**C语言扫雷工程代码报告**

组员：陈诗翰，张倬豪，易鸣奇。

**1、总体功能描述**

《扫雷》是一款大众类的益智小游戏，游戏目标是在最短的时间内根据点击格子出现的数字找出所有非雷格子，同时避免踩雷。

游戏的基本操作包括左键单击（Left Click）、右键单击（Right Click）、右键单击两次三种。其中左键用于打开安全的格子，推进游戏进度；右键用于标记地雷，以辅助判断。

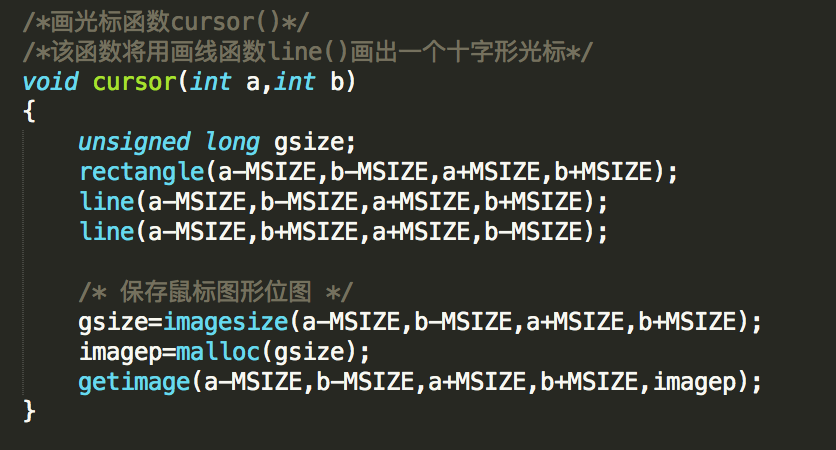
左键单击：在判断出不是雷的方块上按下左键，可以打开该方块。如果方块上出现数字，则该数字表示其周围3×3区域中的地雷数（一般为8个格子，对于边块为5个格子，对于角块为3个格子。所以扫雷中最大的数字为8）；如果方块上为空（相当于0），则可以递归地打开与空相邻的方块；如果不幸触雷，则游戏结束。

右键单击：在判断为地雷的方块上按下右键，可以标记地雷（显示为小红旗）。重复一次或两次操作可取消标记（如果在游戏菜单中勾选了“标记(?)”，则需要两次操作来取消标雷）

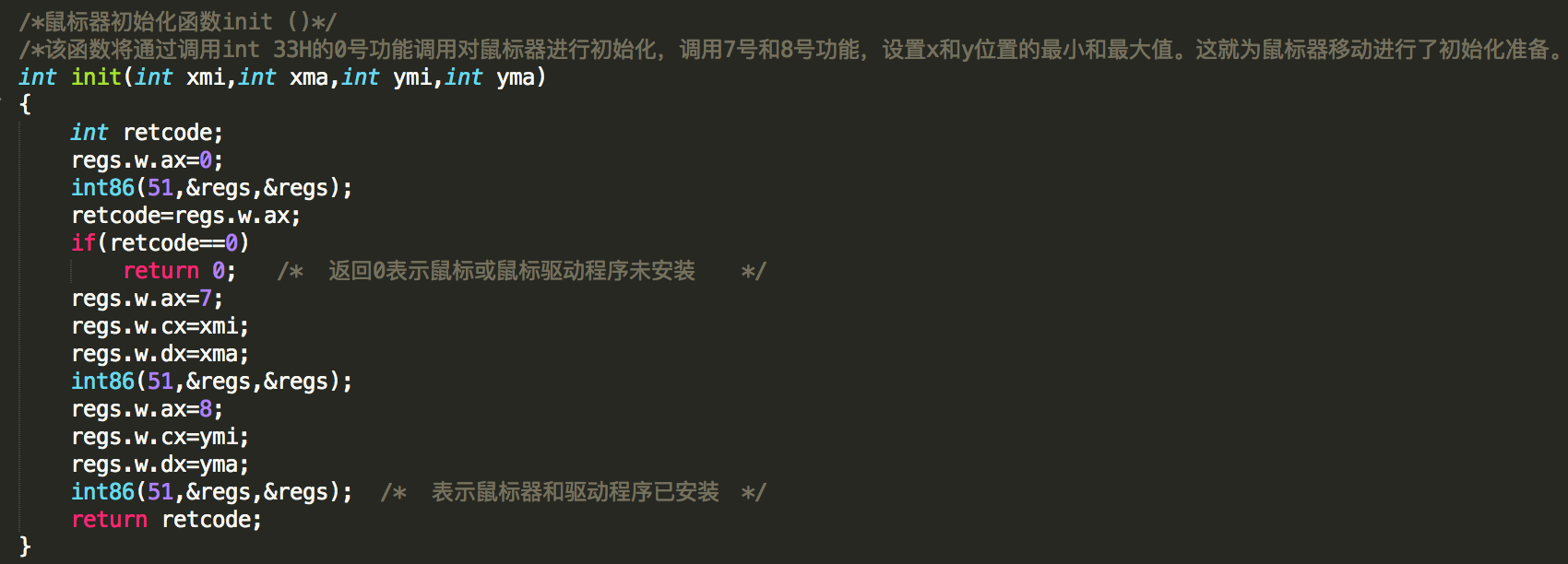
本程序还实现了菜单界面以及具体功能，如新游戏、记录最佳成绩、帮助等。开场动画，爆炸音效，结尾音乐等功能。

**2、关键设计思路方法**

**（1）鼠标操作算法及实现**



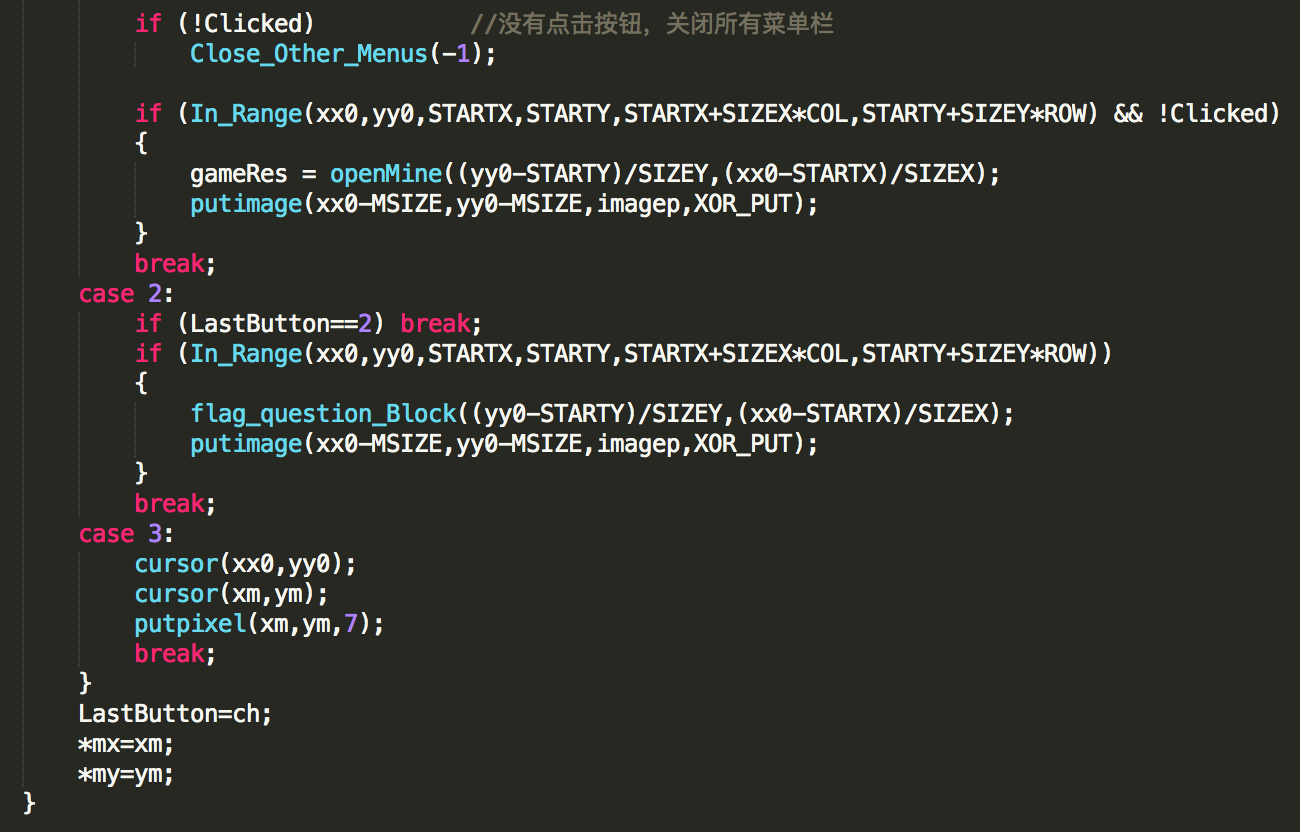
画光标函数cursor().该函数将用画线函数line()画出一个十字形光标



鼠标器初始化函数init ().该函数将通过调用int 33H的0号功能调用对鼠标器进行初始化，调用7号和8号功能，设置x和y位置的最小和最大值。这就为鼠标器移动进行了初始化准备。由于0号功能调用是测试鼠标驱动程序是否安装，因此在运行该程序前必须首先执行鼠标驱动程序mouse.com，若调用该函数执行了0号功能调用，当返回值为0时(即返回参数为0)，表示未安装成功，这可能是鼠标器或驱动程序末安装。这时程序将显示Mouse or Mouse Driver Absent，并回到系统。

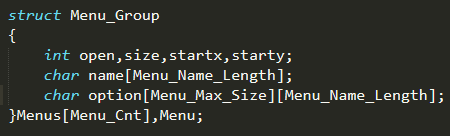


读鼠标的位置和按钮状态函数read().该函数将通过调用int 33H的3号功能调用，读鼠标的位置和按钮状态。鼠标的x、y位置值将由指针mx和my给出，而按钮状态则由mbutt指针给出。

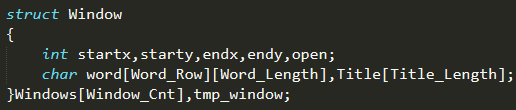


鼠标操作最重要的函数，读入并实现鼠标操作，具体见图中注释或附录源代码。

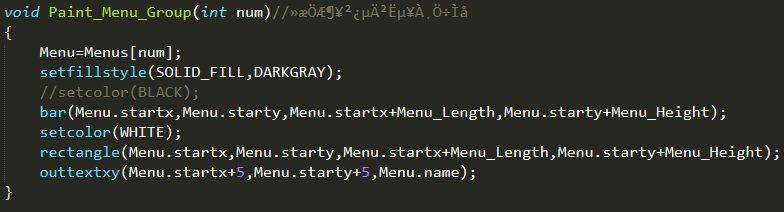
**（2）菜单相关算法及实现**



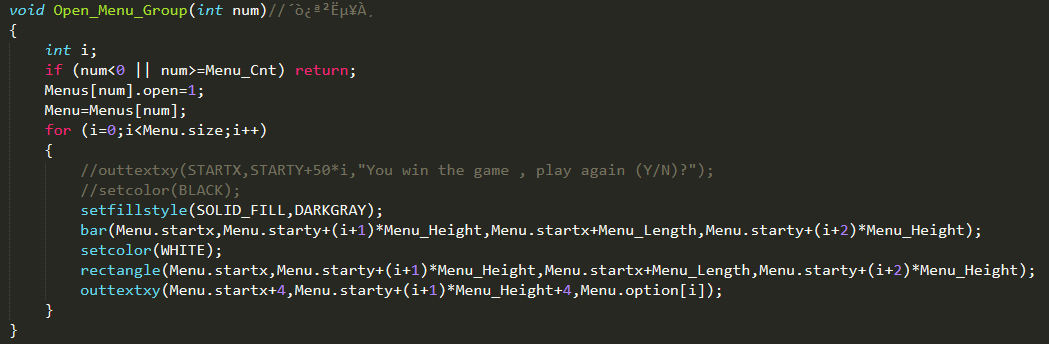
Menu\_Group定义了下拉的一个菜单，open表示是否打开，startx和starty表示了右上角的位置。Name表示菜单的名字，而option表示了各个选项的名字。



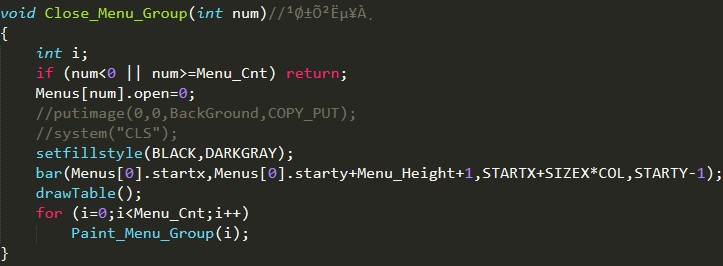
Window定义了一个弹出的窗口（如帮助等）。Open、startx等变量与Menu\_Group中的定义类似。Word存储了将要显示的文字，而Title储存了窗口的名字。在本程序中，Windows中定义的3个Window类，分别代表记录、帮助和游戏相关。



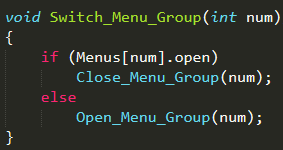
Paint\_Menu\_Group(num)将第num个菜单的主体画出(就是点击能控制菜单开关的那个按钮)。使用了rectangle和bar两个绘图函数，使用outtextxy输出菜单的名字。



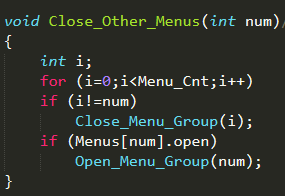
Open\_Menu\_Group这个函数负责打开一个菜单。我们将菜单内的选项以与之前类似的方式依次画出即可。



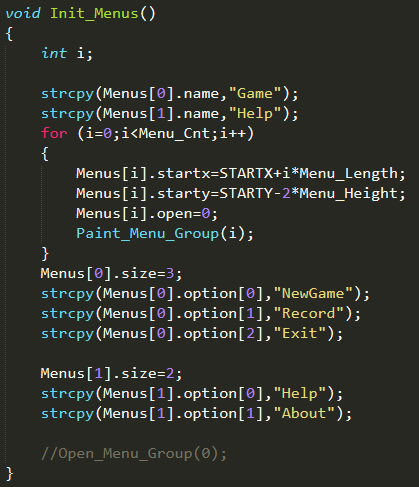
Close\_Menu\_Group这个函数负责关闭一个菜单。我们实现的方式是用一个黑色的矩形覆盖弹出的菜单所处的位置。为了避免可能出现的显示问题，我们在画了矩形之后重新绘制菜单栏和雷区。



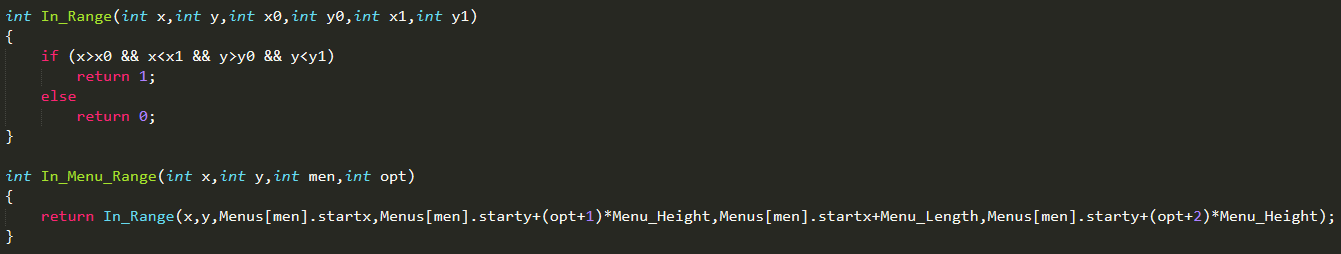
Switch\_Menu\_Group负责切换一个菜单的状态。原先是开着的，我们就关闭它，原先是关闭的，我们就打开它。



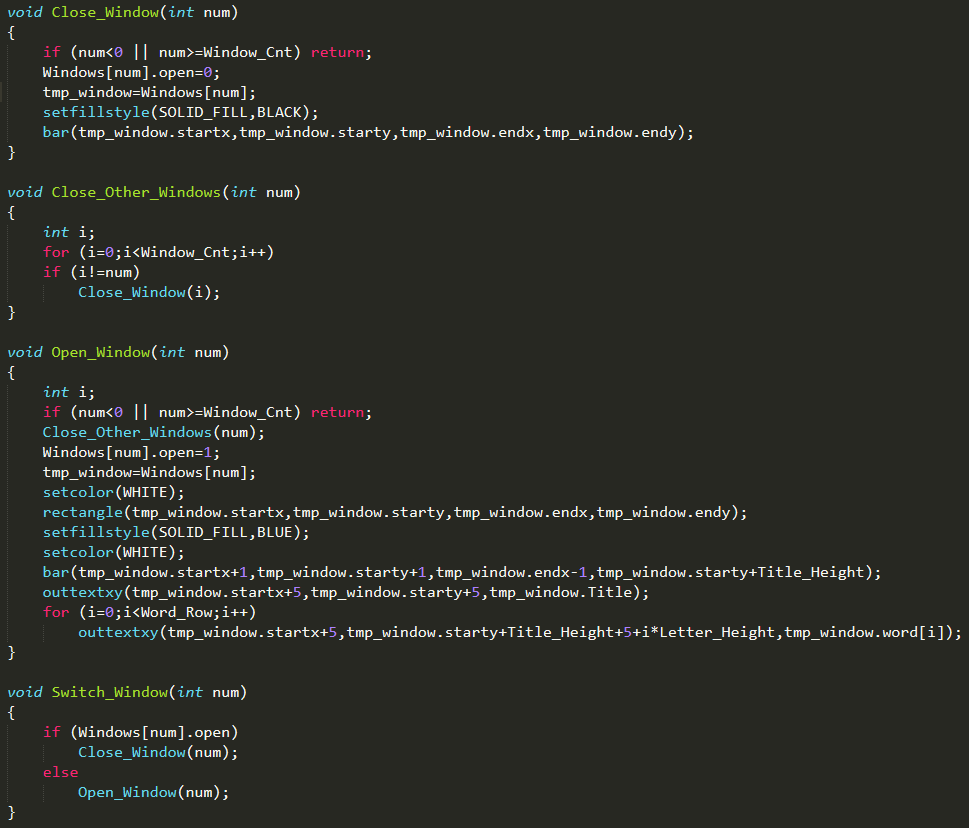
Close\_Other\_Menus负责关闭除了一个特定菜单之外的所有菜单。由于我们关闭的方式（黑色矩形）比较简单粗暴，因此为了避免显示问题，如果特定菜单是开着的，我们重新绘制一下它内部的选项。



Init\_Menus负责初始化菜单的名字等属性。如果之后需要改动功能，在这里就可以比较方便地修改。



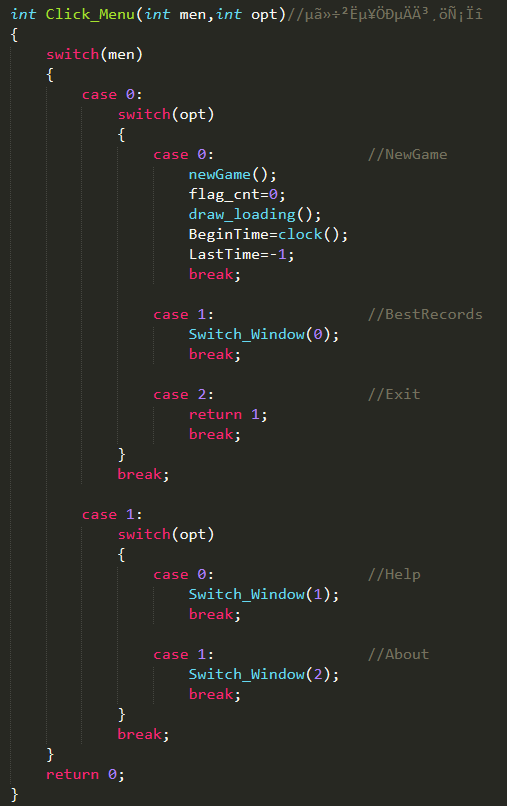
这是两个辅助函数。我们在处理点击时，经常需要判断鼠标和一个给定矩形的位置关系，因此用这两个函数来简化代码。



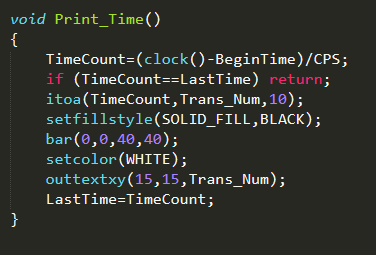
有关窗体开闭的一些函数，和之前菜单的几个函数非常类似，因此在此不多加赘述。



窗体的初始化。我们使用了比较简单粗暴的方式，将想要显示的内容依次strcpy了进去。



当我们点击了菜单中的某个选项时，使用switch语句进行分支选择，进行相应的操作。而这个函数的返回值代表是否直接结束游戏，以此来实现exit的功能。

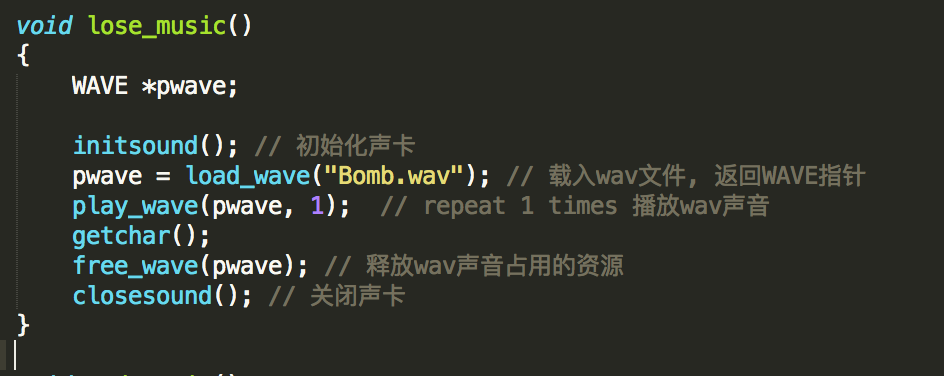


这个函数实现了计时并输出。为了防止一秒钟内多次重复输出而导致数字闪动，我们判断了时间（精确到秒）是否发生了变动。

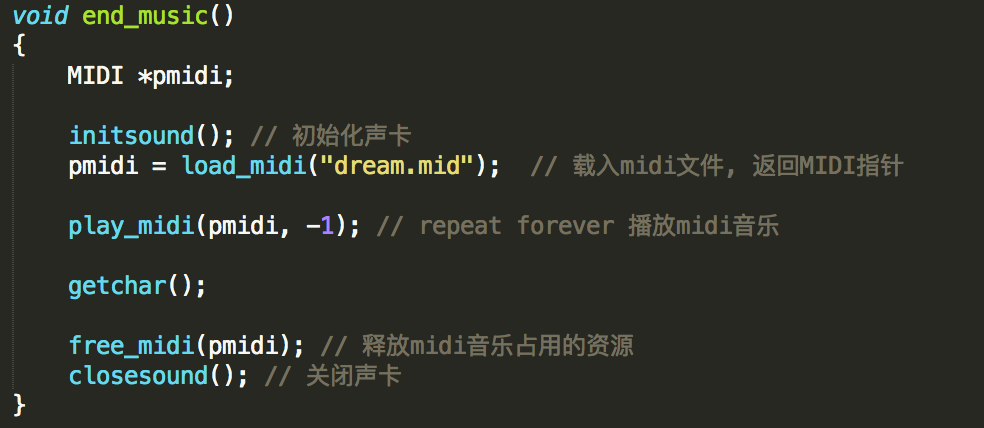


在程序的主循环（即循环判断鼠标状态）中，我们每次判断是否有菜单及其内部选项被点击，如果有，则进行相应的操作。如果没有而鼠标在雷区范围内，则再进行方块的打开。就这样，我们实现了简单的菜单。

（3）音乐及动画相关算法及实现

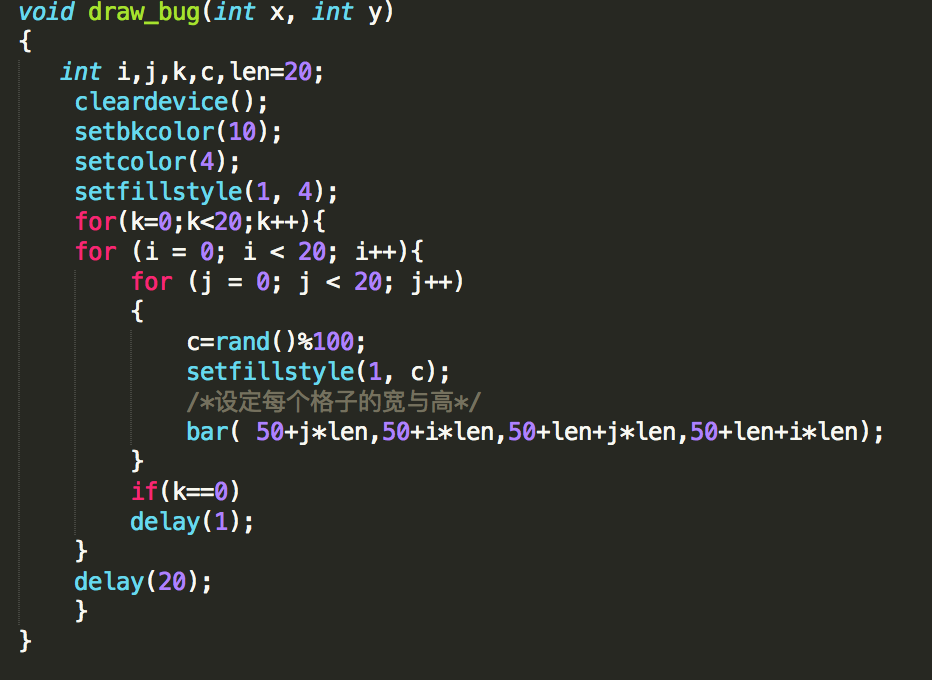


点到雷时发出爆炸声主要通过load\_wave与play\_wav这两个函数实现。在初始化声卡之后，载入wav文件，返回pwave指针，通过play\_wave设定播放次数（1次）并实现播放功能，最后释放wav所占资源并关闭声卡。最后在miner.c的confirm函数中在re==1的条件分支下调用lose函数，实现音乐播放功能。



该函数的实现与爆炸声的实现向类似，只是指针类型由WAVE\*变为MIDI\*，load\_wave与play\_wave变为load\_midi与play\_midi。由于结尾音乐的时间过长，若用wav格式则音乐文件太大不能成功播放。该条函数在miner函数最后跳出do while循环时调用。

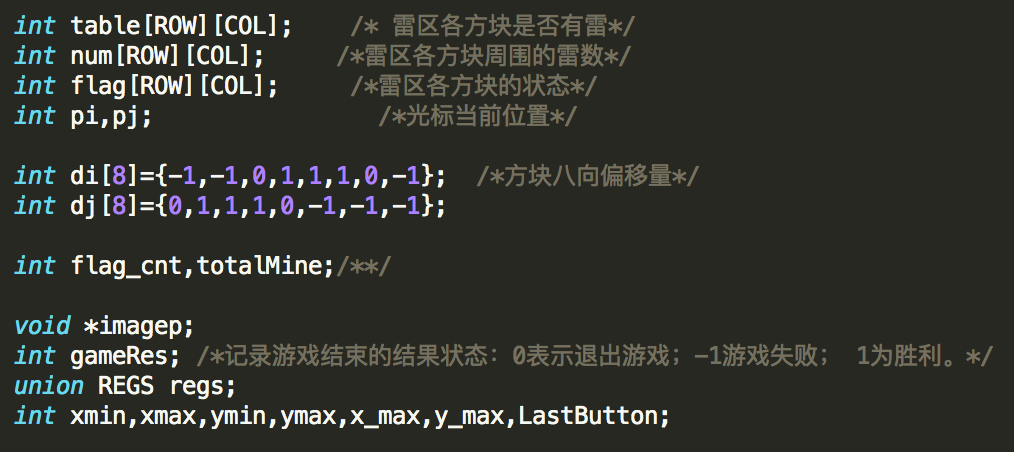




动画的整体效果为20\*20方块的颜色转换，参考扁平化的画面风格，利用视觉延缓形成一种色彩流动的效果。方框整体的大小与扫雷游戏界面中每个方块的大小位置相同，使得画面切换不会显得太突兀。

该函数的核心为随机，在合适的位置构造方块群后，通过rand使方块颜色随机刷新以实现上述功能。

**（4）扫雷内部算法及实现**



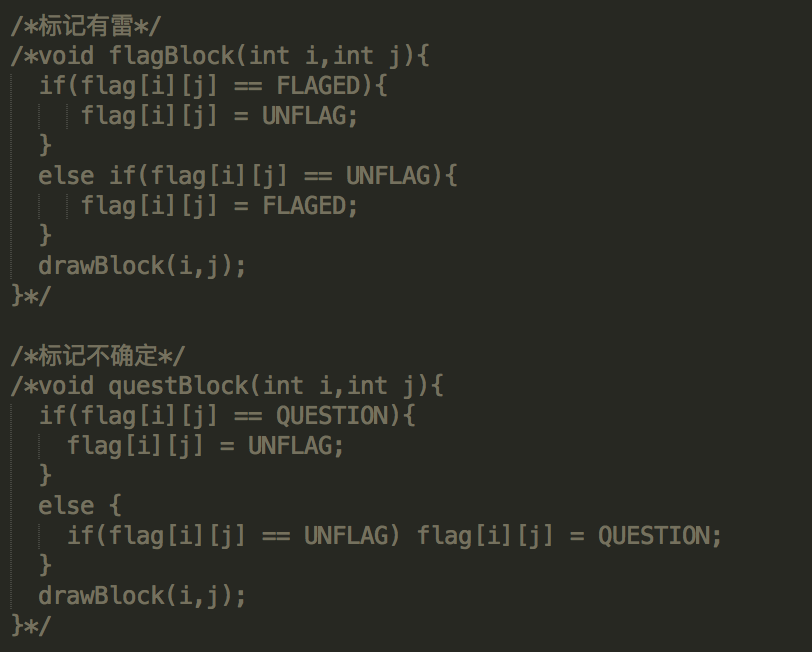
数据结构实现。具体见图中注释。



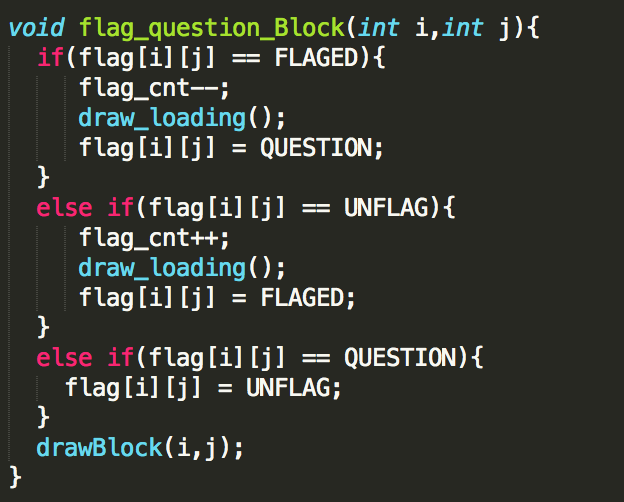
随机生成地雷的分布。

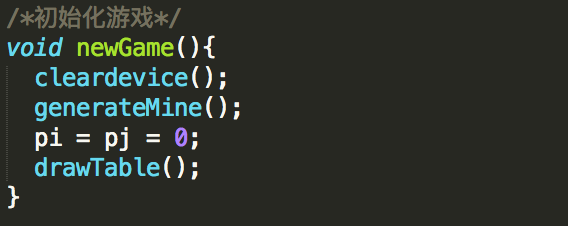


绘制小方块。

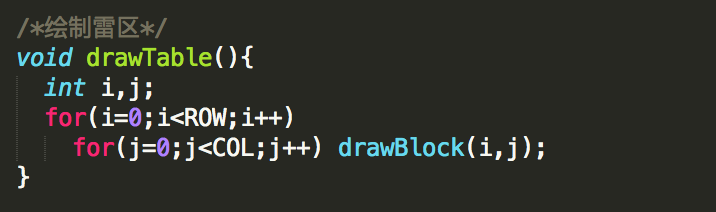


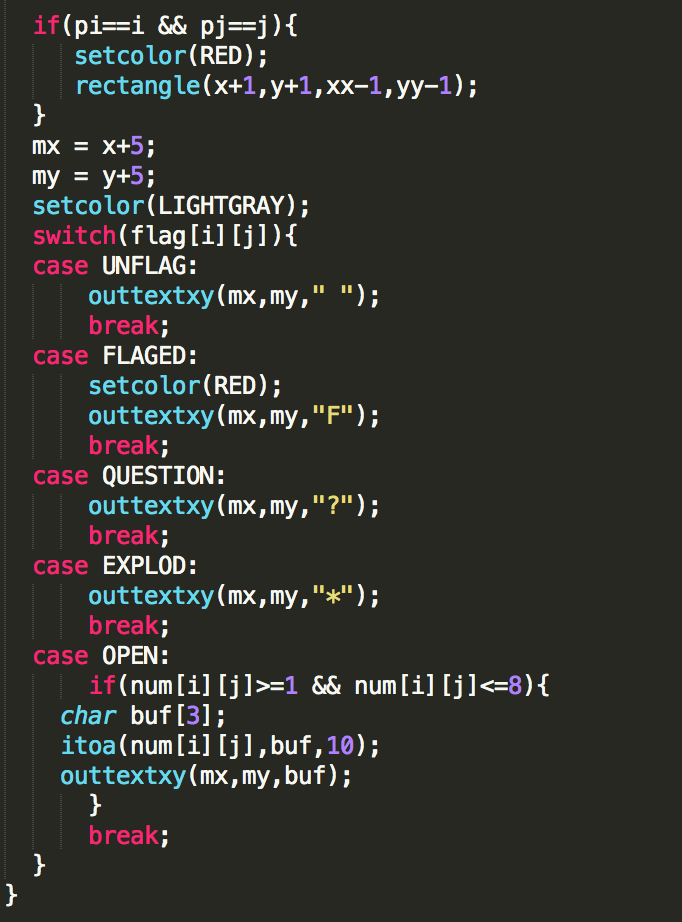
原来的键盘操作中，标记有雷和不确定分别是输入不同的字符，现在改成鼠标操作以后，这两个函数应当合并成以下函数：



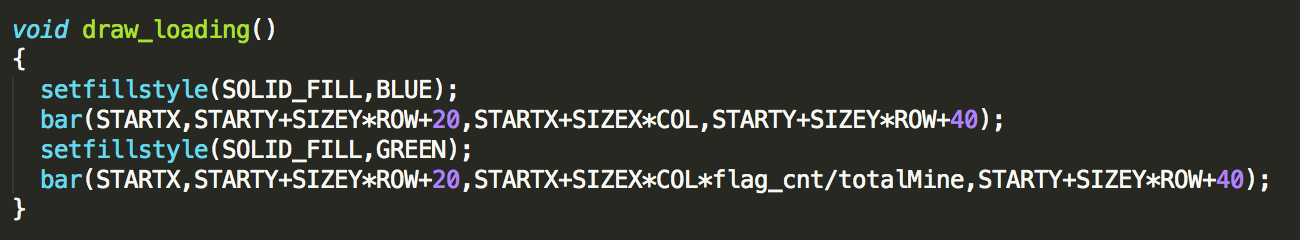


初始化游戏

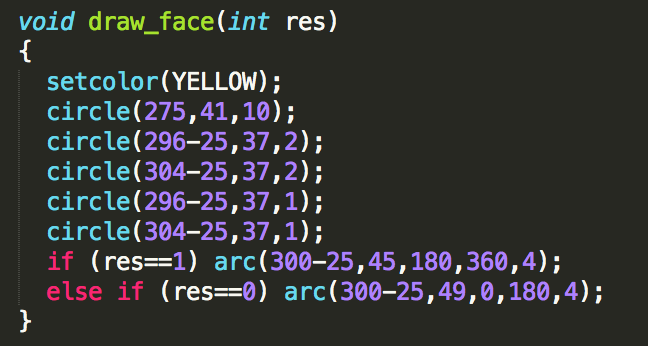




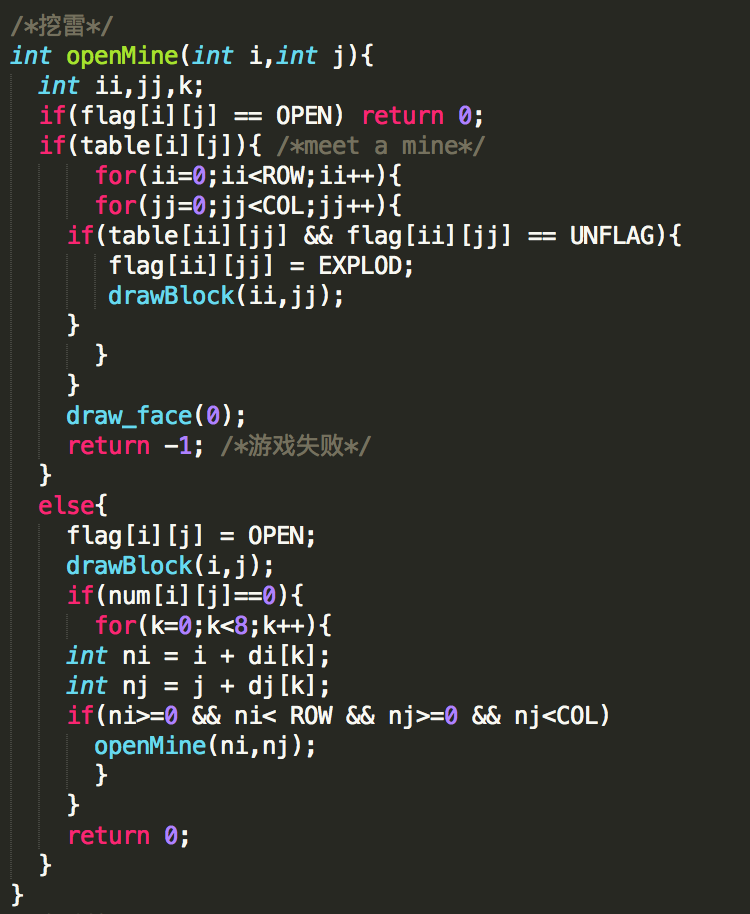
绘制雷区。



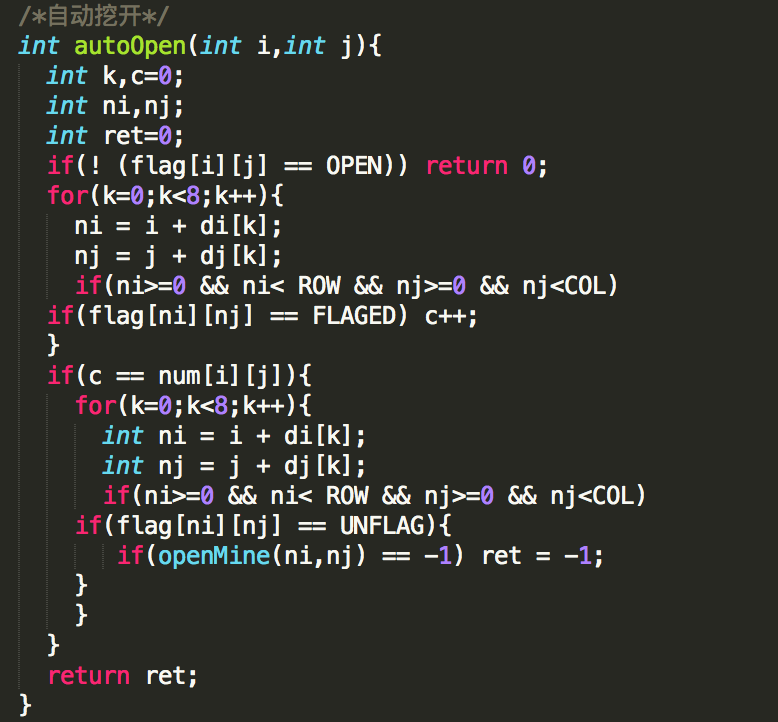
显示扫雷进度状态栏。



画笑脸和哭脸。

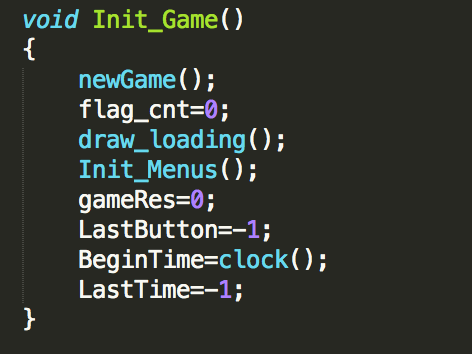


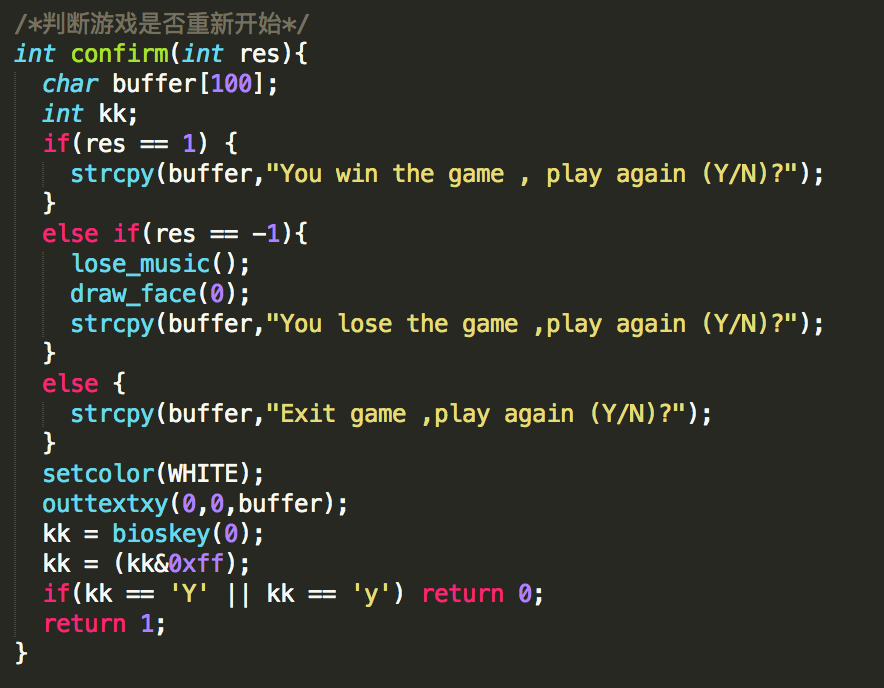
挖雷函数。



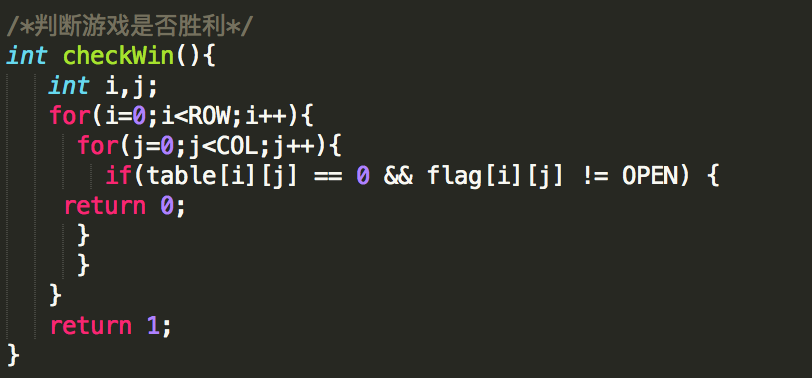
自动挖雷

（5）主函数及其实现





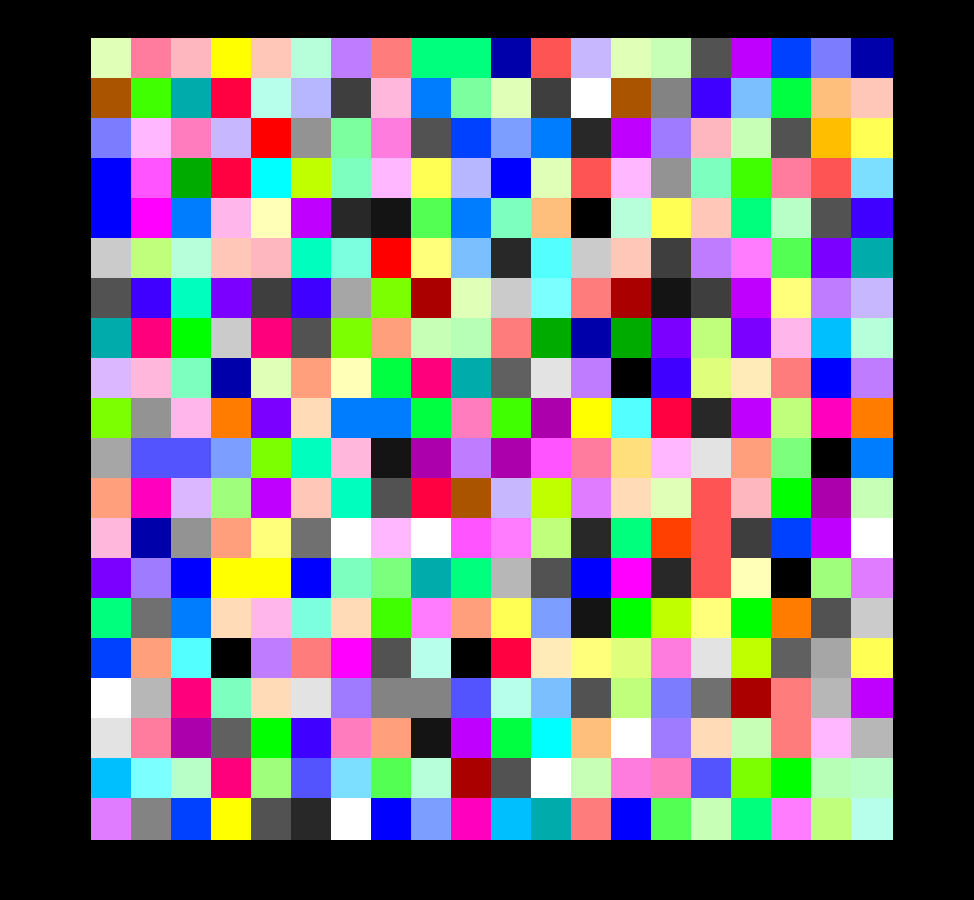




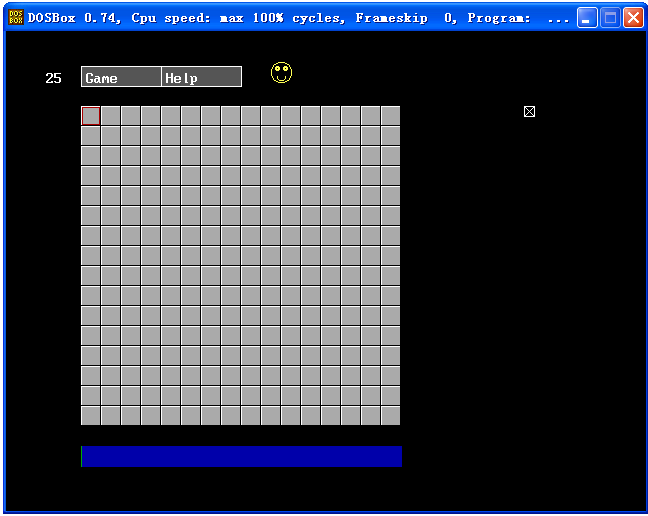


1. **测试报告**

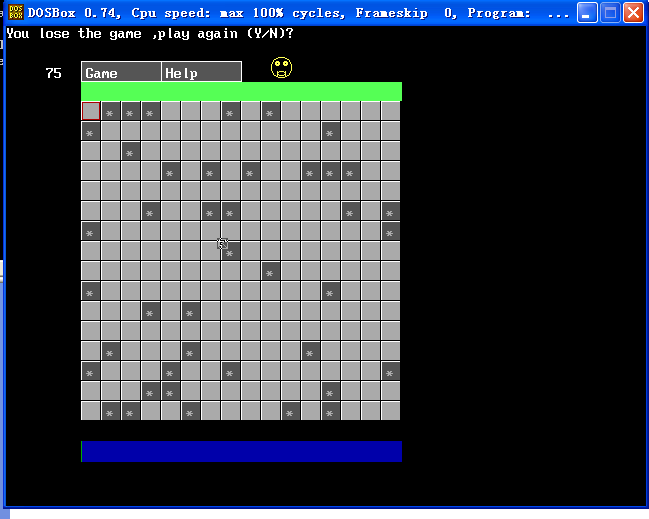
如图所示：



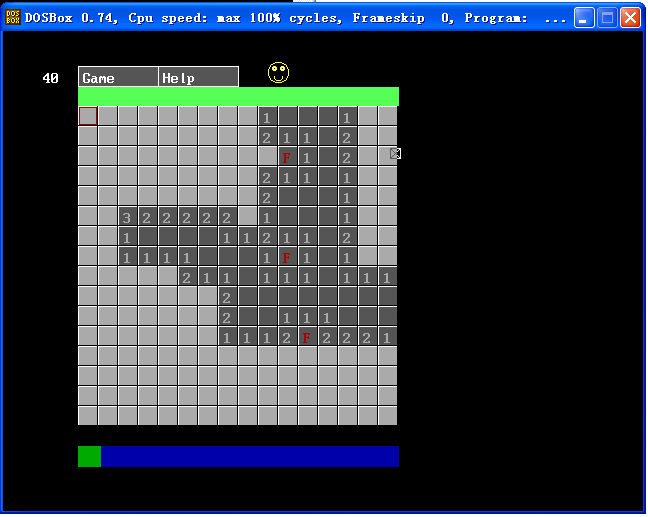
此为动画界面



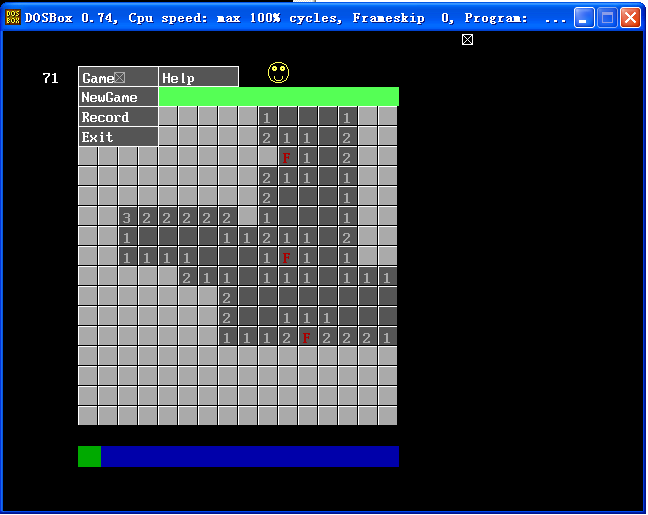
此为游戏开始界面

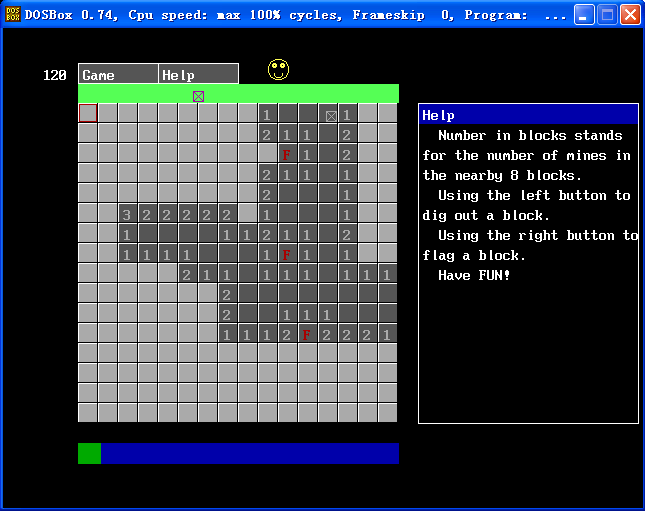
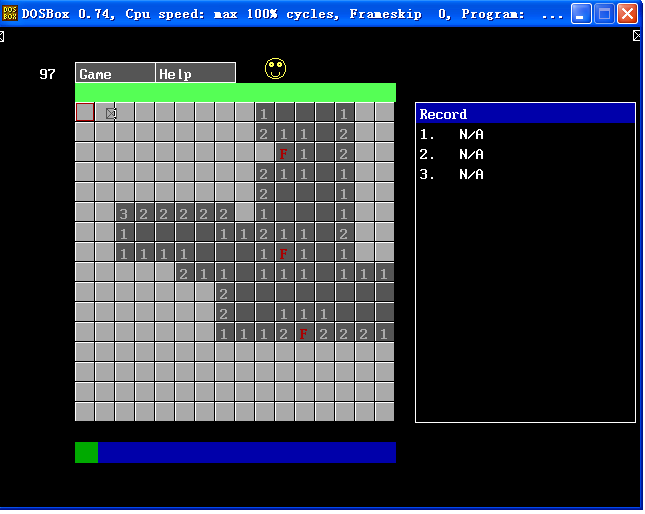


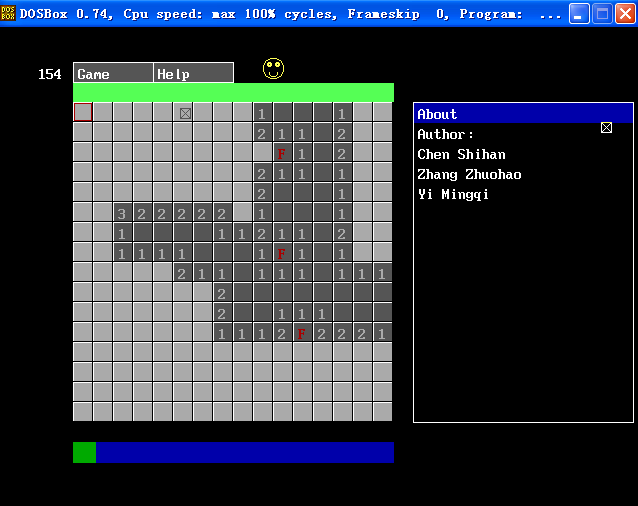
此为游戏结束画面



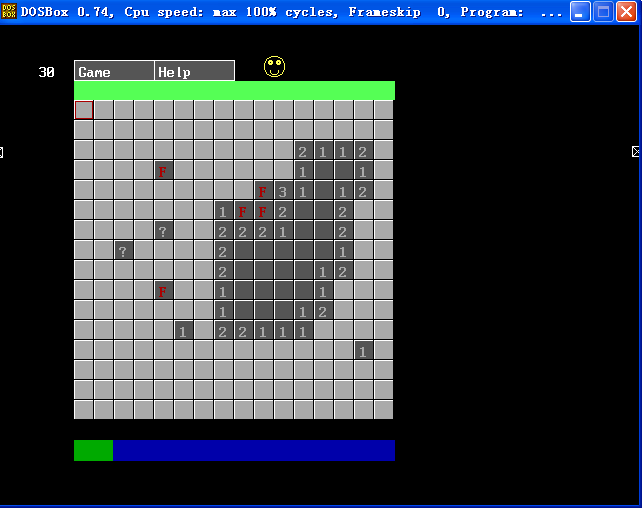
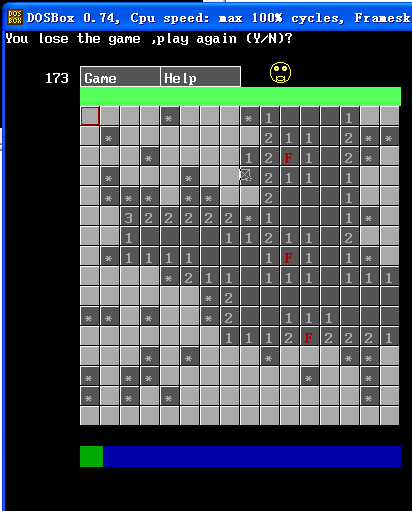
此为右键标记画面。







此为菜单画面



其他画面（触雷死亡、问号标记等。）

**4、开发体会**

感受到了真正做一个工程的困难，做完这一个工程不知道有多少个大大小小的bug，这么多个日日夜夜慢慢调试，最后慢慢成型，我们的不管是代码能力还是团队协作能力都得到了很大的提高。

**5、人员职责及互评**

指出小组相关人员的角色和职责，包括名称、对应人员姓名以及职责。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 角色 | 姓名 | 学号 | 职责 | 打分 |
| 组长 | *陈诗翰* | 3150103690 | * 菜单等相关函数实现 * 测试 * 撰写《project报告》 | 3 |
| 组员 | *张倬豪* | 3150102418 | * 鼠标等代码实现 * 测试 * 撰写《project报告》 | 2 |
| 组员 | *易鸣奇* | 3150102390 | * 音乐动画等代码实现 * 测试 * 撰写《project报告》 | 1 |

**6、附录**

源代码：

**1、main.c**

#include "miner.h"

int main()

{

int x,y,xx=10,yy=10,buttons; /\* (x,y)鼠标当前位置 (xx,yy)鼠标前一位置 \*/

x\_max=639;

y\_max=479;

initGraph(); /\*初始化图形显示方式\*/

cartoon();

Init\_Windows();

do {

Init\_Game();

xmin=2;

xmax=x\_max-1;

ymin=8;

ymax=y\_max-2;

if(init(xmin,xmax,ymin,ymax)==0 ) /\* 调用init函数对鼠标器初始化\*/

{

printf("Mouse or Mouse Driver Absent,P1ease install!");

delay(5000);

exit(1);

}

x=10;y=10;

cursor(x,y); /\* 置十字光标在屏幕中心。 \*/

do {

Print\_Time();

draw\_face(1);

newxy(&x,&y,&buttons);

if(checkWin()) gameRes = 1;

} while(!gameRes);

if (gameRes==1)

Update\_Record();

} while (!confirm(gameRes));

end\_music();

delay(500);

system("CLS");

return 0;

}

**2、menu\_mouse.c**

#include "miner.h"

char Trans\_Num[10];//负责将数字转化成字符串的临时变量

clock\_t BeginTime;

int TimeCount,LastTime;

int record[Record\_Size];

void Paint\_Menu\_Group(int num)//绘制顶部的菜单栏主体

{

Menu=Menus[num];

setfillstyle(SOLID\_FILL,DARKGRAY);

//setcolor(BLACK);

bar(Menu.startx,Menu.starty,Menu.startx+Menu\_Length,Menu.starty+Menu\_Height);

setcolor(WHITE);

rectangle(Menu.startx,Menu.starty,Menu.startx+Menu\_Length,Menu.starty+Menu\_Height);

outtextxy(Menu.startx+5,Menu.starty+5,Menu.name);

}

void Init\_Record()

{

int i;

for (i=0;i<Record\_Size;i++)

record[i]=INF;

}

void Update\_Record()

{

int i,j,T=(clock()-BeginTime)/CPS;

for (i=0;i<Record\_Size;i++)

if (T<record[i])

{

for (j=Record\_Size-1;j>i;j--)

record[j]=record[j-1];

record[i]=T;

break;

}

for (i=0;i<Record\_Size;i++)

{

if (record[i]==INF)

{

strcpy(Windows[0].word[i]+5,"N/A");

}

else

{

itoa(record[i],Trans\_Num,10);

strcpy(Windows[0].word[i]+5,Trans\_Num);

}

}

}

void Open\_Menu\_Group(int num)//打开菜单栏

{

int i;

if (num<0 || num>=Menu\_Cnt) return;

Menus[num].open=1;

Menu=Menus[num];

for (i=0;i<Menu.size;i++)

{

//outtextxy(STARTX,STARTY+50\*i,"You win the game , play again (Y/N)?");

//setcolor(BLACK);

setfillstyle(SOLID\_FILL,DARKGRAY);

bar(Menu.startx,Menu.starty+(i+1)\*Menu\_Height,Menu.startx+Menu\_Length,Menu.starty+(i+2)\*Menu\_Height);

setcolor(WHITE);

rectangle(Menu.startx,Menu.starty+(i+1)\*Menu\_Height,Menu.startx+Menu\_Length,Menu.starty+(i+2)\*Menu\_Height);

outtextxy(Menu.startx+4,Menu.starty+(i+1)\*Menu\_Height+4,Menu.option[i]);

}

}

void Close\_Menu\_Group(int num)//关闭菜单栏

{

int i;

if (num<0 || num>=Menu\_Cnt) return;

Menus[num].open=0;

//putimage(0,0,BackGround,COPY\_PUT);

//system("CLS");

setfillstyle(BLACK,DARKGRAY);

bar(Menus[0].startx,Menus[0].starty+Menu\_Height+1,STARTX+SIZEX\*COL,STARTY-1);

drawTable();

for (i=0;i<Menu\_Cnt;i++)

Paint\_Menu\_Group(i);

}

void Switch\_Menu\_Group(int num)//切换某菜单栏的开闭

{

if (Menus[num].open)

Close\_Menu\_Group(num);

else

Open\_Menu\_Group(num);

}

void Init\_Windows()

{

int i;

memset(Windows,0,sizeof(Windows));

Init\_Record();

strcpy(Windows[0].Title,"Record");

strcpy(Windows[1].Title,"Help");

strcpy(Windows[2].Title,"About");

for (i=0;i<Window\_Cnt;i++)

{

Windows[i].startx=STARTX+SIZEX\*COL+20;

Windows[i].starty=STARTY;

Windows[i].endx=STARTX+SIZEX\*COL+240;

Windows[i].endy=STARTY+SIZEY\*ROW;

Windows[i].open=0;

}

/\*Init Record Begin\*/

strcpy(Windows[0].word[0],"1. N/A");

strcpy(Windows[0].word[1],"2. N/A");

strcpy(Windows[0].word[2],"3. N/A");

/\*Init Record End\*/

/\*Init Help Begin\*/

strcpy(Windows[1].word[0]," Number in blocks stands");

strcpy(Windows[1].word[1],"for the number of mines in");

strcpy(Windows[1].word[2],"the nearby 8 blocks.");

strcpy(Windows[1].word[3]," Using the left button to");

strcpy(Windows[1].word[4],"dig out a block.");

strcpy(Windows[1].word[5]," Using the right button to");

strcpy(Windows[1].word[6],"flag a block.");

strcpy(Windows[1].word[7]," Have FUN!");

/\*Init Help End\*/

/\*Init About Begin\*/

strcpy(Windows[2].word[0],"Author:");

strcpy(Windows[2].word[1],"Chen Shihan");

strcpy(Windows[2].word[2],"Zhang Zhuohao");

strcpy(Windows[2].word[3],"Yi Mingqi");

/\*Init About End\*/

}

void Close\_Window(int num)

{

if (num<0 || num>=Window\_Cnt) return;

Windows[num].open=0;

tmp\_window=Windows[num];

setfillstyle(SOLID\_FILL,BLACK);

bar(tmp\_window.startx,tmp\_window.starty,tmp\_window.endx,tmp\_window.endy);

}

void Close\_Other\_Windows(int num)

{

int i;

for (i=0;i<Window\_Cnt;i++)

if (i!=num)

Close\_Window(i);

}

void Open\_Window(int num)

{

int i;

if (num<0 || num>=Window\_Cnt) return;

Close\_Other\_Windows(num);

Windows[num].open=1;

tmp\_window=Windows[num];

setcolor(WHITE);

rectangle(tmp\_window.startx,tmp\_window.starty,tmp\_window.endx,tmp\_window.endy);

setfillstyle(SOLID\_FILL,BLUE);

setcolor(WHITE);

bar(tmp\_window.startx+1,tmp\_window.starty+1,tmp\_window.endx-1,tmp\_window.starty+Title\_Height);

outtextxy(tmp\_window.startx+5,tmp\_window.starty+5,tmp\_window.Title);

for (i=0;i<Word\_Row;i++)

outtextxy(tmp\_window.startx+5,tmp\_window.starty+Title\_Height+5+i\*Letter\_Height,tmp\_window.word[i]);

}

void Switch\_Window(int num)

{

if (Windows[num].open)

Close\_Window(num);

else

Open\_Window(num);

}

int Click\_Menu(int men,int opt)//点击菜单中的某个选项

{

switch(men)

{

case 0:

switch(opt)

{

case 0: //NewGame

newGame();

flag\_cnt=0;

draw\_loading();

BeginTime=clock();

LastTime=-1;

break;

case 1: //BestRecords

Switch\_Window(0);

break;

case 2: //Exit

return 1;

break;

}

break;

case 1:

switch(opt)

{

case 0: //Help

Switch\_Window(1);

break;

case 1: //About

Switch\_Window(2);

break;

}

break;

}

return 0;

}

void Close\_Other\_Menus(int num)//关闭除了num之外的所有菜单组

{

int i;

for (i=0;i<Menu\_Cnt;i++)

if (i!=num)

Close\_Menu\_Group(i);

if (Menus[num].open)

Open\_Menu\_Group(num);

}

void Init\_Menus()

{

int i;

strcpy(Menus[0].name,"Game");

strcpy(Menus[1].name,"Help");

for (i=0;i<Menu\_Cnt;i++)

{

Menus[i].startx=STARTX+i\*Menu\_Length;

Menus[i].starty=STARTY-2\*Menu\_Height;

Menus[i].open=0;

Paint\_Menu\_Group(i);

}

Menus[0].size=3;

strcpy(Menus[0].option[0],"NewGame");

strcpy(Menus[0].option[1],"Record");

strcpy(Menus[0].option[2],"Exit");

Menus[1].size=2;

strcpy(Menus[1].option[0],"Help");

strcpy(Menus[1].option[1],"About");

//Open\_Menu\_Group(0);

}

int In\_Range(int x,int y,int x0,int y0,int x1,int y1)

{

if (x>x0 && x<x1 && y>y0 && y<y1)

return 1;

else

return 0;

}

int In\_Menu\_Range(int x,int y,int men,int opt)

{

return In\_Range(x,y,Menus[men].startx,Menus[men].starty+(opt+1)\*Menu\_Height,Menus[men].startx+Menu\_Length,Menus[men].starty+(opt+2)\*Menu\_Height);

}

void Print\_Time()

{

TimeCount=(clock()-BeginTime)/CPS;

if (TimeCount==LastTime) return;

itoa(TimeCount,Trans\_Num,10);

setfillstyle(SOLID\_FILL,BLACK);

bar(25,25,65,65);

setcolor(WHITE);

outtextxy(40,40,Trans\_Num);

LastTime=TimeCount;

}

/\*-----------------------------------------菜单部分函数结束------------------------------------------\*/

/\*画光标函数cursor()\*/

/\*该函数将用画线函数line()画出一个十字形光标\*/

void cursor(int a,int b)

{

unsigned long gsize;

rectangle(a-MSIZE,b-MSIZE,a+MSIZE,b+MSIZE);

line(a-MSIZE,b-MSIZE,a+MSIZE,b+MSIZE);

line(a-MSIZE,b+MSIZE,a+MSIZE,b-MSIZE);

/\* 保存鼠标图形位图 \*/

gsize=imagesize(a-MSIZE,b-MSIZE,a+MSIZE,b+MSIZE);

imagep=malloc(gsize);

getimage(a-MSIZE,b-MSIZE,a+MSIZE,b+MSIZE,imagep);

}

/\*鼠标器初始化函数init ()\*/

/\*该函数将通过调用int 33H的0号功能调用对鼠标器进行初始化，调用7号和8号功能，设置x和y位置的最小和最大值。这就为鼠标器移动进行了初始化准备。由于0号功能调用是测试鼠标驱动程序是否安装，因此在运行该程序前必须首先执行鼠标驱动程序mouse.com，若调用该函数执行了0号功能调用，当返回值为0时(即返回参数为0)，表示未安装成功，这可能是鼠标器或驱动程序末安装。这时程序将显示Mouse or Mouse Driver Absent，并回到系统。\*/

int init(int xmi,int xma,int ymi,int yma)

{

int retcode;

regs.w.ax=0;

int86(51,&regs,&regs);

retcode=regs.w.ax;

if(retcode==0)

return 0; /\* 返回0表示鼠标或鼠标驱动程序未安装 \*/

regs.w.ax=7;

regs.w.cx=xmi;

regs.w.dx=xma;

int86(51,&regs,&regs);

regs.w.ax=8;

regs.w.cx=ymi;

regs.w.dx=yma;

int86(51,&regs,&regs); /\* 表示鼠标器和驱动程序已安装 \*/

return retcode;

}

/\*读鼠标的位置和按钮状态函数read()\*/

/\*该函数将通过调用int 33H的3号功能调用，读鼠标的位置和按钮状态。鼠标的x、y位置值将由指针mx和my给出，而按钮状态则由mbutt指针给出。\*/

int read(int \*mx,int \*my,int \*mbutt)

{

int xx0=\*mx,yy0=\*my,but0=0,mb;

int xnew, ynew;

do {

regs.w.ax=3;

int86(51,&regs,&regs);

xnew=regs.w.cx;

ynew=regs.w.dx;

\*mbutt=regs.w.bx;

} while(xnew==xx0 && ynew==yy0 && \*mbutt == but0 );

\*mx=xnew;

\*my=ynew;

mb=(\*mbutt);

if(mb){

if(mb==1) return 1; /\*左键按下\*/

if (mb==2) return 2; /\*右键按下\*/

return 3; /\*其它的按键情况\*/

}

else

return 0;

}

/\*该函数将通过调用read()函数来判断是否有按钮按下，若按下，则调用cursor()函数在新位置画出一十字光标\*/

void newxy(int \*mx,int \*my,int \*mbutt)

{

int ch,xx0=\*mx,yy0=\*my,x,y,i,j,Clicked=0;

int xm,ym;

ch=read(&xm,&ym,mbutt);

switch (ch) {

case 0:

putimage(xx0-MSIZE,yy0-MSIZE,imagep,XOR\_PUT);

putimage(xm-MSIZE,ym-MSIZE,imagep,XOR\_PUT);

break;

case 1:

if (LastButton==1) break;//防止按住鼠标产生问题

for (i=0;i<Menu\_Cnt;i++)

{

if (In\_Menu\_Range(xx0,yy0,i,-1))//点击了菜单组

{

Switch\_Menu\_Group(i);

Close\_Other\_Menus(i);

Clicked=1;

break;

}

if (Menus[i].open)

{

for (j=0;j<Menus[i].size;j++)

if (In\_Menu\_Range(xx0,yy0,i,j))

{

if (Click\_Menu(i,j)) gameRes=2;

Close\_Other\_Menus(-1);

Clicked=1;

break;

}

}

}

if (!Clicked) //没有点击按钮，关闭所有菜单栏

Close\_Other\_Menus(-1);

if (In\_Range(xx0,yy0,STARTX,STARTY,STARTX+SIZEX\*COL,STARTY+SIZEY\*ROW) && !Clicked)

{

gameRes = openMine((yy0-STARTY)/SIZEY,(xx0-STARTX)/SIZEX);

putimage(xx0-MSIZE,yy0-MSIZE,imagep,XOR\_PUT);

}

break;

case 2:

if (LastButton==2) break;

if (In\_Range(xx0,yy0,STARTX,STARTY,STARTX+SIZEX\*COL,STARTY+SIZEY\*ROW))

{

flag\_question\_Block((yy0-STARTY)/SIZEY,(xx0-STARTX)/SIZEX);

putimage(xx0-MSIZE,yy0-MSIZE,imagep,XOR\_PUT);

}

break;

case 3:

cursor(xx0,yy0);

cursor(xm,ym);

putpixel(xm,ym,7);

break;

}

LastButton=ch;

\*mx=xm;

\*my=ym;

}

void Init\_Game()

{

newGame();

flag\_cnt=0;

draw\_loading();

Init\_Menus();

gameRes=0;

LastButton=-1;

BeginTime=clock();

LastTime=-1;

}

**3、miner.c**

#include "miner.h"

int table[ROW][COL]; /\* 雷区各方块是否有雷\*/

int num[ROW][COL]; /\*雷区各方块周围的雷数\*/

int flag[ROW][COL]; /\*雷区各方块的状态\*/

int pi,pj; /\*光标当前位置\*/

int di[8]={-1,-1,0,1,1,1,0,-1}; /\*方块八向偏移量\*/

int dj[8]={0,1,1,1,0,-1,-1,-1};

int flag\_cnt,totalMine;/\*\*/

void \*imagep;

int gameRes; /\*记录游戏结束的结果状态：0表示退出游戏；-1游戏失败； 1为胜利。\*/

union REGS regs;

int xmin,xmax,ymin,ymax,x\_max,y\_max,LastButton;

/\*初始化图形显示方式\*/

void initGraph(){

/\* 要求自动检测 \*/

int gdriver = DETECT, gmode, errorcode;

/\* 初始化图形模式 \*/

initgraph(&gdriver, &gmode, "");

/\* 读取初始化结果 \*/

errorcode = graphresult();

if (errorcode != grOk) /\* 发生错误 \*/

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1); /\* 具有错误代码的返回 \*/

}

}

/\*随机生成地雷的分布\*/

void generateMine(){

int i,k;

int ri,rj;

int ni,nj;

static int seed=0;

totalMine = ROW \* COL / 6;

memset(table, 0 ,sizeof(table));

memset(num, 0 ,sizeof(num));

srand(seed++);

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

do{

ri = rand()%ROW;

rj = rand()%COL;

}while(table[ri][rj]);

for(k=0;k<8;k++){

ni = ri + di[k];

nj = rj + dj[k];

if(ni>=0 && ni< ROW && nj>=0 && nj<COL) num[ni][nj]++;

}

table[ri][rj] = 1;

}

memset(flag,UNFLAG,sizeof(flag));

}

/\*绘制小方块\*/

void drawBlock(int i, int j){

int x,y,xx,yy;

int color;

int mx,my;

x = STARTX+j\*SIZEX;

y = STARTY+i\*SIZEY;

xx = x+SIZEX-1;

yy = y+SIZEY-1;

color = (flag[i][j]==UNFLAG)?LIGHTGRAY:DARKGRAY;

setcolor(WHITE);

rectangle(x,y,xx,yy);

setfillstyle(SOLID\_FILL,color);

//floodfill(x+1,y+1,WHITE);

bar(x+1,y+1,xx-1,yy-1);

setcolor(WHITE); line(x,y,xx,y);line(x,y,x,yy);

setcolor(BLACK); line(xx,y,xx,yy);line(x,yy,xx,yy);

if(pi==i && pj==j){

setcolor(RED);

rectangle(x+1,y+1,xx-1,yy-1);

}

mx = x+5;

my = y+5;

setcolor(LIGHTGRAY);

switch(flag[i][j]){

case UNFLAG:

outtextxy(mx,my," ");

break;

case FLAGED:

setcolor(RED);

outtextxy(mx,my,"F");

break;

case QUESTION:

outtextxy(mx,my,"?");

break;

case EXPLOD:

outtextxy(mx,my,"\*");

break;

case OPEN:

if(num[i][j]>=1 && num[i][j]<=8){

char buf[3];

itoa(num[i][j],buf,10);

outtextxy(mx,my,buf);

}

break;

}

}

/\*绘制雷区\*/

void drawTable(){

int i,j;

for(i=0;i<ROW;i++)

for(j=0;j<COL;j++) drawBlock(i,j);

}

/\*初始化游戏\*/

void newGame(){

cleardevice();

generateMine();

pi = pj = 0;

drawTable();

}

/\*判断游戏是否胜利\*/

int checkWin(){

int i,j;

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

for(j=0;j<COL;j++){

if(table[i][j] == 0 && flag[i][j] != OPEN) {

return 0;

}

}

}

return 1;

}

/\*判断游戏是否重新开始\*/

int confirm(int res){

char buffer[100];

int kk;

if(res == 1) {

strcpy(buffer,"You win the game , play again (Y/N)?");

}

else if(res == -1){

lose\_music();

draw\_face(0);

strcpy(buffer,"You lose the game ,play again (Y/N)?");

}

else {

strcpy(buffer,"Exit game ,play again (Y/N)?");

}

setcolor(WHITE);

outtextxy(0,0,buffer);

kk = bioskey(0);

kk = (kk&0xff);

if(kk == 'Y' || kk == 'y') return 0;

return 1;

}

/\*标记有雷\*/

/\*void flagBlock(int i,int j){

if(flag[i][j] == FLAGED){

flag[i][j] = UNFLAG;

}

else if(flag[i][j] == UNFLAG){

flag[i][j] = FLAGED;

}

drawBlock(i,j);

}\*/

/\*标记不确定\*/

/\*void questBlock(int i,int j){

if(flag[i][j] == QUESTION){

flag[i][j] = UNFLAG;

}

else {

if(flag[i][j] == UNFLAG) flag[i][j] = QUESTION;

}

drawBlock(i,j);

}\*/

void draw\_loading()

{

setfillstyle(SOLID\_FILL,BLUE);

bar(STARTX,STARTY+SIZEY\*ROW+20,STARTX+SIZEX\*COL,STARTY+SIZEY\*ROW+40);

setfillstyle(SOLID\_FILL,GREEN);

bar(STARTX,STARTY+SIZEY\*ROW+20,STARTX+SIZEX\*COL\*flag\_cnt/totalMine,STARTY+SIZEY\*ROW+40);

}

void draw\_face(int res)

{

setcolor(YELLOW);

circle(275,41,10);

circle(296-25,37,2);

circle(304-25,37,2);

circle(296-25,37,1);

circle(304-25,37,1);

if (res==1) arc(300-25,45,180,360,4);

else if (res==0) arc(300-25,49,0,180,4);

}

void flag\_question\_Block(int i,int j){

if(flag[i][j] == FLAGED){

flag\_cnt--;

draw\_loading();

flag[i][j] = QUESTION;

}

else if(flag[i][j] == UNFLAG){

flag\_cnt++;

draw\_loading();

flag[i][j] = FLAGED;

}

else if(flag[i][j] == QUESTION){

flag[i][j] = UNFLAG;

}

drawBlock(i,j);

}

/\*挖雷\*/

int openMine(int i,int j){

int ii,jj,k;

if(flag[i][j] == OPEN) return 0;

if(table[i][j]){ /\*meet a mine\*/

for(ii=0;ii<ROW;ii++){

for(jj=0;jj<COL;jj++){

if(table[ii][jj] && flag[ii][jj] == UNFLAG){

flag[ii][jj] = EXPLOD;

drawBlock(ii,jj);

}

}

}

draw\_face(0);

return -1; /\*游戏失败\*/

}

else{

flag[i][j] = OPEN;

drawBlock(i,j);

if(num[i][j]==0){

for(k=0;k<8;k++){

int ni = i + di[k];

int nj = j + dj[k];

if(ni>=0 && ni< ROW && nj>=0 && nj<COL)

openMine(ni,nj);

}

}

return 0;

}

}

/\*自动挖开\*/

int autoOpen(int i,int j){

int k,c=0;

int ni,nj;

int ret=0;

if(! (flag[i][j] == OPEN)) return 0;

for(k=0;k<8;k++){

ni = i + di[k];

nj = j + dj[k];

if(ni>=0 && ni< ROW && nj>=0 && nj<COL)

if(flag[ni][nj] == FLAGED) c++;

}

if(c == num[i][j]){

for(k=0;k<8;k++){

int ni = i + di[k];

int nj = j + dj[k];

if(ni>=0 && ni< ROW && nj>=0 && nj<COL)

if(flag[ni][nj] == UNFLAG){

if(openMine(ni,nj) == -1) ret = -1;

}

}

}

return ret;

}

**4、music.c**

#include "miner.h"

void lose\_music()

{

WAVE \*pwave;

initsound(); // 初始化声卡

pwave = load\_wave("Bomb.wav"); // 载入wav文件, 返回WAVE指针

play\_wave(pwave, 1); // repeat 1 times 播放wav声音

getchar();

free\_wave(pwave); // 释放wav声音占用的资源

closesound(); // 关闭声卡

}

void end\_music()

{

MIDI \*pmidi;

initsound(); // 初始化声卡

pmidi = load\_midi("dream.mid"); // 载入midi文件, 返回MIDI指针

play\_midi(pmidi, -1); // repeat forever 播放midi音乐

getchar();

free\_midi(pmidi); // 释放midi音乐占用的资源

closesound(); // 关闭声卡

}

5、cartoon.c

#include "miner.h"

void cartoon()

{

int x=0, y;

clrscr();

for(y=0; y<100; y++)

{

draw\_bug(x, y);

delay(10);

if(kbhit()) // 检测敲键

goto done;

}

done: ;

}

void draw\_bug(int x, int y)

{

int i,j,k,c,len=20;

cleardevice();

setbkcolor(10);

setcolor(4);

setfillstyle(1, 4);

for(k=0;k<20;k++){

for (i = 0; i < 20; i++){

for (j = 0; j < 20; j++)

{

c=rand()%100;

setfillstyle(1, c);

/\*设定每个格子的宽与高\*/

bar( 50+j\*len,50+i\*len,50+len+j\*len,50+len+i\*len);

}

if(k==0)

delay(1);

}

delay(20);

}

}

**6、miner.h**

#include <graphics.h>

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include "time.h"

#include <dos.h>

#define Menu\_Height 20

#define Menu\_Length 80

#define Menu\_Name\_Length 10

#define Menu\_Max\_Size 5

#define Window\_Cnt 3

#define Title\_Height 20

#define MSIZE 5

#define Menu\_Cnt 2

#define MAXX 500

#define MAXY 500

#define Record\_Size 3

#define CPS CLOCKS\_PER\_SEC

#define INF 1000000000

#define Word\_Row 20

#define Word\_Length 45

#define Title\_Length 20

#define Letter\_Height 20

/\*以上为扫雷界面所需宏定义\*/

#define ROW 16

#define COL 16

#define STARTX 75

#define STARTY 75

#define SIZEX 20

#define SIZEY 20

#define UNFLAG 0 /\*未标记或挖开\*/

#define FLAGED 1 /\*已标记为雷\*/

#define QUESTION 20 /\*不能确定\*/

#define EXPLOD 30 /\*踩雷\*/

#define OPEN 40 /\*挖开\*/

/\*扫雷相关宏定义\*/

extern char Trans\_Num[10];//负责将数字转化成字符串的临时变量

extern int table[ROW][COL]; /\* 雷区各方块是否有雷\*/

extern int num[ROW][COL]; /\*雷区各方块周围的雷数\*/

extern int flag[ROW][COL]; /\*雷区各方块的状态\*/

extern int pi,pj; /\*光标当前位置\*/

extern int di[8]; /\*方块八向偏移量\*/

extern int dj[8];

extern int flag\_cnt,totalMine;/\*\*/

extern void \*imagep;

extern int gameRes; /\*记录游戏结束的结果状态：0表示退出游戏；-1游戏失败； 1为胜利。\*/

extern union REGS regs;

extern int xmin,xmax,ymin,ymax,x\_max,y\_max,LastButton;

typedef struct

{

char shape, color;

} NODE;

NODE bug[5];

struct Menu\_Group

{

int open,size,startx,starty;

char name[Menu\_Name\_Length];

char option[Menu\_Max\_Size][Menu\_Name\_Length];

}Menus[Menu\_Cnt],Menu;

struct Window

{

int startx,starty,endx,endy,open;

char word[Word\_Row][Word\_Length],Title[Title\_Length];//记录窗口的标题和内容

}Windows[Window\_Cnt],tmp\_window;

extern clock\_t BeginTime;

extern int TimeCount,LastTime;

extern int record[Record\_Size];

/\*以上菜单界面所需变量声明\*/

/\*初始化图形显示方式\*/

void initGraph();

/\*随机生成地雷的分布\*/

void generateMine();

/\*绘制小方块，输入第i行第j列，绘制当前坐标的方格\*/

void drawBlock(int i, int j);

/\*绘制雷区\*/

void drawTable();

/\*初始化游戏\*/

void newGame();

/\*判断游戏是否胜利\*/

int checkWin();

/\*判断游戏是否重新开始，输入res=1则游戏重新开始，反之亦然\*/

int confirm(int res);

/\*显示扫雷进度和笑哭脸的状态栏\*/

void draw\_loading();

void draw\_face();

/\*鼠标右键点击时标记雷，点一下标记F，点两下标记？\*/

void flag\_question\_Block(int i,int j);

/\*挖雷\*/

int openMine(int i,int j);

/\*自动挖开\*/

int autoOpen(int i,int j);

/\*绘制顶部的菜单栏主体 \*/

void Paint\_Menu\_Group(int num);

void Init\_Record();

void Update\_Record();

/\*打开菜单栏 \*/

void Open\_Menu\_Group(int num);

/\*关闭菜单栏 \*/

void Close\_Menu\_Group(int num);

/\*切换某菜单栏的开闭\*/

void Switch\_Menu\_Group(int num);

void Init\_Windows();

void Close\_Window(int num);

void Close\_Other\_Windows(int num);

void Open\_Window(int num);

void Switch\_Window(int num);

/\*点击菜单中的某个选项 \*/

int Click\_Menu(int men,int opt);

void Close\_Other\_Menus(int num);//关闭除了num之外的所有菜单组

void Init\_Menus();

int In\_Range(int x,int y,int x0,int y0,int x1,int y1);

int In\_Menu\_Range(int x,int y,int men,int opt);

void Print\_Time();

/\*画光标函数cursor()\*/

/\*该函数将用画线函数line()画出一个十字形光标\*/

void cursor(int a,int b);

/\*鼠标器初始化函数init ()\*/

/\*该函数将通过调用int 33H的0号功能调用对鼠标器进行初始化，调用7号和8号功能，设置x和y位置的最小和最大值。这就为鼠标器移动进行了初始化准备。由于0号功能调用是测试鼠标驱动程序是否安装，因此在运行该程序前必须首先执行鼠标驱动程序mouse.com，若调用该函数执行了0号功能调用，当返回值为0时(即返回参数为0)，表示未安装成功，这可能是鼠标器或驱动程序末安装。这时程序将显示Mouse or Mouse Driver Absent，并回到系统。\*/

int init(int xmi,int xma,int ymi,int yma);

/\*读鼠标的位置和按钮状态函数read()\*/

/\*该函数将通过调用int 33H的3号功能调用，读鼠标的位置和按钮状态。鼠标的x、y位置值将由指针mx和my给出，而按钮状态则由mbutt指针给出。\*/

int read(int \*mx,int \*my,int \*mbutt);

/\*该函数将通过调用read()函数来判断是否有按钮按下，若按下，则调用cursor()函数在新位置画出一十字光标\*/

void newxy(int \*mx,int \*my,int \*mbutt);

void Init\_Game();

void lose\_music();

/\*播放结束音乐\*/

void end\_music();

/\*播放开场动画\*/

void cartoon();

/\*绘制开场动画\*/

void draw\_bug(int x, int y);

讨论聊天记录

