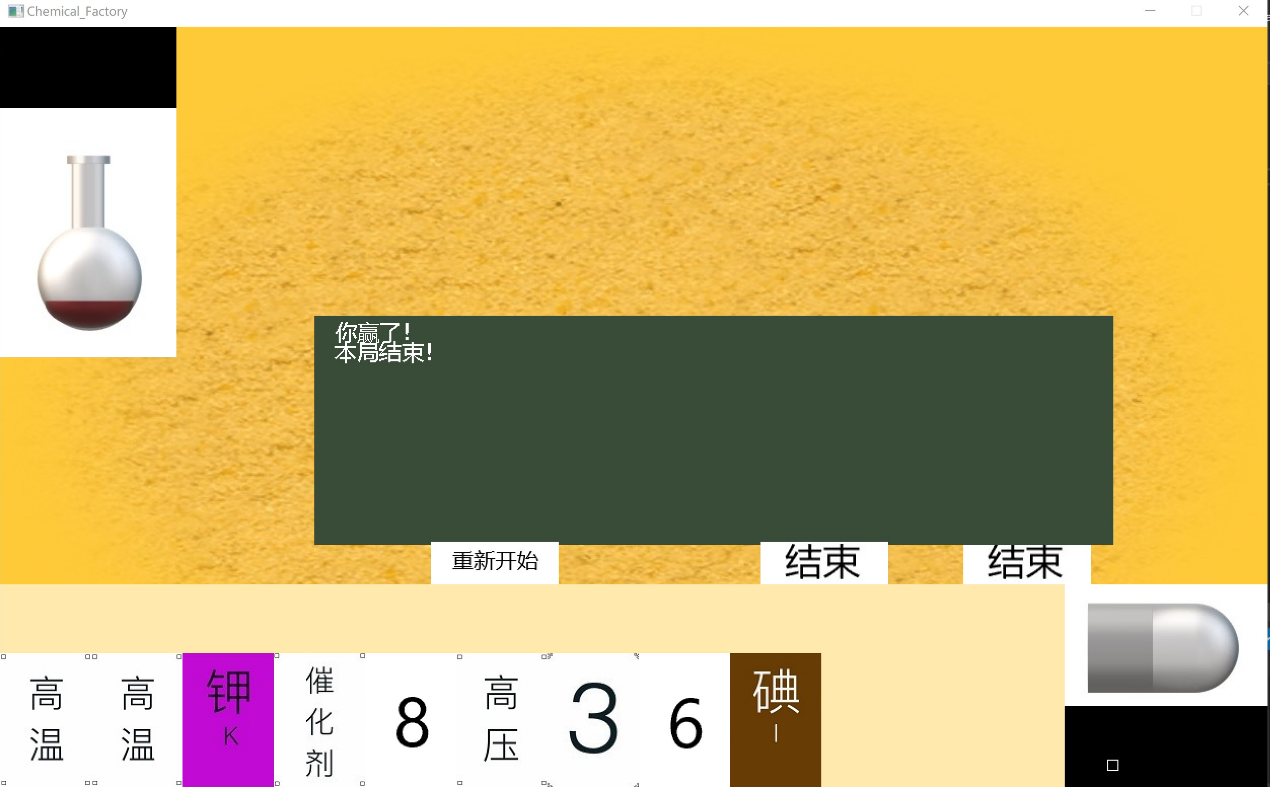
摸牌界面



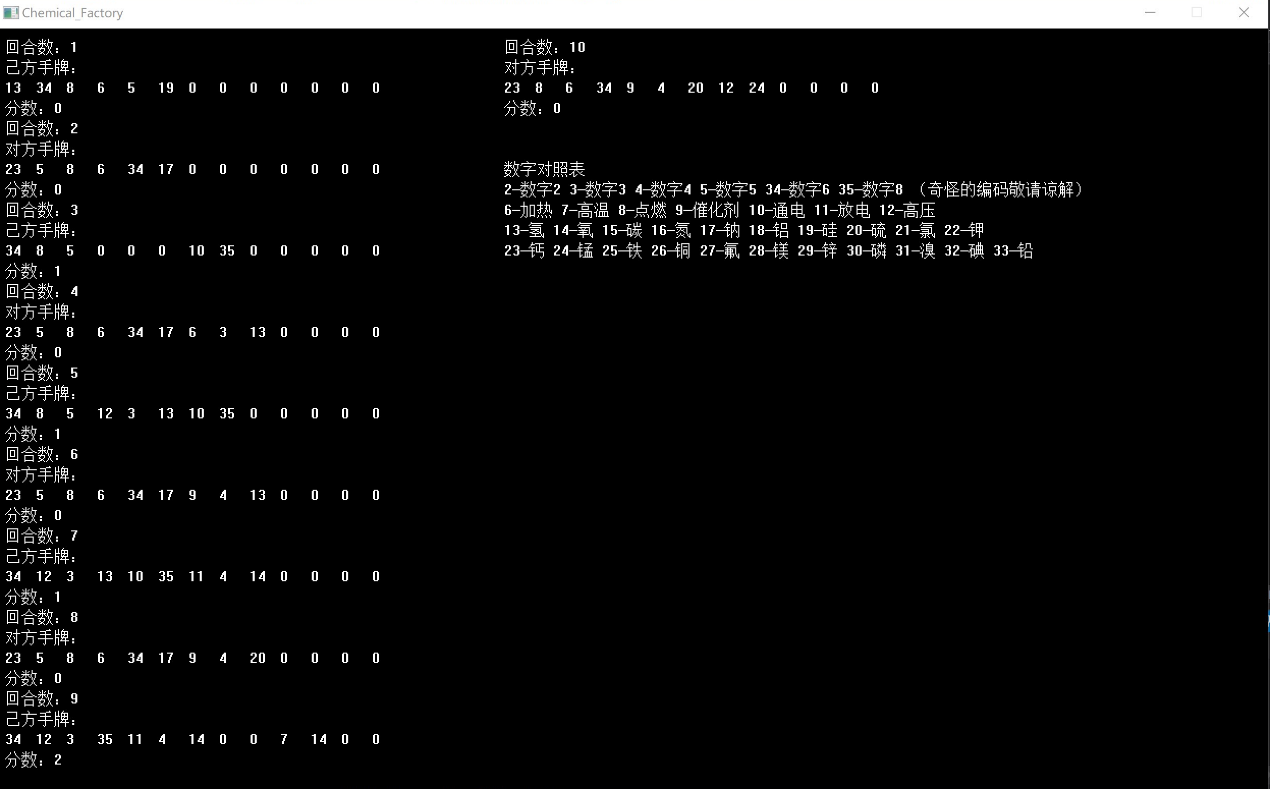
手动选择界面（暂无弃牌功能）



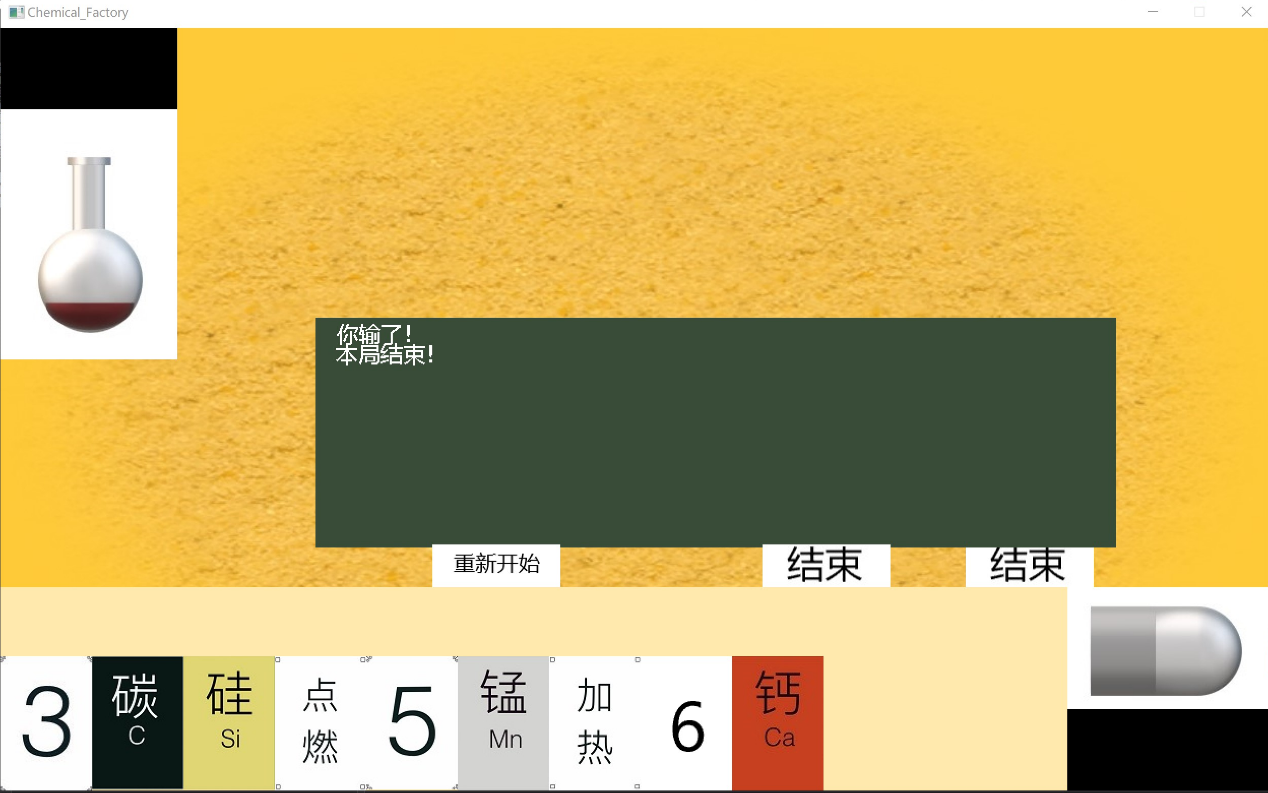
结束界面（胜利）



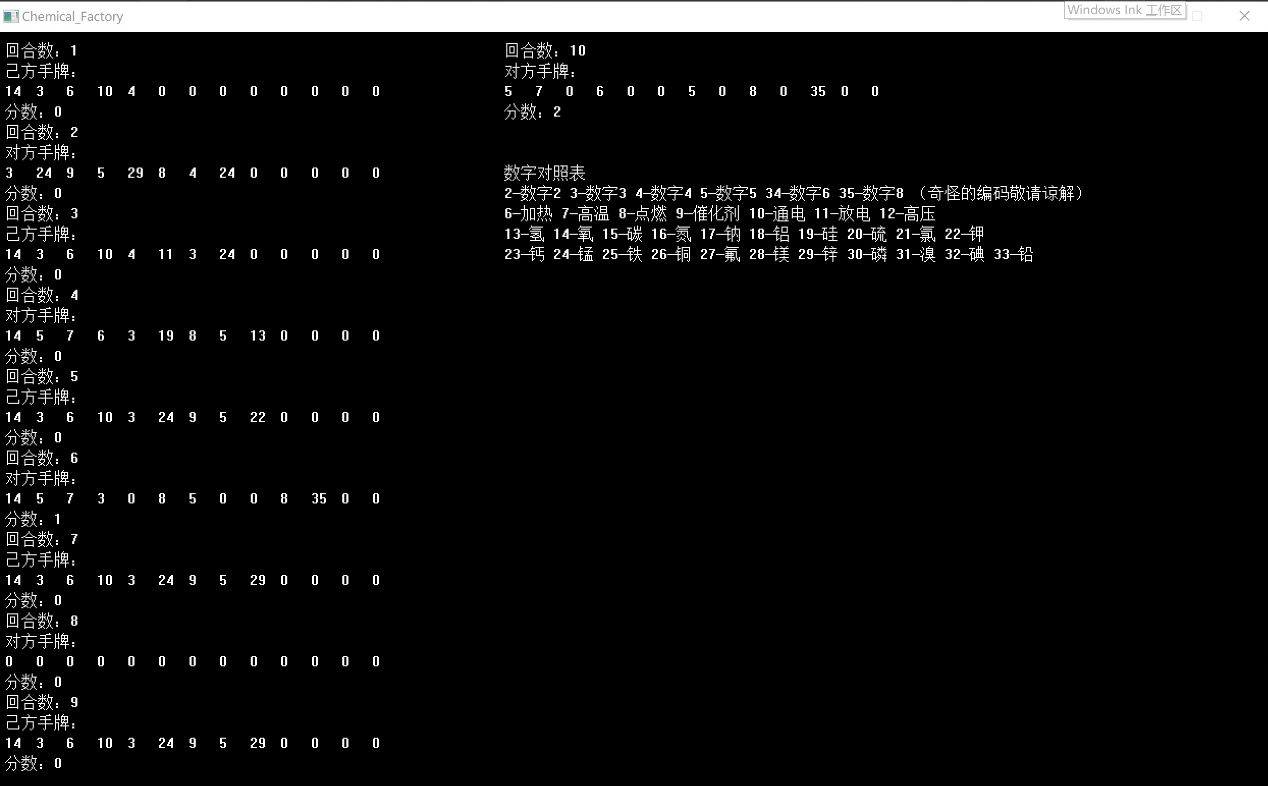
胜利后数据回放：对方分数为0，己方分数为2



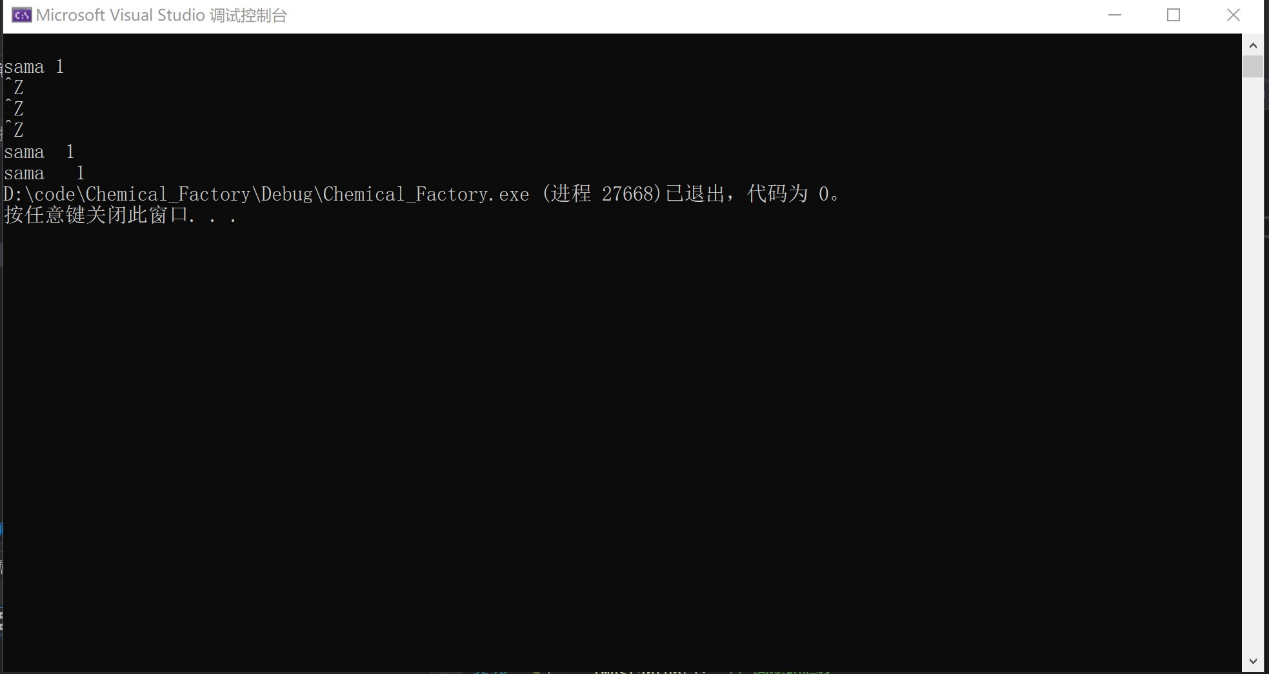
结束界面（失败）



失败后数据回放：对方分数为2，己方分数为0



排名界面



7.结语

本学期进行的C语言项目瑕疵甚至于影响游戏体验之处尚存。最大的遗憾便是小组最终并未完成流畅的手动弃牌流程的编写，在尝试当中出现了逻辑问题，导致空有选牌函数，无法将其应用在游戏中。但小组在大体上实现了预期的界面效果，进行了数千行代码的编辑并构造出了可以运行的游戏进程，以本小组成员的C语言能力而言也算是取得了可观的结果。同时我们也体会到了C语言在框架构建时的难度与考验，习得了相关的知识，本学期的学习未留大的遗憾。项目与报告的诸多不足之处，还请老师包涵。

笔者 杨竞翔