大型程序设计综合实验总结报告

课程名称：程序设计综合实验

选题名称：炸弹人小游戏

成员姓名(学号)：

许端清老师班：覃明(3100101629)

吕红兵老师班：汪嘉恒(3100103065)　陈向南(3100102812)

报告提交日期：2011 年 6 月 13 日

一、 系统功能说明

本游戏的开发思路源于网络游戏炸弹超人，并据此改编成单机版，不过受到水平限制，最终实现了游戏的功能——人机对战和双人对战，设计上加了一些新颖的道具，增添了游戏的乐趣。“Enjoy the Game !”始终是创作者的期望，因此我们尽力做的最好。但是限于 TC 的功能限制，游戏界面完全不能与强大的 ASL 引擎制作的炸弹超人游戏相比。

游戏功能如下：

1．此游戏提供了人机对战和双人对战两个模式，游戏方式为利用方向键和放炸弹键进行控制，可以炸开路上的箱子以获得道具，增加自己的能力。目标是将另一个玩家（或电脑）杀死。因为游戏中有很多道具，这就既考验玩家的技术也考验玩家的 RP 了。当然输代表不了什么，因为游戏的目的是娱乐。游戏结束后再进行游戏。

2．限于时间和能力限制，本游戏只仅准备了 3 张地图。

3．每个箱子炸掉后可能会出现道具，完全随机出现，每一局都不同。出现的道具只要移动到该格就吃掉了。道具不能被炸掉。炸弹爆炸时产生的威力柱可以穿过道具。

4，玩家放置炸弹后会计时，到一定时刻爆炸，在爆炸之前炸弹会有一定的动画效果。爆炸后会引爆周围四个方向上爆炸范围内的炸弹。

5，玩家死亡会间隔一定时间再复活。此期间不能使用换位道具。

6，每次游戏每个玩家刚开使有 3 次生命，当然玩家在游戏时可以通过捡加生命道具来增加自己的生命值，当然，生命上线是 5，生命耗尽后就结束了。当然不服输的可以结束后选择 Restart 继续游戏。游戏过程中始终会显示玩家的生命值。

7，游戏菜单和游戏过程中都设有背景音乐。由扬声器发出，使游戏不再平淡乏味。

8．提供了简单易懂的游戏帮助，能够让玩家快速的了解游戏。

9．游戏的关于界面中写有作者的联系方式，欢迎提出宝贵的意见，并长期提供技术支持。

二、 使用说明

1. 安装手册

1．将 炸弹人v2.0.rar 解压至任意目录下。

2．在解压后的目录下有两个文件夹，其中Game文件夹下包含已生成的应用程序，可以直接运行，Source 文件夹下包含文件的源程序，可以用TC2.0 编译。（建议使用winTC，因为本程序是在winTC环境下测试通过的）

3．如果你的电脑没有安装TC2.0请直接运行其中的Game目录下的bomber.exe。

4． 如果你的电脑安装了TC2.0，请打开Source文件夹，用右键单击MAIN.C，选择打开方式→选择程序→浏览→切换到你的 TC2.0 目录下，双击 TC.EXE，点确定即可打开源代码。

5．按 ALT+O选择Options 里的Directories，将Output directory 一栏清空。

6．按ALT+O选择Options里的Compiler将Mode一项改成 Huge(或者 Compact以上的任何一 项)。

7．按 F9 编译并连接程序。

8．这时在你在第一步解压的文件夹下的 Source 目录下已经生成了 MAIN.EXE，你

可以直接运行 这个程序，也可以在 TC 下按 CTRL+F9 来运行。

9. Game和Source文件夹下的其他文件说明：

EGAVGA.BGI：显卡驱动程序

*.CHR ：程序用到的字体文件

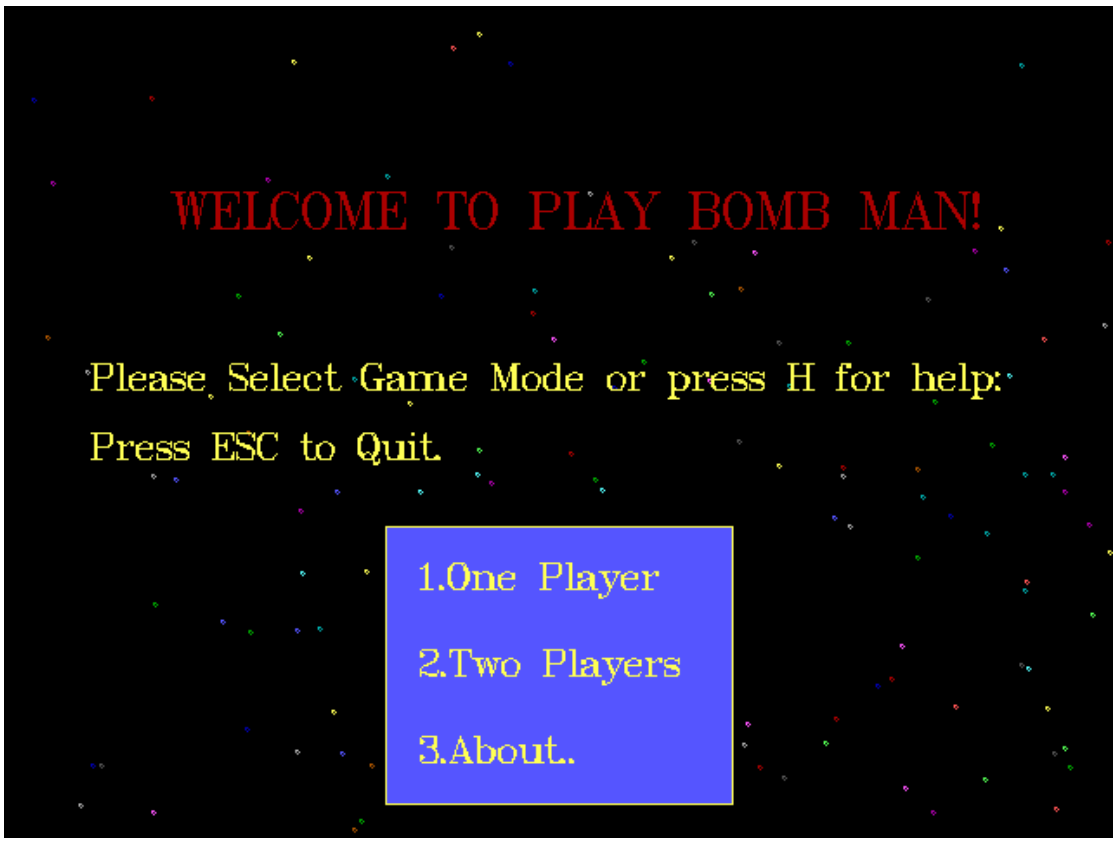
HELP.bmp ：游戏需要的16色位图文件（用于显示帮助信息）

About.bmp ：游戏需要的16色位图文件（用于显示关于信息）

2. 使用手册

顾名思义，就是放炸弹进行游戏，只要以前玩过其他版本的弹超人或是玩过网游泡泡堂的玩家都能充分体验到游戏的乐趣，而没玩过的玩家，也能很快熟悉游戏，发现游戏的乐趣。运行本游戏后，可以听到背景音乐并看到如下画面：

屏幕背景是星空，黑暗的夜空闪烁着五彩的星星。



屏幕上方显示的是游戏名和欢迎的字样Welcome to Play Bomb Man!，下方是游戏菜单提示说按下H可以进入帮助画面，按ESC键可以退出。然后接着是两个供玩家选择的游戏模式——单人游戏(即人机对战)和双人对战。3是关于信息，是游戏的开发信息以及技术支持，联系方式等

通过按1、2来选游戏模式，按ESC键退出，按H进入帮助界面。按3可以进入关于界面，里面是制作人的信息。由于中文实现不太简单，所以游戏使用英文界面，但是是非常容易懂的 ，不影响游戏的进行。

按H进入帮助后界面如下图所示：

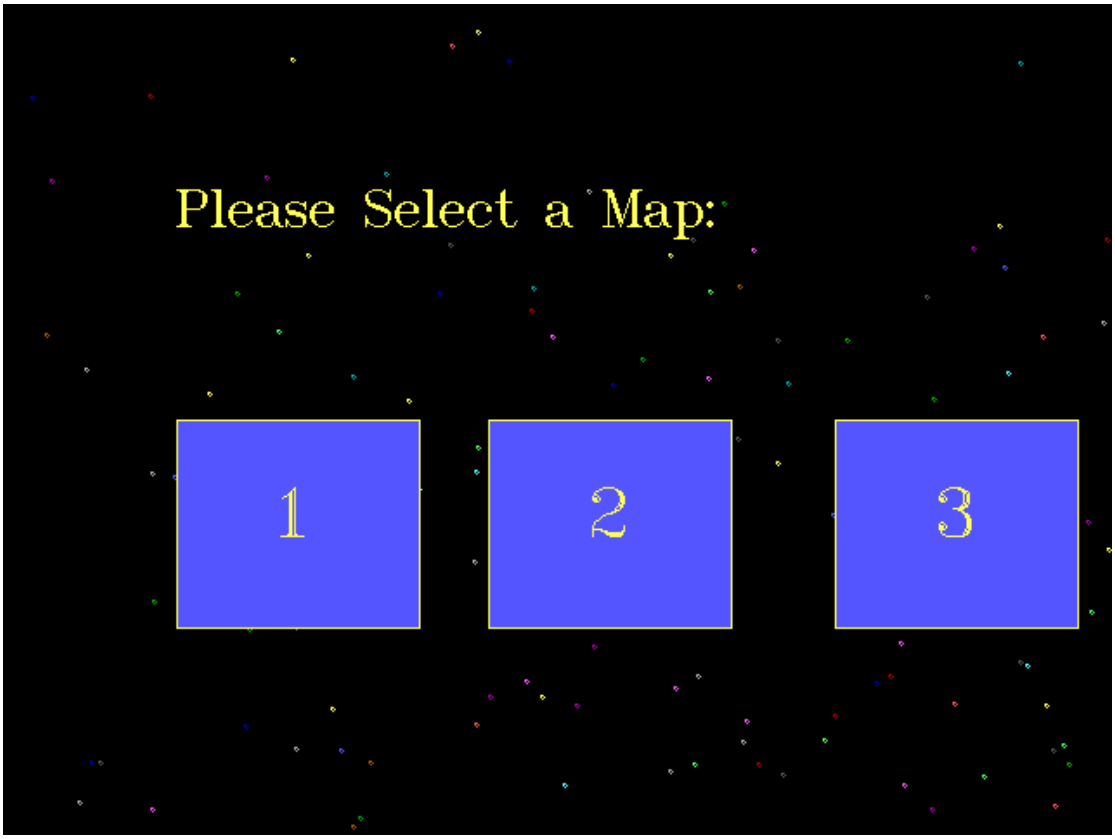
可以看到针对游戏的道具提供了详尽的帮助信息。其他方面信息也不是很重要，有些会在游戏中说明。在此界面按下回车键即可返回主菜单。

返回主菜单后按1或2选择一种模式，如果按1，这时候会出现如下界面：

因为1是单人模式，会要求玩家选择游戏难度，其中Easy最简单，Crazy最难，玩家爱可已根据自己水平选择最适合的难度，此时任选一种难度，会进入如下界面，要求

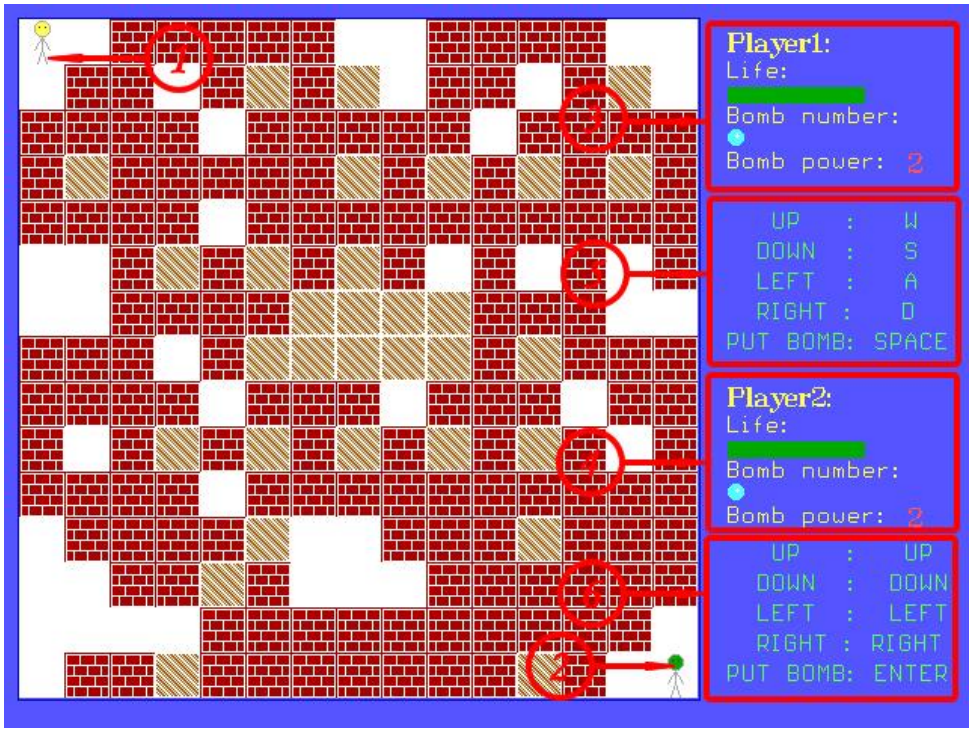
玩家选择地图：

本游戏有3张地图供玩家选择，此时可以选择1、2或3进入游戏，进入游戏后的界面如下图所示，如果一开始选择的是2(双人对战)那么就会直接出现以上的地图选择菜



单，任选地图即可进入游戏，进入游戏后，会出x现如下界面：

说明：①、②玩家1和玩家2



③、④玩家1和玩家2的信息，Life为生命，Bomb number为为炸弹数，Bomb

Power为炸弹的威力

⑤、⑥玩家1和玩家2的操作说明

在游戏过程中的任何时候你都可以按 ESC 键退出游戏。（因为暂停会影响背景音乐和游戏中炸弹爆炸等的计时，所以游戏没有设置暂停,同样没有设置按ESC键确认退出的菜单，因为一旦误按ESC想返回游戏的话，以上说的都会出错）。经过激烈的对抗，游戏会以某一方的失败而告终，这时弹出的界面为：

在此，按下ESC结束游戏，按下ENTER重新开始游戏，一切恢复开始状态。

三、 程序结构

1. 程序结构说明：

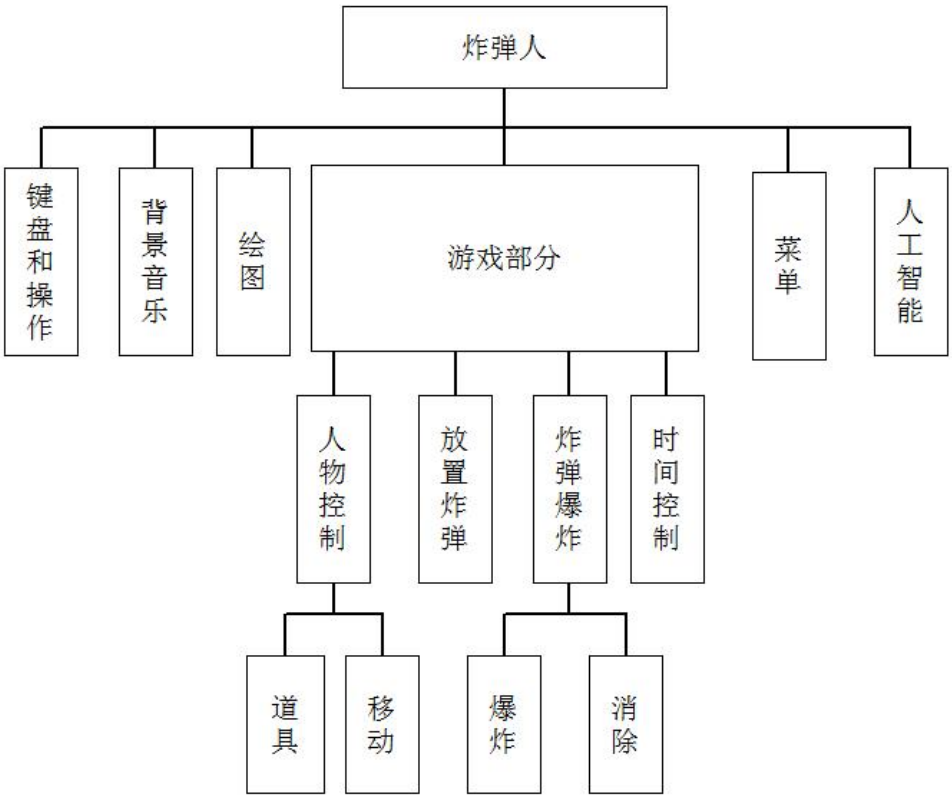
程序的结构还是设计得比较简单易懂的。为了充分发挥 C 语言的结构化，将程序分为几个模块，分写在几个.c 文件中，最终由 main.c 统领全部，作为程序的主程序。

所有文件功能说明：

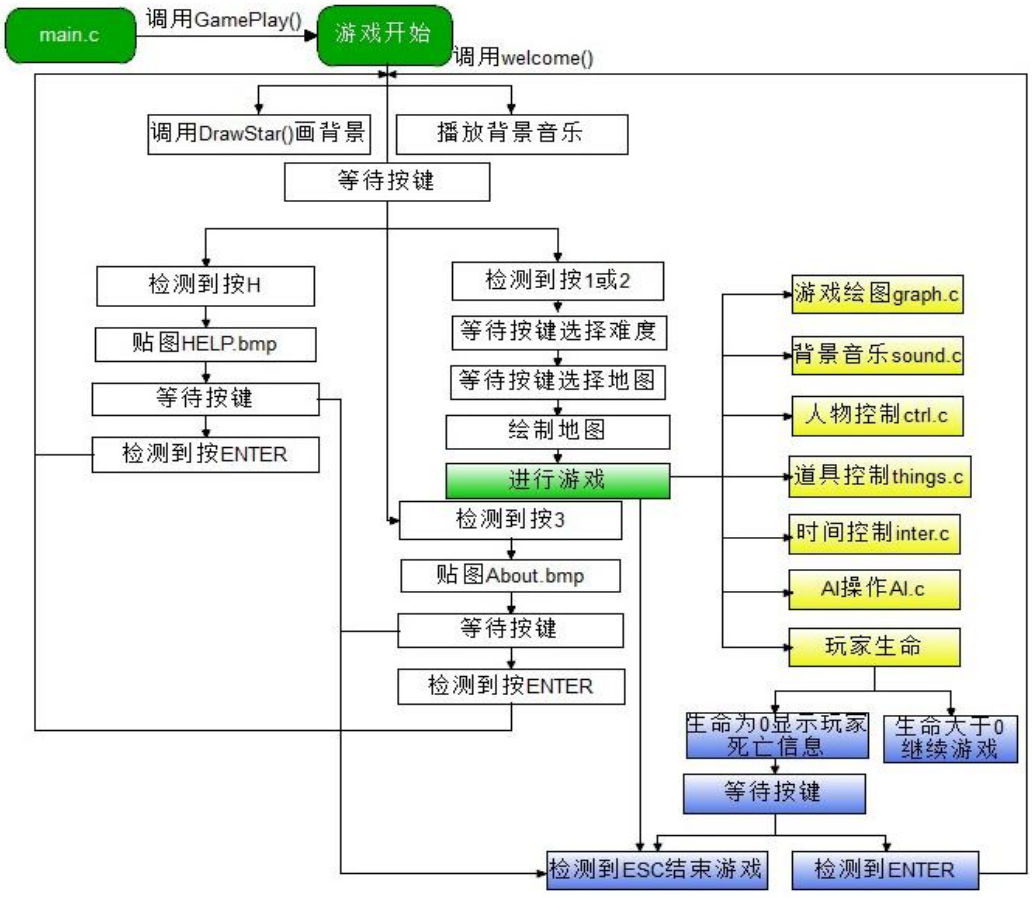
```
┌Source\
  ├MAIN.C----------主程序，总领全局，游戏的入口
  ├key.c----------读取按键信息并处理，转化为游戏操作
  ├INTER.C----------键盘和时间中断
  ├graph.C----------游戏的绘图
  ├DEFINE.C----------存放了游戏所有变量函数以及宏定义
  ├ctrl.C----------人物控制
  ├bomb.c----------与炸弹相关的所有函数
  ├AI.c----------人工智能引擎
```

├menu.c----------构筑菜单所有函数
└things.c----------实现游戏道具功能

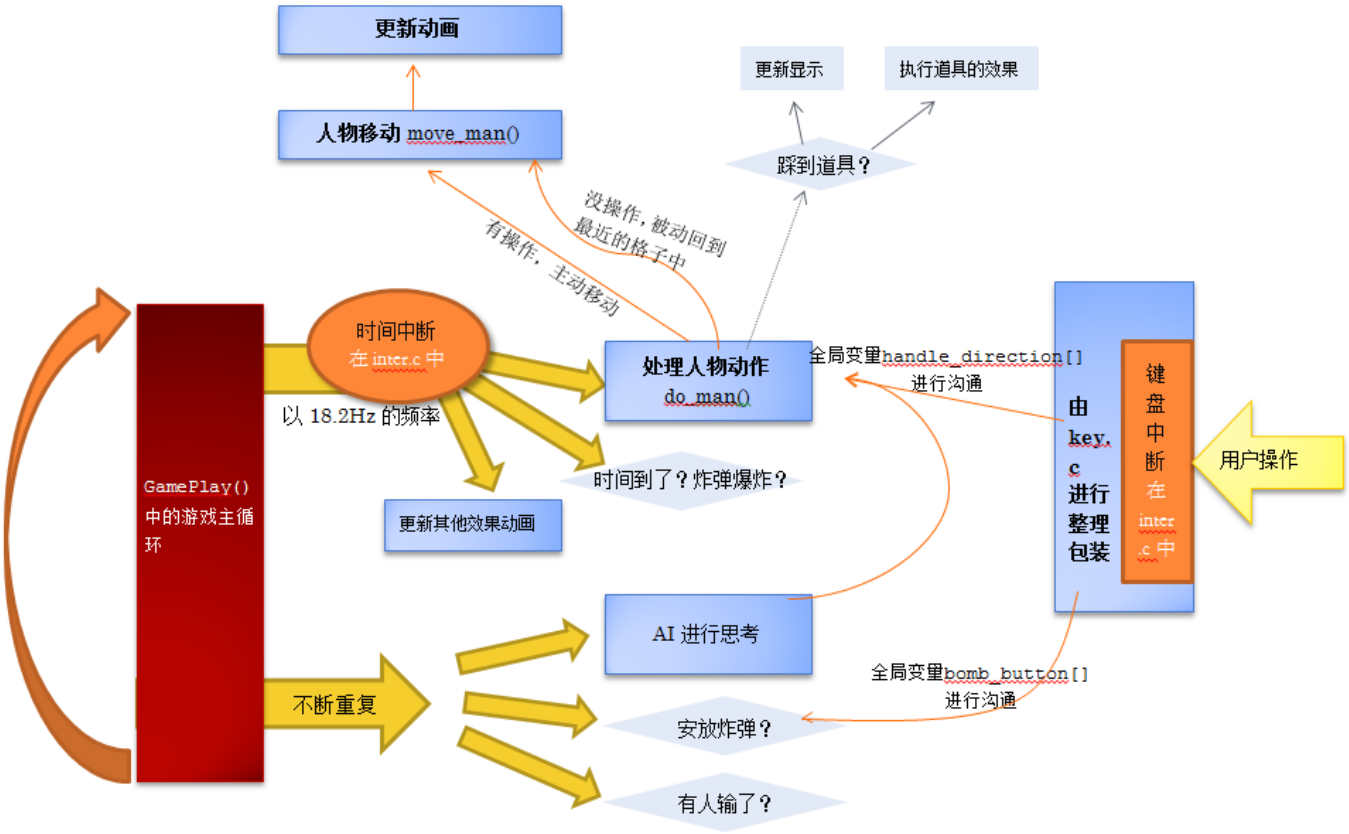
程序层次图：



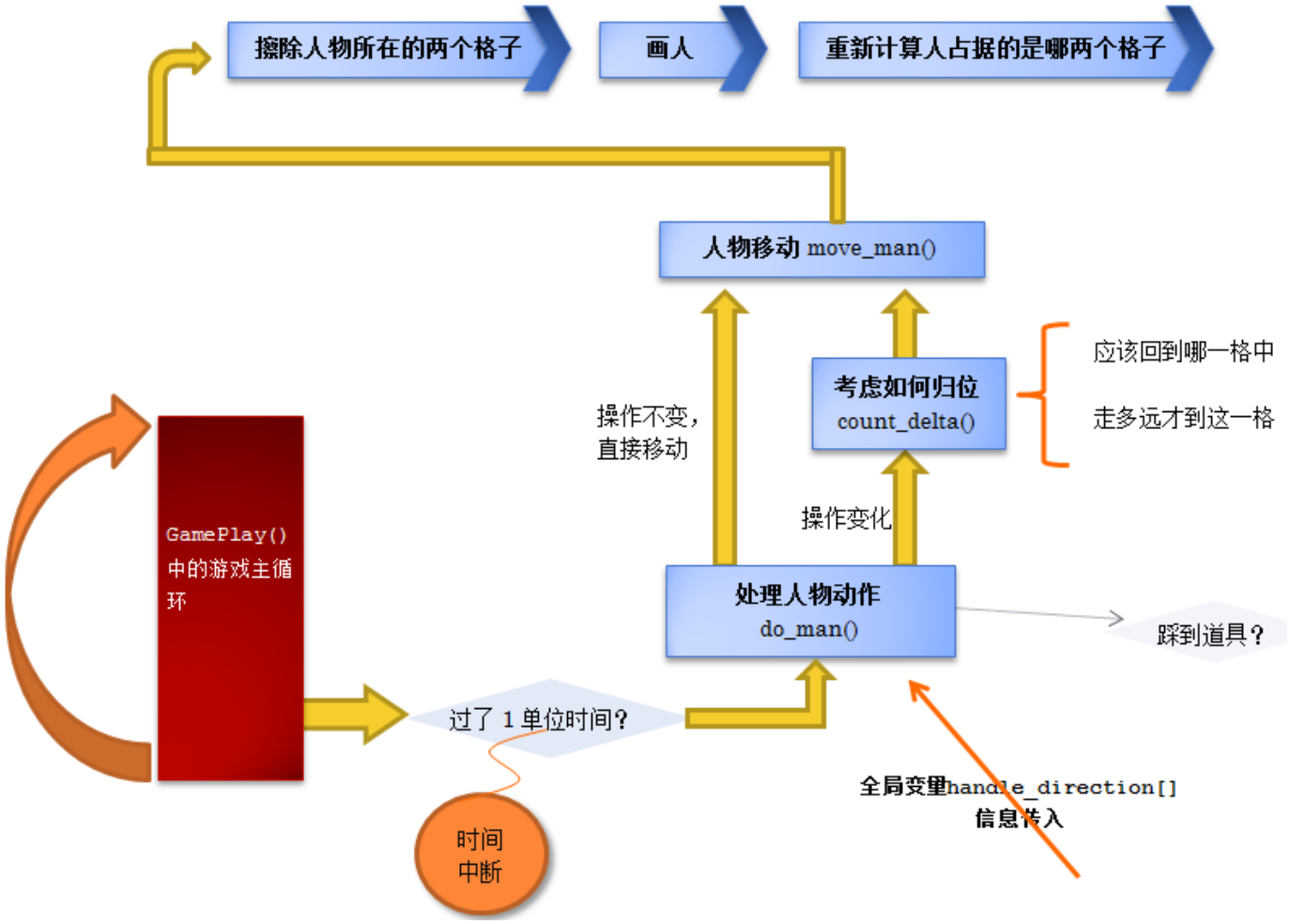
程序流程图：



游戏部分 工作原理示意图



人物控制 工作原理示意图



2. 程序重要数据变量说明

a) `int newtime;`全局时间，由时间中断来更新。每过 1/18.2 秒，newtime 就加 1

b) `int thetime;`用于与 newtime 进行比较，如果二者不等，则说明过了一个单位时间，从而执行一系列动作。

c) `int px[2] , pi[2][2];` `int py[2] , pj[2][2];`
人物的当前位置。注意：x 和 y 表示在屏幕上的坐标，i 和 j 表示在地图上的格点坐标

d) `char **a;`
经过一些转换后，a 成为一个二维数组，即相当于定义 `char a[MAP_SIZE][MAP_SIZE];`
它存放的是地图上的信息。例如，0 表示道路，1 表示砖墙，3 表示岩石（坚不可摧的墙）7 表示玩家零，8 表示玩家一，18 表示爆炸图案，31 到 44 表示场地上的道具

e) `int handle_direction[2];`玩家的操作方向，即想要人物移动的方向，其值为0不动，1上，2下，3左，4右。此全局变量是操作模块与人物控制模块沟通的桥梁。

f) `int direction[2];`人物当前的速度方向。如果它与上面的 handle_direction相等，则继续往该方向前进；如果不相等，则进行归位动画，让人物回到离他最近的一个中。

注：其余变量在 define.c 中都有详细说明。

3. 程序函数清单（详细的函数描述，参见源代码中的注释）

```
/*****      AI 模块   在 AI.C 中    *****/
/*
函数原型：int danger(int cx,int cy,int t)
函数功能：判断编号为 t 的炸弹对(cx,cy)点是否有威胁
参数说明：
入口：AI 的格点坐标(cx,cy)，和炸弹编号 t
出口：返回 0 和 1，0 表示无威胁，1 表示有威胁
算法描述：无特殊算法
*/
/*
函数原型：int potential(int cx,int cy,int j,int t);
函数功能：判断与玩家(cx,cy)点同行或同列的炸弹是否是潜在威胁
参数说明：
入口：AI 的格点坐标(cx,cy)，炸弹的列号或行号 j，同列 t 取 0，同行 t 取 1
出口：返回 0 和 1，0 表示这颗炸弹不是潜在威胁，1 表示  是潜在威胁
算法描述：无特殊算法
*/

/*
函数原型：int AI_danger(int cx,int cy);
函数功能：判断与玩家(cx,cy)点是否有危险参数说明：
参数说明：
入口：AI 的格点坐标(cx,cy)
出口：返回 0 和 1，0 表示 AI 在该点无危险，1 表示有危险
算法描述：无特殊算法
*/
```

```
/*
函数原型：void AI_OP(int b);
函数功能：AI 操作的接口，用以统领 AI 的全部功能
参数说明：
入口：b 为 AI 现在可以放的炸弹数
出口：无
算法描述：人工智能策略，程序注释中有详细说明
*/

/*
函数原型：int AI_Search(int t);
函数功能：搜索一条到达指定目标的路径，返回值为 0 表示周围没有要找的东西，1 为找到
          了，并且会返回一条到达该点的路径信息在 pt[]数组中
参数说明：
入口：t=0 为炸弹，1 为对手，2 为道具，3 为砖块，4 为躲避点,5 为判断该点放了炸弹后是
          否可以躲避
出口：返回 0 和 1，0 表示没有这样的路径，1 表示存在这样的路径
算法描述：A*寻路算法，答辩 PPT 中会详细介绍
*/

/*
函数原型：void AI_Move(int t);
函数功能：制 AI 的移动
参数说明：
入口：移动的策略 t,1 为躲避炸弹策略，2 为攻击对手，3 为寻找道具，4 为寻找可以炸掉的
          墙
出口：无
算法描述：对 AI 不同移动策略作不同的操作，没有特别的算法
*/

/*
函数原型：void AI_Move(int t);
函数功能：计算 AI 当前所在的格子坐标
*/

/*
函数原型：int cov(int x,int y);
函数功能：判断(x,y)点是否可以到达
参数说明：
入口：坐标(x,y)
出口：返回 0 和 1，0 表示无法到达，1 表示可以到达
算法描述：无特殊算法
*/
```

```
/*
函数原型：void AI_Reset(void);
函数功能：重置 AI 的路径信息
*/

/*      菜单模块  在 menu.c 中*/
/*
函数原型：void drawStar(void);
函数功能：画背景的漫天星星效果
*/

/*
函数原型：void welcome(void);
函数功能：游戏的主菜单，也即欢迎界面
*/
/*
函数原型：void selectmap(void);
函数功能：画选择地图界面的函数
*/

/*
函数原型：void putbmp(char *bmpname);
功能说明：贴图
参数说明
入口：需要贴图文件路径指针
出口：无
算法描述：详见报告下面的 16 位 BMP 贴图部分
*/
函数原型：void selectmap(void);
功函数能：画选择地图界面的函数
/*
函数原型：void selectlevel(void);
函数功能：画选择游戏难度界面的函数
*/

/*
函数原型：int musiC(void);
函数功能：在菜单中播放背景音乐，并接受菜单的按键操作
*/

/*      炸弹模块    在 bomb.c 中*/
/*
函数原型：int set_bomb(int who);
函数功能：安放炸弹 int serial_blast(int j,int i);
```

```
*/

/*
函数原型：int serial_blast(int j,int i);
函数功能：连续引爆其他炸弹。
*/

/*
函数原型：int blast(int p);
函数功能：引爆一个炸弹，绘制爆炸图样
*/
/*
函数原型：int remove_blast(int p);
函数功能：消除爆炸图样
*/

/*      人物控制模块     在 ctrl.c 中*/
/*
函数原型：int check_if_someone_lost();
函数功能：检查双方生命值
参数说明：
入口：无
出口：全局变量 loser
算法描述：无特殊算法
*/

/*
函数原型：int count_ij(int who , int y , int x , int j[] , int i[]);
函数功能：根据当前人物在屏幕上的坐标，计算出他所在地图上的两个格点坐标，
          顺便在地图数组 a 中把这两格标记为相应数字：player0 记为 7，player1 记为 8
参数说明：
入口：人物编号 who，在屏幕上的纵坐标 y 和横坐标 x
出口：数组  int j[] , int i[]
算法描述：无特殊算法
*/

/*
函数原型：int check_next(int who, int y, int x);
函数功能：检查人物移动方向上，下一个是否有障碍物
参数说明：
入口：人物序号 who，屏幕坐标（y，x）
出口：返回 0 表示无障碍，返回 1 表示有障碍
算法描述：无特殊算法
*/
```

```
/*
函数原型：int move_man(int who,int dy,int dx);
函数功能：将人物移动一段距离，实现一帧动画
参数说明：
入口：人物序号 who，在屏幕上的坐标偏移量（△y，△x）
出口：调用 draw_player()等绘图函数
算法描述：无特殊算法
*/


/*
函数原型：int speed(int time);
函数功能：计算人物能够前进的一段小长度，也就是速度
参数说明：
入口：人物已经行走的时间
出口：返回速度
算法描述：无特殊算法
*/

/*
函数原型：int count_delta(int who,int xy);
函数功能：计算人与最近的一格的偏移量，以便在释放操作时，自动回到最近一格中。
参数说明：
入口：人物序号 who，当前的一个坐标 y 或 x
出口：返回偏移量。如果偏移量为正，说明他需要向下（或右）走才能归位；负的，则向上
                （或左）
算法描述：无特殊算法
*/

/*
函数原型：int do_man(int who);
函数功能：控制人物行为
参数说明：
入口：人物序号 who
出口：无返回值。主要调用 move_man 进行操作
算法描述：无特殊算法
*/

/*
函数原型：int if_killed_by_bomb(int who);
函数功能：判断一个人有没有被炸弹炸死
*/

/*
```

```
函数原型：int die(int who);
函数：让人物死亡
*/

/*
函数原型：int born(int who);
函数功能：让人物出生
*/

/*    操作模块   在 key.c 中    */
/*
函数原型：int keypress(int who,int k);
函数功能：当键盘上有新的按键按下时，就把此按键加入队列，并将其设置为用户的当前操
          作 handle_direction。
*/

/*
函数原型：int keyrelease(int who,int k);
函数功能：当键盘上有按键释放时，将 keystate 设为释放状态，
      然后在队列中寻找上一个按下的、并仍然按着的按键，将这个按键设为用户操作
          handle_direction。
*/

/*      图形模块，  在 graph.c 文件中*/
/*
函数原型：int xy_to_ij(int xy)
函数功能：把屏幕上的坐标  转换成  地图上的格点坐标。
*/

/*
函数原型：int ij_to_xy(int ij)
函数功能：与上面函数相反
*/

/*
函数原型：void installGraph(void);
函数功能：图形模式初始化
*/

/*
函数原型：int draw_UI(int mode);
函数功能：绘制游戏界面，主要是右边的信息显示面板
*/
```

```
/*
函数原型：int draw_map(void);
函数功能：绘制游戏场景，主要包括墙、路面等
*/

/*
函数原型：int draw_player(int who,int y,int x);
函数功能：画人。并且随时间的不同，所画的手和脚也不同，实现动画
参数说明：
入口：人物序号 who，在屏幕上的坐标（y,x）
出口：无返回值。
算法描述：无特殊算法
*/

/*
函数原型：int draw_road(int y,int x);
函数功能：画路面
*/

/*
函数原型：int draw_wall(int y,int x);
函数功能：画墙
*/

/*
函数原型：int draw_stone(int y,int x);
函数功能：画坚不可摧的墙
*/

/*
函数原型：int draw_bomb(int y,int x);
函数功能：画炸弹
*/

/*
函数原型：int draw_blast(int y,int x);
函数功能：画爆炸图样
*/

/*
函数原型：int draw_pluslife(int y,int x);
函数功能：绘制加生命道具
*/
```

```
/*
函数原型：int draw_plusbomb(int y,int x);
函数功能：绘制加炸弹道具
*/

/*
函数原型：int draw_pluspower(int y,int x);
函数功能：绘制加炸弹威力道具
*/

/*
函数原型：int draw_blastall(int y,int x);
函数功能：绘制“引爆所有炸弹”道具
*/

/*
函数原型：int draw_exchange(int y,int x);
函数功能：画“换位”道具
*/

/*
函数原型：int draw_deathpotion(int y,int x);
函数功能：绘制“死亡药水”道具
*/

/*
函数原型：int draw(int j,int i,int state);
函数功能：绘制一个给定的格子。这是图形模块的入口函数
参数说明：
入口：格子的行与列（j,i）  和 要绘制的东西
出口：调用本文件的一些以“draw_”为前缀的函数
算法：无特殊算法
*/

/*
函数原型：int animat_bomb(void);
函数功能：实现炸弹的动画效果，让炸弹看起来有是一个有弹性的球
*/

/*
函数原型：int write_life(int who );
函数功能：显示当前生命值
*/
```

```
/*
函数原型：int write_bomb_num(int who);
函数功能：显示当前各玩家拥有的炸弹数
*/

/*
函数原型：int write_bomb_power(int who);
函数功能：显示当前各玩家的炸弹威力
*/;

/*    中断函数    在 inter.c 中*/

/*
函数原型：void interrupt newTimer();
函数功能：时间中断函数。
参数说明：无
算法描述：此函数将以 18.2Hz 的频率被调用，从而让计时器变量每过 1/18.2s 就增加 1
*/

/*
函数原型：void setTimer(void interrupt (*IntProc)());
函数功能：安装时间中断
*/;

/*
函数原型：void killTimer();
函数功能：卸载时间中断
*/;

/*
函数原型：void far interrupt NewInt9(void);
函数功能：键盘中断。有按键按下或释放时被调用
*/;

/*
函数原型：void InstallKeyboard(void);
函数功能：安装键盘中断
*/;

/*
函数原型：void ShutDownKeyboard(void);
函数功能：关闭键盘中断
*/;
```

```
/*    游戏运行函数    在 main.c 中*/
/*
函数原型：int init_game(int which_map);
函数功能：初始化游戏。把一切相关变量设为初始值，然后读取地图信息。
参数说明
入口：地图序号
出口：无
算法描述：无特殊算法
*/

/*
函数原型：int gameover();
函数功能：当检测到 loser > 0 时结束游戏
*/

/*
函数原型：  int pausegame();
函数功能：暂停游戏
*/

/*
函数原型：int GamePlay(void);
函数功能：游戏正式开始的入口
*/


/*    道具模块    在 things.c 中*/
/*
函数原型：int exchange2nums(int *a , int * b);
函数功能：交换 a,b 两数
*/

/*
函数原型：int random_thing();
函数功能：在地图上随机产生道具
*/

/*
函数原型：int pluslife(int who);
函数功能：“增加生命”道具效果
*/

/*
函数原型：int plusbomb(int who);
```

函数功能：“增加炸弹个数”道具效果
*/

/*
函数原型：int pluspower(int who);
函数功能：“增加炸弹威力”道具效果
*/

/*
函数原型：int blastall();
函数功能：“引爆所有炸弹”道具效果
*/

/*
函数原型：int exchange();
函数功能：“换位”道具效果
*/

/*
函数原型：int deathpotion(int who);
函数功能：“死亡药水”道具效果
*/

/*
函数原型：int things(int who,int j,int i);
函数功能：检查人物碰到的物品，并实现物品效果，然后把物品从地图上擦除。
*/

4. 核心控制和游戏动画

提交中期报告的时候，我们并没有计划做人物移动的动画，只打算每次移动一格就可以了。这样代码编写起来比较简单。但后来我们觉得，这样的游戏太简陋了。一件事情，要做，就要把事情做到最好。于是我们把人物的移动改成连续的移动。然而这么一改，就等于把整个程序的核心控制部分全部重写。例如，按照原来的计划，人物的坐标只需要记录在地图中的行和列即可，但现在要改成记录人在屏幕上的坐标，然后根据坐标再反过来计算他所在的格点，进而才能执行其他操作。不管难度多大，我们都重新规划，重新编码，最终完成了。

我们首先从键盘的按键开始说起。当按键按下，产生键盘中断，这当中有可能两个人的按键同时按下，甚至两个人分别同时按住一两个按键。这些复杂的信息，经过键盘操作模块（在 key.c 中），处理成简单的 handle_direction[]变量，输入给控制模块。例如 handle_direction[0]==2，表示第 0 号玩家的操作是向下（1 上，2 下，3 左，4 右）。

控制模块中，控制人物的 do_man()函数是以 18.2Hz 的频率被执行的。每次，它检测有没有操作，如果有，就向操作方向移动一小段距离。如果操作和上一次执行的操作不一样，就进入归位程序，回到离他最近的一个格子中，然后再按照新的方向行进。这样就实现了拐弯。

移动模块是控制模块的一个子模块，其中的 move_man()函数负责上文提到的“移动一小段距离”。因为人物的移动是连续的，因此一个人静止时踩在一格上，行走时踩在两格上。要移动时，先在这两格上画上路面，然后把人画在新的位置上，并重新计算他脚下是哪两个格子，为下次重画做准备。这样一次一次地重画连在一起，就形成了动画。

5. 游戏的 AI 模块

这个游戏能带给大家的最大乐趣也许就是实现了人机对战，那么要实现人机对战其核心就是一个人工智能算法，在 AI.c 中写了一个人工智能的引擎，其核心就是一个 A*寻路算法，再加上若干策略，经过多次修改，发现 AI 难度已经过大，就连我自己也好不容易才能损失 AI 的一条命。所以可能很多玩家在和电脑对战的时候会很不爽。AI 在设计的时候 Bug 较多，所以大部分时间都花在了改 Bug 上，所以对难度的控制上可能略有不足。后来就想了个办法，将游戏难度分为 3 级：简单：Easy，中等：Medium，疯狂：Crazy，这样我把不同时期改出来的 AI 放为不同难度，适当给玩家放点水，这样可以增加游戏的乐趣和互动性，游戏不至于太难。 总之，本游戏设计的 AI 大家仔细琢磨琢磨还是挺有意思的，AI 模块的具体情况在程序注释中已经写得很详细了，这里就不再赘述。

6. 16 位 BMP 贴图

做个帮助界面本来想一个个画的，但是后来觉得太麻烦，就考虑用贴图了。结果找了很多资料都没有找到好的贴图方法，有些网上的方法贴图结果是花屏。本想用第三方 BGI 的例程贴 256 色的图，但是由于我们使用 640*480 的分辨率发现如果不用 DOSBOX 根本运行不了，但是在 DOSBOX 下运行游戏速度却非常慢，严重影响游戏体验。所以就想找别的办法，后来选择了一种相对比较容易理解的贴图方法，当然效率也很低（导致贴图需要几秒钟，这中间背景音乐会卡几秒）。

用 Windows 自带的画图程序可以将图象保存为 16 色位图格式。在 16 色

位图中，前 118 个字 节保存的是图片信息，其后每 4 位保存一个象素颜色值，并按从下到上，从左到右的顺序保存。 这样只要用一个字符型的变量跳过前 118 个字节，在之后的区域内逐次读取，它的前 4 位即为 第一个象素的颜色值，后 4 位为第二个象素的颜色值，再用双重循环语句就能将位图中的数据 转成颜色值按顺序打印在屏幕上了。

但是 TC 的 VGA 模式默认的调色板与 Windows 画图程序的 16 色调色板保存的颜色值不尽相同， 且顺序也大不相同。虽然可以将调色板中的颜色索引值逐个替换，但是这样将要改写所有原先 使用的画图函数，且单寻找对应的颜色的索引值就将花去大量时间。最后只好将画图程序下的 16 种颜色对应的值与 TC 中的最相近的颜色值对应起来，最终得到的就是 putbmp()函数。 它将一个 640*480 的 16 色位图中每个象素值从文件中读取出来，并转换成相应的 TC 中的颜 色打印在屏幕上。

最后得到的打印图片效果虽然很慢（约需要 3 秒才能打印出整个屏幕的图片），但这样的屏幕从下往上刷新倒正好可以制造良好的效果。

7. 游戏道具

由于这个游戏非常依赖道具，有丰富道具才有乐趣，所以要定一些道具并能完全实现功能是一项难事，也是决定游戏成功与否的一个重要因素。 在构思的时候首先考虑到了基本道具，加水泡数量、加爆炸时水柱威力、加玩家生命等。但是增加玩家移动速度的道具不现实，因为程序很难控制这一项，不容易实现。考虑到公平性，也没有设计以上两样加到最大值的道具。道具我觉得还是不能炸掉好，因为道具中有好友坏，坏的道具玩家一定不愿意踩，这样留到最后激战时会有人不得不踩，这样更能增加乐趣。

当然特殊道具的安排才是主要的。首先为了混淆玩家，特地画了一个和加水柱威力道具图案差不多的道具，不过颜色不同，要是不小心碰了对方就偷笑了，因为那是死亡药水。为了增加游戏的乐趣，特地加了两个非常影响战局的道具，一个是瞬间引爆地图上所有水泡，它既可以加快自己炸砖块的速度，又可以找机会杀人，设计非常完美的道具。实现也比较容易，只需把所有水泡的计时改到快爆炸的时间就可以了。另一个是换位道具，顾名思义，两个玩家交换位置，试想某人偷放一个水泡，立刻换一把，另一玩家还摸不着头脑的时候已经被炸死了。这个实现起来看上去容易，不过老出 BUG，最后还是考虑到了很多问题，顺利解决掉了。不过我在怀疑这两个道具是否太 BT 了，影响游戏平衡性。

道具都随机产生，有随机函数控制，起初每次运行道具是一样的，后来才发现随机函数运用还要选随机数种子的，于是就加了一句 srand((unsigned long)time(NULL));总算使道具真正实现随机。

四、 系统设计难点及其解决方法

1、 如果要实现人物移动的动画，就要面临一个问题，人可能停在两个中间，这时候如果想要拐弯，就很难检测障碍物，很难安放炸弹了。解决办法是，每次释放按键，就让人回到某一个格子的正中央，然后才进行下一个操作。具体实施办法是，如果当前操作与当前速度方向不同，说明用户换操作，这时候进入归位程序。计算它应该归入那一格，并且计算他与这一格的偏移量（由count_delta()完成）然后按照这这个偏移量，分若干帧（即，分若干次执行）来回到这一格中央。

2、 程序调试的过程中，我们遇到的一个很大问题就是，总是出现花屏、图像撕裂的现象。在探索了很长时间才发现，这是因为在时间中断函数和主程序的循环中，都调用了绘图的函数。这样很可能主程序的循环刚执行到绘图函数，碰巧遇到了时间中断，在中断里也调用绘图函数。我们猜想，由于绘图函数本身内部的原因，造成花屏。意识到这点后，我们的改进方案是，把所有的事情全部集中到主程序中的游戏循环里，而时间中断函数里面只是给计时器变量赋值（仅留下Musiccount[mi]++; newtime = (newtime+1) % 30000;），问题就解决了。

3、 地图是需要一个边界的，否则人会走地图，炸弹的爆炸范围也会超出地图，这些都会造成记录地图信息的数组a[][]发生越界。简单的办法就是，给地图的最外面一圈赋值成3（也就是岩石）。但我们意识到这个问题的时候，已经从a的第0个元素开始使用，使用必须要给第-1个元素赋成3. 这怎么办呢？最后的解决办法是，把数组重新定义如下：

```
char **a , *b[MAP_SIZE+2] , c[MAP_SIZE+2][MAP_SIZE+2];
```

c是比地图大的二维数组，b是一维的指针数组，a是一个指向指针的指针。
先给b赋值为c的第二列的元素的地址，然后给a赋值为b的第二个元素的地址。
这样一来，就可以得到一个新的二维数组a，它的特点是：a[-1][..]和a[..][-1]也是有效的。
这时在给a[-1][..]和a[..][-1]赋值成3，就不会越界了。
这个方法是我们突然想出来的，也没指望它管用。但是试了之后，竟然成功了。

4、 游戏的人工智能部分的设计上碰倒的很多的困难，以前从未接触过人工智能，这次的AI是在学习了人工智能原理后的处女作，虽然设计并不困难，但是设计完成之后碰倒了很多意外，比如一开始AI根本不按我们设计的思想去做，这是后来在多次调试和修改下最终才完成了目前这个比较稳定的AI版本。

五、 不足之处

1. 游戏菜单设计比较简单，没有成型的菜单，影响一个游戏的外部形象。

2. 背景音乐是用机箱发声的，但是由于是用乐 谱写的，声音时间长度的控制不是太精确所以导致音乐很难听，这是受 TC 环境的限制而在现阶段无法完成的（虽然我看到有TC下使用中断用DSP芯片来播放wav的方法，但是由于这种方法程序复杂，而且可移植性差，而且wav格式本身占空间巨大，限于TC对内存的限制，最后不得不放弃此法）。

3. 道具画得不好，都只是一些简单的符号，甚至不看帮助都不知道意思。这还是受到TC限制，又没有太多时间去研究贴图，所以本游戏和用强大的ASL引擎做出来的游戏是肯定不能相比的。

4. 无法让玩家的自定义游戏按键。

5. 地图设计很少，也没有地图编辑器，来不及开发了，而且限于游戏现在存储地图的方式即使开发了地图编辑器使用也不是非常方便。只有3张地图的游戏显然比较乏味。

六、 人员分工说明

汪嘉恒	美工、AI 模块、整体测试、菜单、背景音乐
覃明	人物控制、中断、模块测试、动画、道具
陈向南	音乐乐谱、地图、实验报告