**高级语言程序设计**

**课设报告**

**题 目\_\_\_\_\_\_\_\_黄金矿工\_\_\_\_\_\_\_\_\_\_\_\_**

**学 号\_\_\_\_\_\_\_\_17071027\_\_\_\_\_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_马浩洋\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_\_\_陈彩\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_\_\_2018.5.2\_\_\_\_\_\_\_\_\_\_\_\_**

**成绩评价表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **界面功能** | **程序执行情况** | **程序功能实现** | **问题回答情况** | **最终成绩** |
| **□内容丰富、炫酷**  **□扩展基本要求**  **□基本完成基本要求**  **□未完成基本要求** | **□流畅**  **□界面有闪动**  **□操作不灵活**  **□有错误** | **□模块和数据组织合理**  **□合理、少量不妥**  **□基本合理**  **□不合理** | **□清楚、正确**  **□基本正确**  **□回答有部分错误**  **□不能回答问题** |  |
| **报告总体设计分析** | **报告技术实现分析** | **其他内容及写作规范** | **总体评价** | |
| **□深入、全面**  **□正确**  **□基本符合要求**  **□内容不全** | **□多个扩展功能**  **□少量扩展功能**  **□完成基本功能**  **□未完成基本功能** | **□清楚、正确**  **□基本正确**  **□回答有部分错误**  **□较混乱、不规范** |  | |

**教师签字:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

目录

[1 需求分析 4](#_Toc512871917)

[1.1 功能需求 4](#_Toc512871918)

[（1）基础功能 4](#_Toc512871919)

[（2）扩展功能 4](#_Toc512871920)

[1.2 数据需求 4](#_Toc512871921)

[（1）输入数据 4](#_Toc512871922)

[（2）中间数据 5](#_Toc512871923)

[（3）输出数据 5](#_Toc512871924)

[1.3界面需求 5](#_Toc512871925)

[（1）主菜单 5](#_Toc512871926)

[（2）游戏界面 5](#_Toc512871927)

[1.4 开发与运行环境需求 6](#_Toc512871928)

[（1）开发环境 6](#_Toc512871929)

[（2）运行环境 6](#_Toc512871930)

[2 游戏程序总体设计 6](#_Toc512871931)

[2.1 程序模块设计 6](#_Toc512871932)

[（1）主函数 6](#_Toc512871933)

[（2）用户界面交互与用户交互游戏 6](#_Toc512871934)

[（3）其它 10](#_Toc512871935)

[2.2 程序动态流程设计 11](#_Toc512871936)

[（1）模块调用图 11](#_Toc512871937)

[（2）主程序流程图 12](#_Toc512871938)

[2.3 主要数据结构 12](#_Toc512871939)

[（1）主要结构体类型 12](#_Toc512871940)

[（2）主要文件存储类型 13](#_Toc512871941)

[3 游戏实现技术点分析 13](#_Toc512871942)

[3.1 静态画面设计 13](#_Toc512871943)

[3.2 游戏动画设计 17](#_Toc512871944)

[3.3 游戏交互设计 20](#_Toc512871945)

[4 测试 21](#_Toc512871946)

[4.1 测试用例 21](#_Toc512871947)

[4.2 测试结果 22](#_Toc512871948)

[5 用户手册 24](#_Toc512871950)

[（1）应用程序功能 24](#_Toc512871951)

[（2）运行环境 24](#_Toc512871952)

[（3）安装与启动方法 24](#_Toc512871953)

[（4）界面交互及操作方法 24](#_Toc512871954)

[（5）友好化输入限制 24](#_Toc512871955)

[（6）交互命令 24](#_Toc512871956)

[6 总结提高 25](#_Toc512871957)

[（1）课程设计总结 25](#_Toc512871958)

[（2）课程建议 25](#_Toc512871959)

[附录一 课程设计代码量分布 26](#_Toc512871960)

[附录二 课程设计工作记录 26](#_Toc512871961)

[附录三 课程设计代码 27](#_Toc512871963)

# 需求分析

## 功能需求

### （1）基础功能

初始时记录玩家姓名信息，建立相关文件记录并存档。在矿洞的背景中，有一辆玩家控制的矿车和其他很多矿石（矿石的大小、种类不同，分值不同），由玩家用键盘A D（左右）控制自己的矿车移动，并用下键控制钩子抓取矿石。每抓取到一个矿石，矿石与钩子一起上升，并生成一个新的矿石。玩家用A D（左右）键控制自己的矿车向左、右方向移动。松开按键时，玩家控制的矿车就停止移动；玩家按下键控制钩子下降，抓到矿石则一起上升，反之则触底反弹，玩家控制的钩子抓到矿石，并的分值加分。玩家控制的矿车移动到屏幕边界时，不能继续往前移动，按反方向键返回。游戏设定时间为180秒，时间到则游戏结束，也可中途按下“空格”按钮结束游戏。在游戏运行过程中同时显示剩余时间和已得分数。

### （2）扩展功能

1）主菜单（带按钮效果）

2）帮助（游戏操作提示）

3）成就系统（与游戏积分相关）

4）排行榜系统（记录各玩家游戏成绩）

5）音乐及音效（不同界面具有不同音效）

6）游戏界面（带按钮效果、显示玩家信息、显示得分、最高得分、显示剩余时间）

7）新游戏

8）游戏菜单（游戏选项、帮助、成就、退出至主菜单、退出游戏）

9）游戏截图（实时截取游戏图片）

10）游戏相册（可以显示之前的截图）

11）矿石系统（分为大、中、小金块，岩石，钻石，魔法瓶，增加游戏的娱乐性）

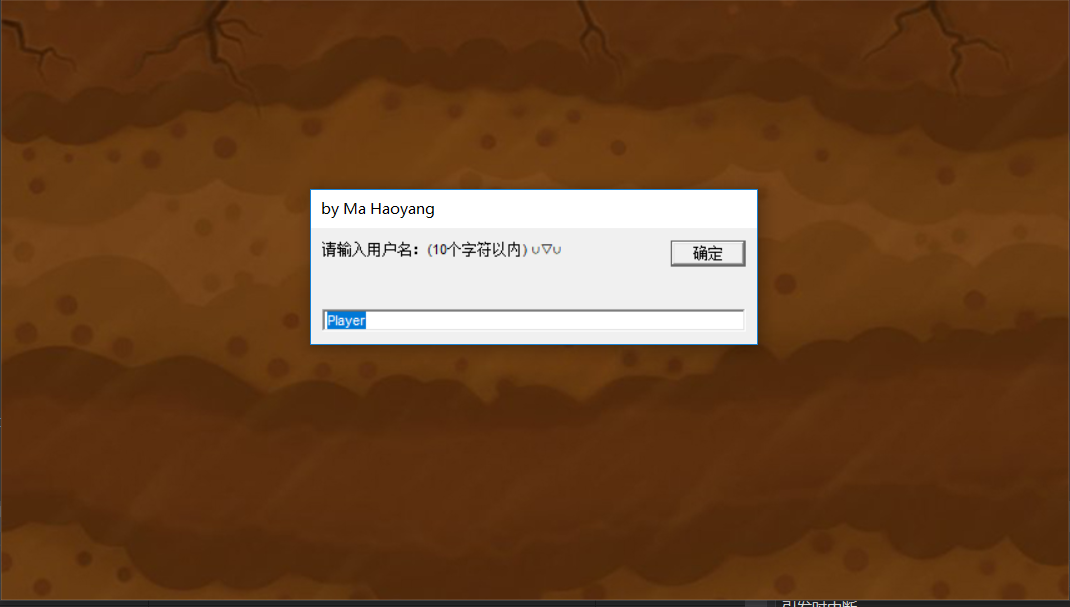
12）游戏提示

13）附中文注释

## 1.2 数据需求

## （1）输入数据

（1）玩家姓名（void）



**图1-2-1 游戏开始界面**

### （2）中间数据

1. 玩家的结构属性（struct player），包括玩家姓名与玩家分数
2. 种类的结构属性（struct kind），包括矿石的图片，掩码图，矿石对应分数
3. 矿石的结构属性（struct min），包括矿石横坐标，纵坐标，矿石种类
4. 链表的结构属性（struct node），构建矿石的链表，包括指向下一个节点的指针

### （3）输出数据

1. 游戏得分（int），输出游戏分数
2. 游戏排行（void），输出由分数决定的排名
3. 相册（void），输出游戏中的截图

## 1.3界面需求

### 主菜单



**图1-2-2 游戏菜单界面**

1）背景

2）按钮 A.开始游戏B.帮助C.成就D.相册E.退出

### （2）游戏界面



**图1-2-3 游戏界面**

1. 背景
2. 按钮 A.开始B.帮助C.主菜单

## 1.4 开发与运行环境需求

### （1）开发环境

1）操作系统

Windows 10（64 bit）

1. 开发平台

Visual Studio 2017

MindManager

1. 附加类库（插件）

easyX

### 运行环境

Windows 7 及以上的64位操作系统

# 游戏程序总体设计

## 程序模块设计

### 主函数

1. Main.cpp
2. int main(void)

### 用户界面交互与用户交互游戏

1）Judge.cpp

A．bool touch(min min1, min min2)

**参数**：设定两个结构体min，分别代表两个矿石

**返回值**：true/false

**功能**：判断两个图片是否有交叉

1. bool judge(node s, node\* H)

**参数**：设定结构体 node

**返回值**：true/false

**功能**：遍历矿石链表调用touch函数

1. void c\_min(node\* H, kind kinds[])

**参数**：设定结构体 node ，设定结构体 kind

**返回值**：无

**功能**：随机生成一个新的矿石

1. bool if\_right(node s, node\* H)

**参数**：设定结构体 node

**返回值**：true/false

**功能**：遍历矿石链表调用touch函数

1. bool caught(min g, int \*m1, int \*n)

**参数**：设定结构体 min，代表将要碰撞的矿物，设定两个int类型指针，分别代表矿车的横坐标，钩子的纵坐标

**返回值**：true/false

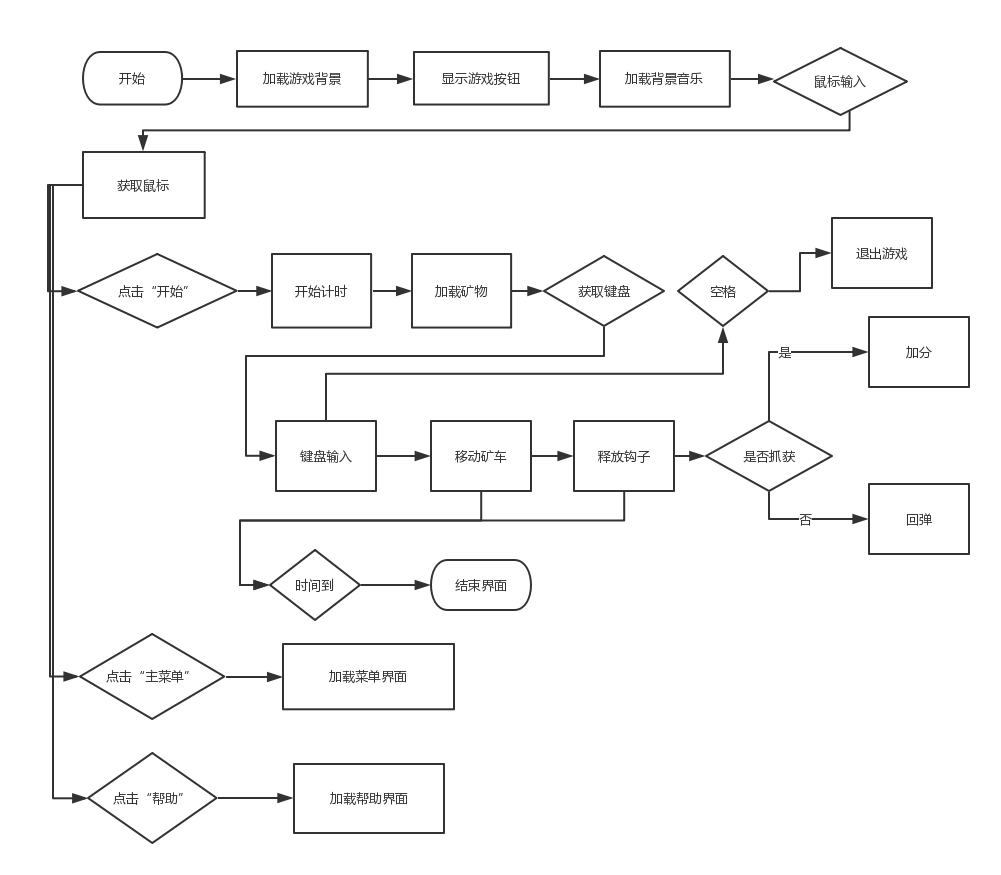
**功能**：判断矿石与钩子是否相遇

1. Game.cpp
2. void Game(struct player \* point\_player, int \*m1, int \*n,bool dec, kind kinds[],node\* H,int\* score, node\*h, node\*s)

**参数**：设定结构体类型指针player，设定两个int类型指针, 分别代表矿车的横坐标，钩子的纵坐标，设定布尔变量，设定结构体kind，设定结构体node

**返回值**：无

**功能**：实现游戏功能，分配按键指令

****

**图2-2-1 Game函数示意图**

1. Create.cpp
2. node\* create(kind kinds[],node\* h,node\* s)

**参数**：设定结构体kind，设定结构体node

**返回值**：H

**功能**: 创建矿石链表

1. Score.cpp
2. void a\_score(int \*score)

**参数**：设定int 类型指针，代表实时分数

**返回值**：无

**功能**: 显示当前分数

1. void sort(player players[], int n)

**参数**：设定player类型结构体，设定int类型变量，代表文件夹中分数的个数

**返回值**：无

**功能**: 对分数进行排序

1. void sort\_file()

**参数**：无

**返回值**：无

**功能**: 读入文件中的分数

1. void put\_rank()

**参数**：无

**返回值**：无

**功能**: 显示排行榜

1. Quit.cpp
2. int Quit(void)

**参数**：无

**返回值**：1

**功能**: 退出游戏

6)Enter.cpp

1. void input\_player\_info(struct player \* point\_player)

**参数**：设定结构体类型指针player

**返回值**：无

**功能**: 显示游戏规则，输入玩家姓名

1. void buffer()

**参数**：无

**返回值**：无

**功能**: 显示缓冲图片

1. int Enter()

**参数**：无

**返回值**：1

**功能**: 游戏缓冲

7）Time.cpp

1. void renew\_time(int time)

**参数**：设定int类型变量，代表系统时间

**返回值**：无

**功能**: 游戏计时

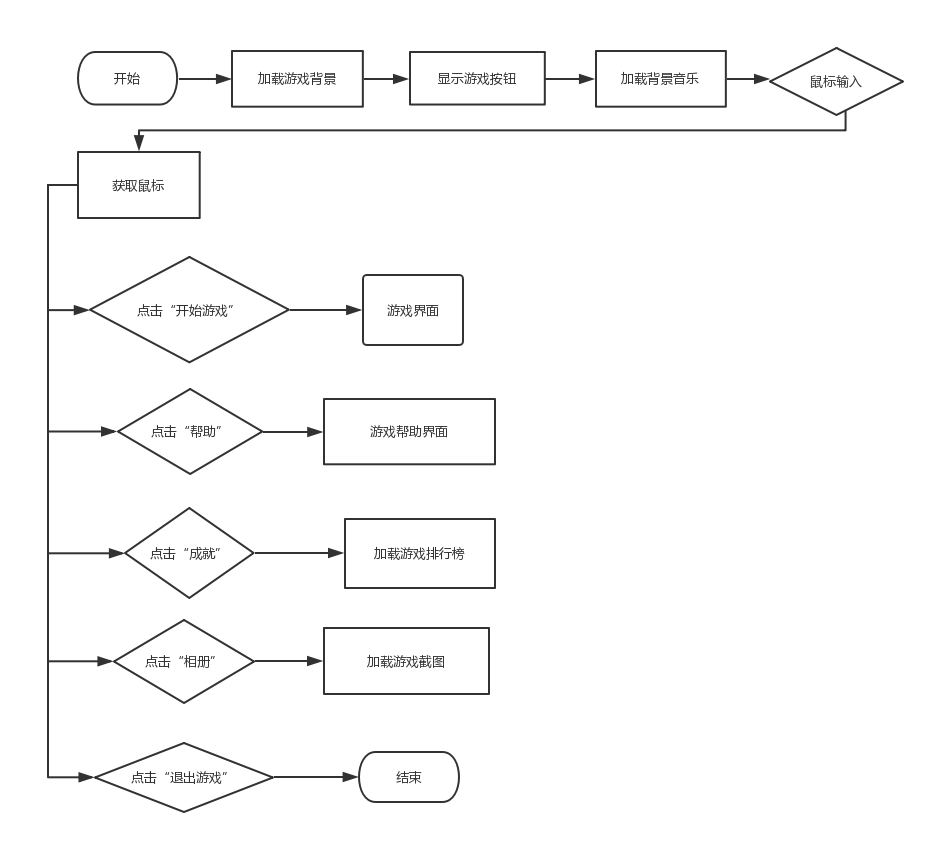
8）Menu.cpp

A. void Menu(struct player \* point\_player, int \*m1, int \*n,bool dec, kind kinds[],node\* H,int\* score, node\*h, node\*s)

**参数**：设定结构体类型指针player，设定两个int类型指针, 分别代表矿车的横坐标，钩子的纵坐标， 设定布尔变量，设定结构体kind，设定结构体node

**返回值**：无

**功能**：菜单功能（新游戏，帮助，成就，退出游戏）

****

**图2-2-2 Menu函数流程示意图**

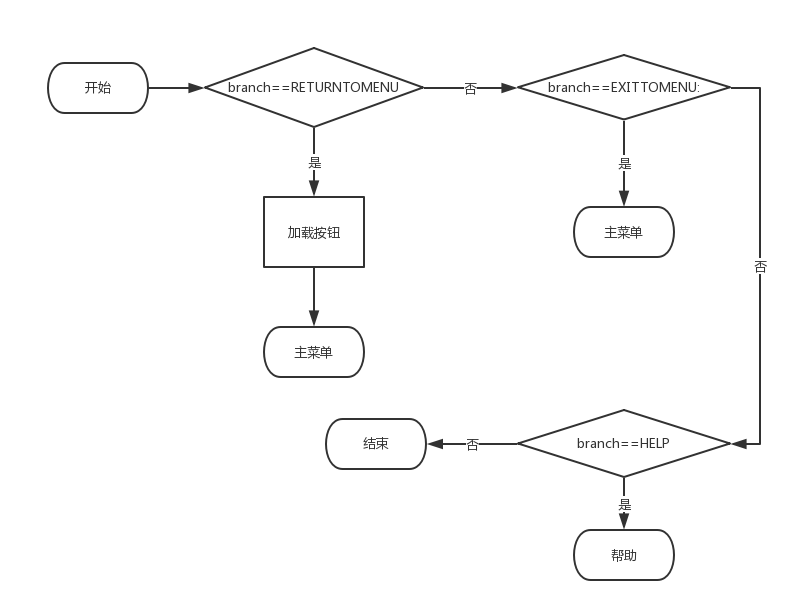
9）WorkSpace.cpp

A. int WorkSpace (struct player \* point\_player, int \*m1, int \*n,bool dec, kind kinds[],node\* H,int\* score, node\*h, node\*s)

**参数**：设定结构体类型指针player，设定两个int类型指针, 分别代表矿车的横坐标，钩子的纵坐标， 设定布尔变量，设定结构体kind，设定结构体node

**返回值**：1

**功能**：实现按钮与返回功能

****

**图2-2-3 Menu函数流程示意图**

10）Motion.cpp

A. int LorR(int Orientation, int \* m1, int \*time\_begin)

**参数**：设定int类型变量，设定一个int类型指针，代表矿车的横坐标

**返回值**：1

**功能**：实现矿车左右移动

1. int Go\_down(int \*m1, int \*n, int \*time\_begin, node\* H)

**参数**: 设定三个int类型指针, 分别代表矿车的横坐标，钩子的纵坐标，开始时系统时间值，设定结构体node

**返回值**：true/false

**功能**：实现钩子下降并判断是否与矿物相碰

1. int Null\_up(int \*m1, int \*n, int \*time\_begin)

**参数**: 设定三个int类型指针, 分别代表矿车的横坐标，钩子的纵坐标，开始时系统时间值

**返回值**：1

**功能**：实现钩子触底反弹

D. void Go\_up(node\* H, kind kinds[], int\* m1, int\* n, int\* time\_begin, int \*score)

**参数**: 设定三个int类型指针, 分别代表矿车的横坐标，钩子的纵坐标，开始时系统时间值，设定结构体kind，设定结构体node

**返回值**：1

**功能**：实现钩子与矿物一起上升

11) Graphics.cpp

1. int DisplayBackground(int branch)

**参数**: 设定int类型变量，代表背景图片种类

**返回值**：1

**功能**：加载背景

1. int OuttextScript()

**参数**: 无

**返回值**：1

**功能**：显示按钮

1. int OuttextChoice(int mode, int choice)

**参数**: 设定int类型变量，代表按钮种类

**返回值**：1

**功能**：显示按钮

D. void Outtextgame()

**参数**: 无

**返回值**：无

**功能**：显示按钮

### 其它

1）Screen.cpp

A. int ScreenShot()

**参数**: 无

**返回值**：1

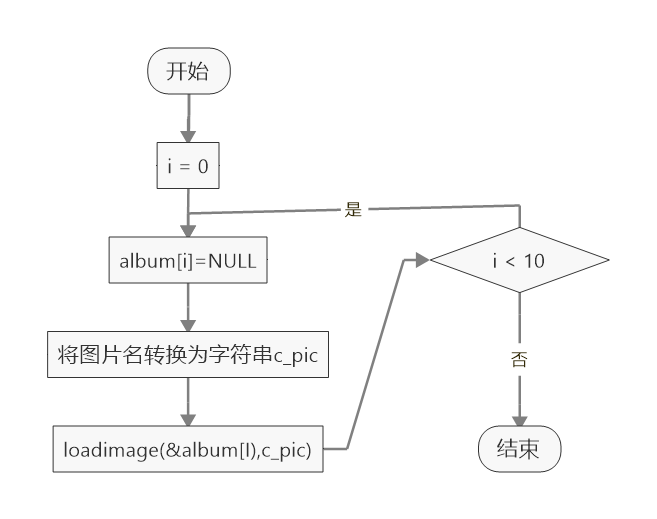
**功能**：获取屏幕截图

1. int LoadAlbum()

**参数**: 无

**返回值**：1

**功能**：展示屏幕截图



**图2-2-4 Screen函数流程示意图**

2) music.cpp

1. int StartBkMusic(int branch)

**参数**: 设定int型变量，代表音乐种类

**返回值**：1

**功能**：播放游戏界面音效

1. int SoundEffect(int branch)

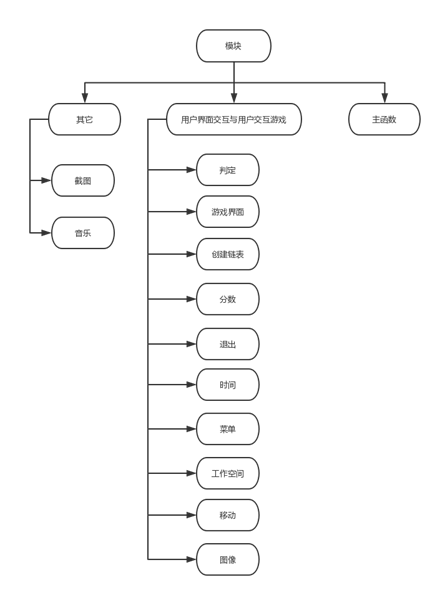
**参数**: 设定int型变量，代表音乐种类

**返回值**：1

**功能**：播放游戏特殊音效、

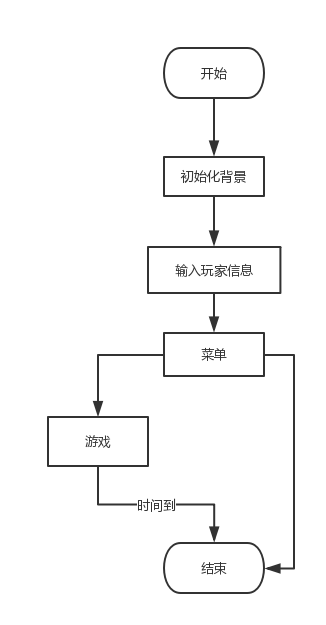
## 程序动态流程设计

### 模块调用图



**图2-2-1 模块调用图**

### 主程序流程图



**图2-2-2 主程序流程图**

## 主要数据结构

### 主要结构体类型

我定义了一个struct类型结构体来存储玩家的姓名与分数，如下：

struct player {

char playername[100] = { 0 };

int score;

};

对于游戏中所涉及的矿石系统我定义了三个struct类型结构体，分别用来存储矿石的图片，掩码图和分数，矿石的坐标与分数，矿石的量表，如下：

struct kind{

IMAGE pic\_1;

IMAGE pic\_2;

int score;

};

struct min{

int x;

int y;

kind min\_kind;

};

struct node{

min min\_s;

node\* next;

};

### 主要文件存储类型

对于玩家信息，分数和排行榜，我将他们存储在txt类型的文档里，如下：

void sort\_file(){

player players[10];

FILE\* fp = NULL;

int n = 0;

fp = fopen("record.txt", "r");

for(n=0;n<10;n++){

fscanf(fp, "%s %d", players[n].playername, &players[n].score);

}

sort(players, 10);

fclose(fp);

fp = fopen("record.txt", "w");

for (n = 0; n < 10; n++){

fprintf(fp, "%s %d\n", players[n].playername, players[n].score);

}

fclose(fp);

}

对于游戏中的截图，我将他们存储在一个文件夹中，如下：

int ScreenShot()

{

char c\_image\_screen\_shot[50];

IMAGE screen\_shot;

static int tag = 0;

getimage(&screen\_shot, 0, 0, 854, 490);

sprintf(c\_image\_screen\_shot, "screenshots/screen\_shot.jpg", 1);

saveimage(c\_image\_screen\_shot, &screen\_shot);

FlushConsoleInputBuffer(GetStdHandle(STD\_INPUT\_HANDLE));

return 1;

}

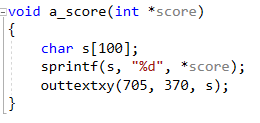
# 游戏实现技术点分析

### 静态画面设计

1）游戏界面分为三个部分：信息区，功能区，游戏区。

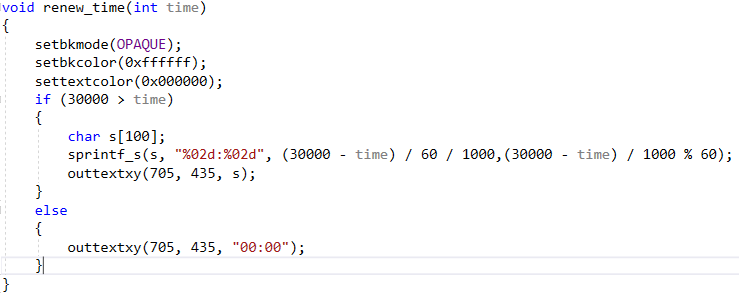
其中，信息区包括：玩家姓名，最高分数，当前得分，与游戏倒计时

定义一个char型数组，通过读取int型指针score，将实时分数投映在指定位置：



**图3-1-1 排行榜的实现**

根据int类型time的变化，在指定位置显示实时游戏剩余时间：



**图3-1-2 游戏时间的实现**

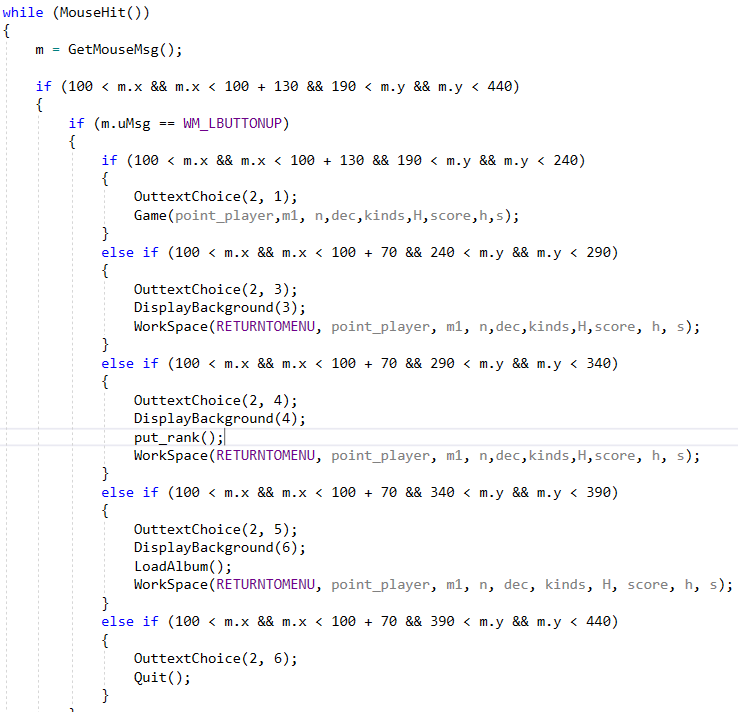
通过easyx库所带的InputBox函数，输入玩家姓名并存储在char型数组中：



**图3-1-3 游戏玩家信息输入的实现**

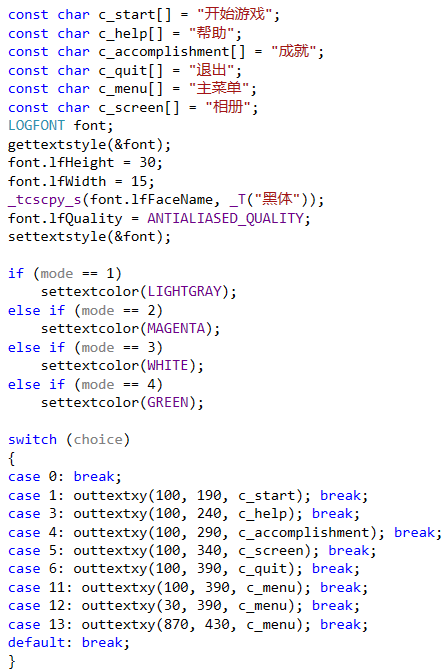
功能区包括：实现菜单与按钮功能，游戏内菜单功能。

获取当前鼠标，设置指定的点击范围，在不同的点击范围中输出不同的按钮和对应的功能函数，以此来实现菜单的功能：



**图3-1-4 菜单的实现**

引用前方的菜单功能，当鼠标点击对应范围时，则输出相同大小，相同内容，不同颜色的文字，以此达到实现按钮的功能：



**图3-1-5 菜单按钮的实现**

游戏区为游戏界面：玩家控制的矿车与矿石会在这里显示：



**图3-1-6 游戏界面背景**

2）矿石的加载

我在main中定义了所有矿石的种类与分数，在first.txt中定义了三个矿石的坐标与种类，当游戏开始时，会先创建一个链表，并在这个链表中读入first,txt中的三个值，在屏幕上形成三个矿石。

金块相关的结构体的实现：

struct kind

{

IMAGE pic\_1;

IMAGE pic\_2;

int score;

};

两个IMAGE分别存储了尽快的图片与掩码图，定义了金块的分数。

struct min

{

int x;

int y;

kind min\_kind;

};

定义了金块的横坐标与纵坐标，同时定义了金块的种类。

struct node

{

min min\_s;

node\* next;

};

链表的结构体：包含金块信息与后继指针。

金块种类的定义：

定义了一个kind类型的数组来定义金块种类：

kind kinds[6];

{

kinds[0].score = 10;

loadimage(&kinds[0].pic\_1, "images/mine\_rock.jpg");

loadimage(&kinds[0].pic\_2, "images/mine\_rock\_b.jpg");

kinds[1].score = 50;

loadimage(&kinds[1].pic\_1, "images/mine\_gold\_1.jpg");

loadimage(&kinds[1].pic\_2, "images/mine\_gold\_1\_b.jpg");

kinds[2].score = 100;

loadimage(&kinds[2].pic\_1, "images/mine\_gold\_2.jpg");

loadimage(&kinds[2].pic\_2, "images/mine\_gold\_2\_b.jpg");

kinds[3].score = 200;

loadimage(&kinds[3].pic\_1, "images/mine\_gold\_3.jpg");

loadimage(&kinds[3].pic\_2, "images/mine\_gold\_3\_b.jpg");

kinds[4].score = 500;

loadimage(&kinds[4].pic\_1, "images/diamond.jpg");

loadimage(&kinds[4].pic\_2, "images/diamond\_b.jpg");

kinds[5].score = 100;

loadimage(&kinds[5].pic\_1, "images/magic potion.jpg");

loadimage(&kinds[5].pic\_2, "images/magic potion\_b.jpg");

}

定义了链表的头结点，尾节点，通过对txt文件中数据的读取，以此来实现矿石系统链表的构建：

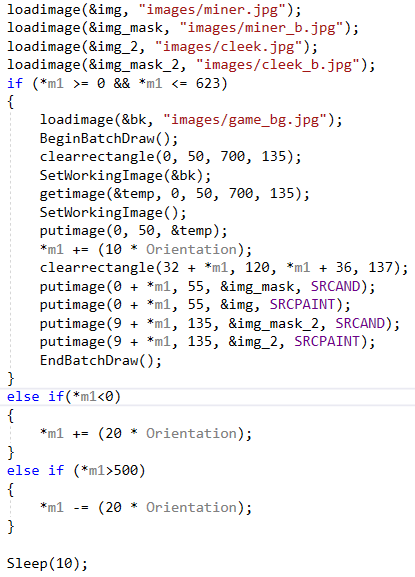


**图3-1-7 矿石链表的实现**

### 游戏动画设计

1. 矿车的移动

设定了矿车的坐标指针\*m，\*n，当矿车左右移动时，通过键盘输入的信息决定矿车的走向，通过擦除当前位置的画面，加载这一块的游戏背景，重新加载矿车图片并向指定方向移动固定的距离实现：

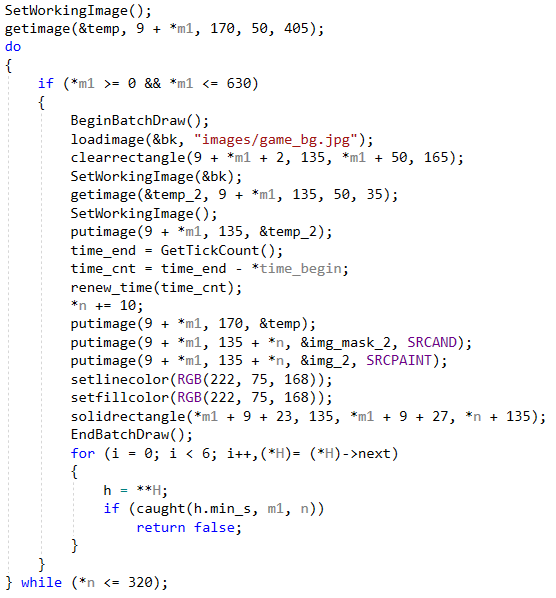


**图3-2-1 矿车移动的实现**

1. 钩子的下降与上升

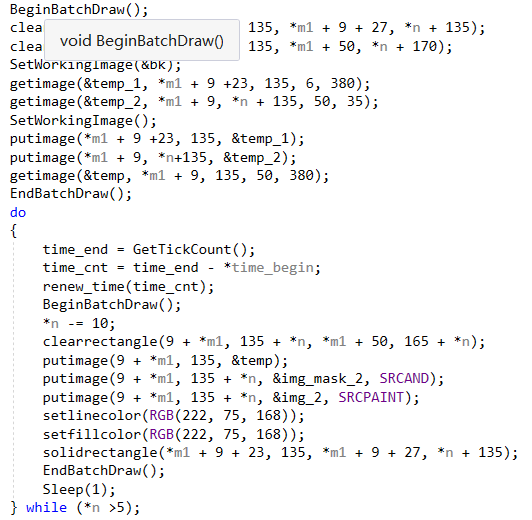
先截取钩子对应下的矩形图片，存储在temp中，同时擦除钩子的图片，同时加载temp并向下以定义速度加载钩子图片，在这个过程中，我们遍历金块的链表，如果钩子低端的坐标与金块的坐标重合，判定两者相撞，将两者一同提升，至矿车处删除这个金块节点；若未碰到金块，到达游戏界面低端时，钩子以更快的速度上升至矿车处（更改Sleep的时间）

当钩子下降时，首先限制钩子坐标\*n的改变范围，在这个范围内，先擦掉当前位置的钩子的图片，加载这一列的游戏背景图片，向下方指定距离加载钩子图片，在变化范围内重复此操作，并在钩子中间位置加载粉色色块当做钩子的绳子，同时遍历矿石链表判断钩子与矿石是否相碰：



**图3-2-2 钩子下降的实现**

当达到指定范围最大值时，若为碰到任何矿石，则擦除钩子所在位置图片，截取这一列的游戏图片并加载，向上方指定距离加载钩子图片，重复此操作，直到钩子的纵坐标减小为0：



**图3-2-3 钩子触底反弹的实现**

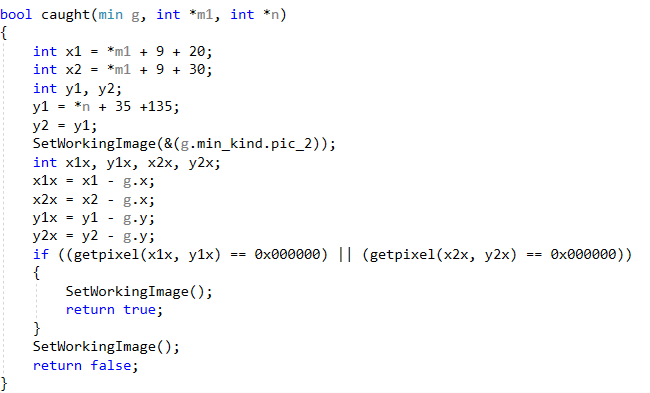
当判断到钩子与矿石的图片相碰时，读取矿石链表中对应矿石的数据，截取此列的游戏画面并加载，根据矿石种类的不同，将钩子与矿石同时向上移动不同的距离，重复此操作，直到钩子纵坐标为零，并删除此矿石在矿石链表中的节点，擦除矿石图片，并加载新的节点：



**图3-2-4 钩子与金块上升的实现**

1. 钩子与金块相撞的判定

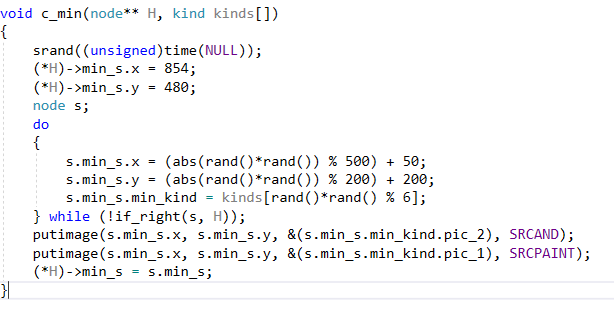
在钩子上定义两个坐标点，当钩子下降时，遍历整个金块的链表，我们将工作范围局限在金块的背景图上，当两个坐标点中有一点与金块相撞，即有一点坐标相对应的背景图为黑色，则两者相撞：



**图3-2-5 钩子与金块相撞的判定**

4)金块的删除与添加

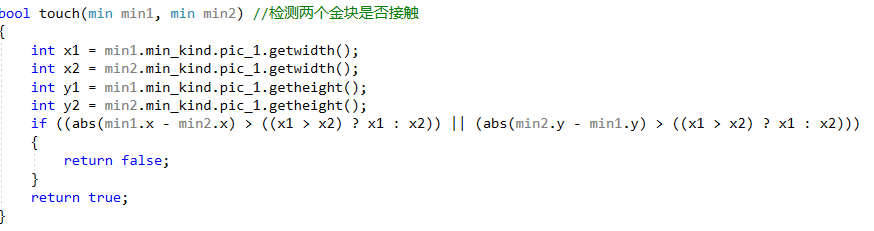
当金块到达矿车时，我们将图片移动到界面外，并删除这个金块所在的链表节点，新建立一个节点，并判断新金块与原有金块是否重叠。



**图3-2-6 矿石的添加与删除**

5）金块重叠的判断

读取两个图片的长宽，进而判断两图片是否重叠



**图3-2-7 金块重叠的判定**

1. 当矿车处于边界时的处理

定义矿车横坐标\*m的范围，当\*m值超出这个范围时，将矿车强制后退

else if(\*m1<0)

{

\*m1 += (20 \* Orientation);

}

else if (\*m1>500)

{

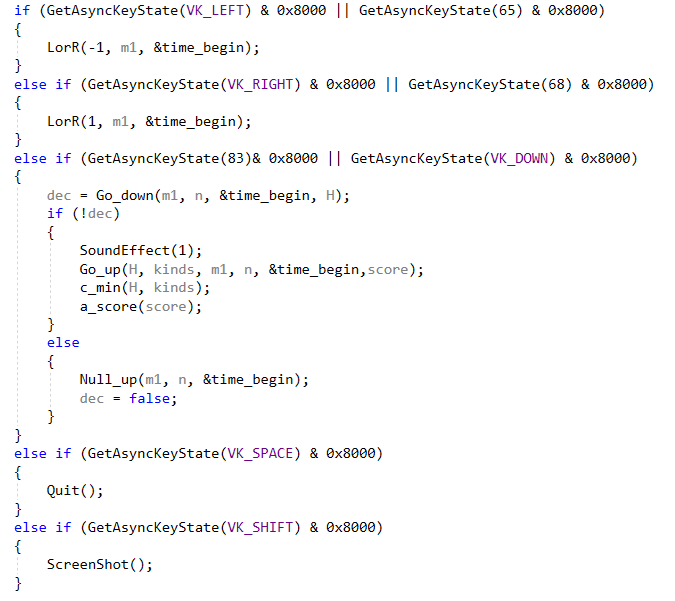
\*m1 -= (20 \* Orientation);

}

### 游戏交互设计

1. 命令控制台设计

首先，通过鼠标和键盘等输入设备接收信息，其次，每一个信息对应一个命令，根据不同的命令，进入相对应的函数执行对应的操作。

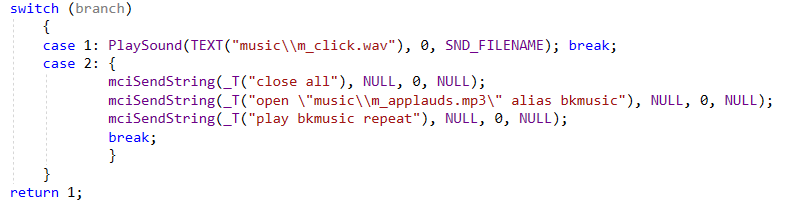


**图3-3-1 游戏的交互设计**

1. 游戏中音乐的交互

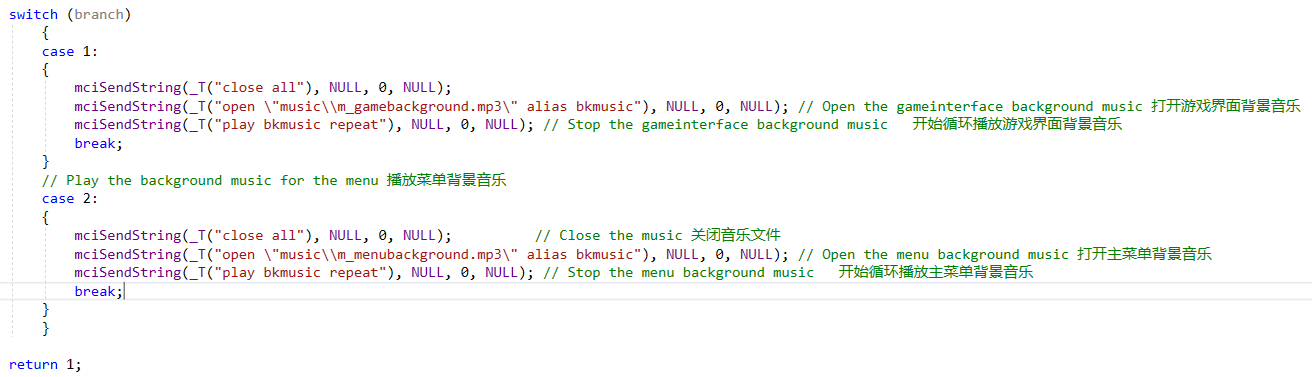
在这部分，用户通过更改页面会播放不同的音乐，在钩子与金块碰撞时也会播放对应的音乐各界面的音乐，根据界面的不同，通过返回值选取不同的音乐读取并循环播放；碰撞音乐则根据caught函数的返回值来判断播放与否。

当钩子与矿石相碰时，通过读取music文件夹中的指定音乐文件，来实现撞击音效：



**图3-3-2 游戏中的特殊音效**

当进入指定的界面时，通过读取music文件夹中的指定音乐文件，并循环播放，来实现界面的背景音效：

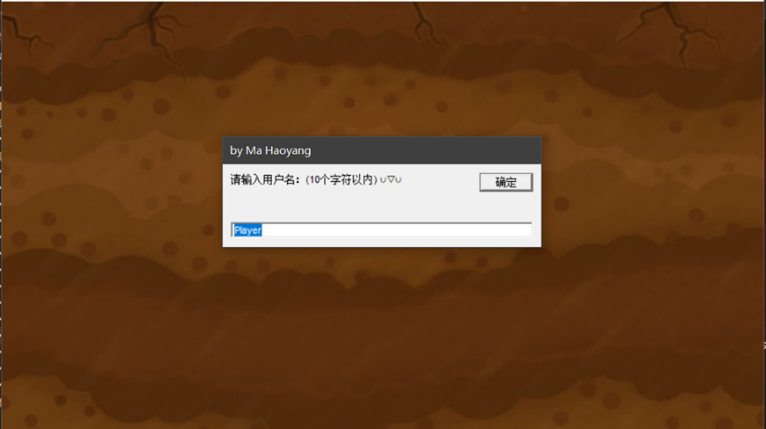


**图3-3-3 游戏中的界面音效**

# 测试

## 测试用例

初始输入玩家姓名“player”：



**图4-1-1 初始界面**

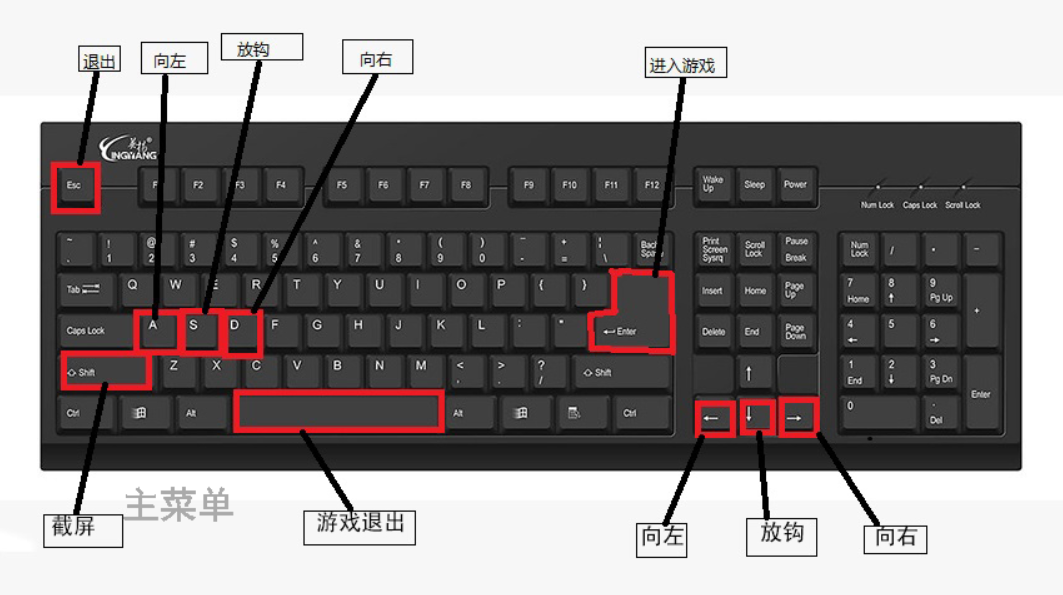
## 测试结果

进入主菜单，如下图：



**图4-2-1 游戏菜单**

点击“帮助”，查看帮助界面，如下图：



# ****图4-2-2 游戏帮助****

点击“主菜单”，返回菜单，点击“开始游戏”，进入游戏界面，如下图：



**图4-2-3 游戏界面**

继续点击“开始”，开始游戏，通过键盘进行游戏，如下图：



**图4-2-4 游戏进行画面**

时间到，游戏结束，结束界面如下图：



**图4-2-5 游戏结束界面**

点击返回主界面，点击“成就”查看排行，如下图：



**图4-2-6 游戏排行榜**

点击返回主菜单，点击“相册”，查看游戏截图，如下图：



**图4-2-7 游戏截图**

# 用户手册

### 应用程序功能

通过移动矿车和释放钩子来抓取矿石与魔法瓶

### 运行环境

Windows 7 及以上的64位操作系统

### 安装与启动方法

免安装版，直接解压压缩包，双击exe文件开始游戏

### 界面交互及操作方法

|  |  |  |  |
| --- | --- | --- | --- |
| S | 释放钩子 | Esc | 退出 |
| A | 向左移动 | Space | 游戏内退出 |
| D | 向右移动 | Shift | 截图 |
| ↓ | 释放钩子 | ← | 向左移动 |
| → | 向右移动 |

### 友好化输入限制

输入非中文字符，且不同时键入三个以上的有效键盘值

### 交互命令

在矿洞的背景中，有一辆玩家控制的矿车和其他很多矿石（矿石的大小、种类不同，分值不同），由玩家用键盘A D（左右）控制自己的矿车移动，并用下键控制钩子抓取矿石。每抓取到一个矿石，矿石与钩子一起上升，并生成一个新的矿石。

玩家用A D（左右）键控制自己的矿车向左、右方向移动。松开按键时，玩家控制的矿车就停止移动；玩家按下键控制钩子下降，抓到矿石则一起上升，反之则触底反弹，玩家控制的钩子抓到矿石，并把矿石拉到矿车上，根据该矿石的分值加分。玩家控制的矿车移动到屏幕边界时，不能继续往前移动，按反方向键返回。游戏设定时间为180秒，时间到则游戏结束，也可中途按下“Esc”按钮结束游戏。在游戏运行过程中同时显示剩余时间和已得分数。

# 总结提高

### 课程设计总结

在C语言课程设计这门课中，我独立完成了一个游戏应用程序的编写。在这个过程中，我通过自学和搜集资料，学会使用帮助手册，学会了一些可视化C编程的技巧。而且，我也十分感谢那些给予我帮助的同学，包括张世新，王天岳，刘涵天他们都解答了我的一些问题。

在这个过程中，我实践了模块化编程，实现了部分代码可重用，结合诸多游戏的特点创新性的提出了一些创新点，并勇于实践。并且由于课设占用了时间，我制定了时间表，合理规划了时间安排。

同时，我也发现了自己的一些不足，如之前学习的链表的部分知识点有些遗忘，并且有写一些冗余代码的习惯，并且容易在一个问题处停滞一段时间。但是，我都尽量努力克服并有所改进。

总之，我在课程设计中，学会了相关知识，也提高了我独立编程设计的能力。

### 课程建议

虽说我之前曾经用easyx插件做过游戏程序，但是对于之前没有接触过相关知识的同学，可能上手会有些困难，我在给其他同学解答问题时发现他们在一些基础概念上理解的有偏差，我建议可以像这学期组织几位有经验并且想分享经验的同学（全是我的舍友）给同学们分析一些相关知识，做成专题课堂。或者通过微信公众号推送一些学习资料。

# 附录一 课程设计代码量分布

**Header.h 66**

**Create.cpp 50**

**Enter.cpp 49**

**Game.cpp 101**

**Graphics.cpp 154**

**Judge.cpp 80**

**Main.cpp 44**

**Menu.cpp 49**

**Motion.cpp 203**

**Quit.cpp 17**

**Score.cpp 62**

**Time.cpp 18**

**Workspace.cpp 64**

**Music.cpp ­47**

**Screen.cpp 24**

# 附录二 课程设计工作记录

# 

# 附录三 课程设计代码

**// Header.h**

#include <conio.h>

#include <easyx.h>

#include <graphics.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include<windows.h>

#include<mmsystem.h>

#pragma comment(lib,"WINMM.LIB")

#define RETURNTOMENU 100

#define EXITTOMENU 101

#define HELP 102

struct player {

char playername[100] = { 0 };

int score;

};

struct kind

{

IMAGE pic\_1;

IMAGE pic\_2;

int score;

};

struct min

{

int x;

int y;

kind min\_kind;

};

struct node

{

min min\_s;

node\* next;

};

void renew\_time(int time); //更新系统时间

int Quit(void); //退出游戏

void input\_player\_info(struct player \* point\_player);

void buffer();

int Enter();

int DisplayBackground(int branch);

int OuttextScript();

int OuttextChoice(int mode, int choice);

void Outtextgame();

int WorkSpace(int branch, struct player \* point\_player, int \*m1, int\* n,bool dec, kind kinds[],node\*\* H, int\* score, node\*h, node\*s);

int LorR(int Orientation, int \* m1, int \*time\_begin);

int Go\_down(int \*m1, int \*n, int \*time\_begin, node\*\* H);

int Null\_up(int \*m1, int \*n, int \*time\_begin);

void Go\_up(node\*\* H, kind kinds[], int\* m1, int\* n,int\* time\_begin,int \*score);

bool caught(min g, int \*m1, int \*n);

bool touch(min min1, min min2);

bool judge(node s, node\*\* H);

void c\_min(node\*\* H, kind kinds[]);

bool if\_right(node s, node\*\* H);

void Menu(struct player \* point\_player, int \*m1, int \*n, bool dec,kind kinds[],node\*\* H, int\* score, node\*h, node\*s);

void Game(struct player \* point\_player, int \*m1, int \*n, bool dec,kind kinds[],node\*\* H,int\* score, node\*h, node\*s);

node\* create(kind kinds[],node\*h,node\*s);

void a\_score(int \*score);

void sort(player players[], int n);

void sort\_file();

void put\_rank();

int SoundEffect(int branch);

int StartBkMusic(int branch);

int ScreenShot();

int LoadAlbum();

**//Create.cpp**

#include"header.h"

node\* create(kind kinds[],node\* h,node\* s)

{

int i;;

FILE\* fpr = NULL;

int x;

int y;

int z;

int k;

node\* H = (node\*)malloc(sizeof(min));

static node a;

static node b[6];

fpr = fopen("first\_min.txt", "r");

fscanf(fpr, "%d", &x);

fscanf(fpr, "%d", &y);

fscanf(fpr, "%d", &z);

a.min\_s.x = x;

a.min\_s.y = y;

k = z;

a.min\_s.min\_kind = kinds[k];

putimage(x, y, &(a.min\_s.min\_kind.pic\_2), SRCAND);

putimage(x, y, &(a.min\_s.min\_kind.pic\_1), SRCPAINT);

H = &a;

s = (node\*)malloc(sizeof(node));

H->next = NULL;

s = H;

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

{

fscanf(fpr, "%d", &x);

fscanf(fpr, "%d", &y);

fscanf(fpr, "%d", &z);

b[i].min\_s.x = x;

b[i].min\_s.y = y;

k = z;

b[i].min\_s.min\_kind = kinds[k];

putimage(x, y, &(b[i].min\_s.min\_kind.pic\_2), SRCAND);

putimage(x, y, &(b[i].min\_s.min\_kind.pic\_1), SRCPAINT);

h = (node\*)malloc(sizeof(node));

h = &b[i];

h->next = NULL;

s->next = h;

s=h;

}

fclose(fpr);

s->next = H;

return H;

}

**//Enter.cpp**

#include"header.h"

void input\_player\_info(struct player \* point\_player)

{

//游戏规则

MessageBox(NULL, \_T("取得胜利的方法：规定时间内得到更高的分数\n\

初始时间为180秒\n\

抓到不同的矿物得到的分数不同\n\

抓到“魔法瓶”可以加5秒游戏时间\n\

点击\"开始\"后开始计时\n\

祝你得到高分，加油！^^")

, \_T("游戏规则"), MB\_ICONASTERISK | MB\_SYSTEMMODAL);

//输入用户名的InputBox

InputBox(point\_player->playername, 20, \_T("请输入用户名：(10个字符以内) ∪▽∪"),

\_T("by Ma Haoyang"), \_T("Player"), 0, 0, 1);

return;

}

void buffer()

{

IMAGE bg;

loadimage(&bg, "images/input\_bg.jpg");

putimage(0, 0, &bg);

}

int Enter()

{

IMAGE img;

MOUSEMSG m;

cleardevice();

loadimage(&img, "images/buffer\_bg.jpg");

putimage(0, 0, &img);

while (true)

{

while (MouseHit())

{

m = GetMouseMsg();

if (m.uMsg == WM\_LBUTTONUP) return 1;

if (\_kbhit()) return 1;

}

if (\_kbhit()) break;

}

return 1;

}

**//Game.cpp**

#include"header.h"

void Game(struct player \* point\_player, int \*m1, int \*n,bool dec, kind kinds[],node\*\* H,int\* score, node\*h, node\*s)

{

static int time\_begin;

int time\_end, time\_cnt = 0;

IMAGE bk;

IMAGE temp;

IMAGE img;

IMAGE img\_mask;

IMAGE img\_2;

IMAGE img\_mask\_2;

FILE \*fp = NULL;

MOUSEMSG a = GetMouseMsg();

loadimage(&bk, "images/game\_bg.jpg");

DisplayBackground(1);

setlinecolor(BLACK);

line(700, 215, 854, 215);

rectangle(737, 30, 817, 60);

rectangle(737, 90, 817, 120);

rectangle(737, 150, 817, 180);

Outtextgame();

outtextxy(705, 252, point\_player->playername);

loadimage(&img, "images/cleek.jpg");

loadimage(&img\_mask, "images/cleek\_b.jpg");

loadimage(&img\_2, "images/miner.jpg");

loadimage(&img\_mask\_2, "images/miner\_b.jpg");

putimage(0, 55, &img\_mask\_2, SRCAND);

putimage(0, 55, &img\_2, SRCPAINT);

putimage(9, 135, &img\_mask, SRCAND);

putimage(9, 135, &img, SRCPAINT);

const char c[] = "20000";

clearrectangle(32 + \*m1, 120, \*m1 + 36, 137);

StartBkMusic(1);

while (true)

{

if (MouseHit)

{

a = GetMouseMsg();

if (737 < a.x&&a.x < 817 && 30 < a.y&&a.y < 60)

{

if (a.mkLButton)

{

time\_begin = GetTickCount();

renew\_time(time\_cnt);

(\*H) = create(kinds, h, s);

while (true)

{

while (\_kbhit)

{

outtextxy(705, 310, c);

time\_end = GetTickCount();

time\_cnt = time\_end - time\_begin;

renew\_time(time\_cnt);

if (time\_cnt > 30 \* 1000)

{

SoundEffect(2);

fp = fopen("record.txt", "a+");

fprintf(fp, "%s ", point\_player->playername);

fprintf(fp, "%d\n", \*score);

fclose(fp);

\*score = 0;

DisplayBackground(5);

WorkSpace(RETURNTOMENU, point\_player, m1, n, dec, kinds, H, score, h, s);

}

if (GetAsyncKeyState(VK\_LEFT) & 0x8000 || GetAsyncKeyState(65) & 0x8000)

{

LorR(-1, m1, &time\_begin);

}

else if (GetAsyncKeyState(VK\_RIGHT) & 0x8000 || GetAsyncKeyState(68) & 0x8000)

{

LorR(1, m1, &time\_begin);

}

else if (GetAsyncKeyState(83)& 0x8000 || GetAsyncKeyState(VK\_DOWN) & 0x8000)

{

dec = Go\_down(m1, n, &time\_begin, H);

if (!dec)

{

SoundEffect(1);

Go\_up(H, kinds, m1, n, &time\_begin,score);

c\_min(H, kinds);

a\_score(score);

}

else

{

Null\_up(m1, n, &time\_begin);

dec = false;

}

}

else if (GetAsyncKeyState(VK\_SPACE) & 0x8000)

{

Quit();

}

else if (GetAsyncKeyState(VK\_SHIFT) & 0x8000)

{

ScreenShot();

}

}

}

}

}

else if (737 < a.x&&a.x < 817 && 90 < a.y&&a.y < 120)

{

WorkSpace(HELP, point\_player, m1, n, dec, kinds, H, score, h, s);

}

else if (737 < a.x&&a.x < 817 && 150 < a.y&&a.y < 180)

{

WorkSpace(RETURNTOMENU, point\_player, m1, n, dec, kinds, H, score, h, s);

}

}

}

}

**//Graphics.cpp**

#include"header.h"

int DisplayBackground(int branch)

{

IMAGE bk;

switch (branch)

{

case 1:

{

cleardevice();

loadimage(&bk, "images/game\_bg.jpg");

putimage(0, 0, &bk);

break;

}

case 2:

{

cleardevice();

loadimage(&bk, "images/menu\_bg.jpg");

putimage(0, 0, &bk);

break;

}

case 3:

{

cleardevice();

loadimage(&bk, "images/help.jpg");

putimage(0, 0, &bk);

break;

}

case 4:

{

cleardevice();

loadimage(&bk, "images/accomplishment\_bg.jpg");

putimage(0, 0, &bk);

break;

}

case 5:

{

cleardevice();

loadimage(&bk, "images/end\_game.jpg");

putimage(0, 0, &bk);

break;

}

case 6:

{

cleardevice();

loadimage(&bk, "images/input\_bg.jpg");

putimage(0, 0, &bk);

}

default:

break;

}

return 1;

}

int OuttextScript()

{

const char c\_start[] = "开始游戏";

const char c\_help[] = "帮助";

const char c\_accomplishment[] = "成就";

const char c\_quit[] = "退出";

const char c\_menu[] = "主菜单";

const char c\_gold[] = "黄金矿工";

const char c\_screen[] = "相册";

LOGFONT font;

gettextstyle(&font);

font.lfHeight = 30;

font.lfWidth = 15;

\_tcscpy\_s(font.lfFaceName, \_T("黑体"));

font.lfQuality = ANTIALIASED\_QUALITY;

settextstyle(&font);

settextcolor(WHITE);

setbkmode(TRANSPARENT);

outtextxy(110, 65, c\_gold);

outtextxy(100, 190, c\_start);

outtextxy(100, 240, c\_help);

outtextxy(100, 290, c\_accomplishment);

outtextxy(100, 340, c\_screen);

outtextxy(100, 390, c\_quit);

return 1;

}

int OuttextChoice(int mode, int choice)

{

const char c\_start[] = "开始游戏";

const char c\_help[] = "帮助";

const char c\_accomplishment[] = "成就";

const char c\_quit[] = "退出";

const char c\_menu[] = "主菜单";

const char c\_screen[] = "相册";

LOGFONT font;

gettextstyle(&font);

font.lfHeight = 30;

font.lfWidth = 15;

\_tcscpy\_s(font.lfFaceName, \_T("黑体"));

font.lfQuality = ANTIALIASED\_QUALITY;

settextstyle(&font);

if (mode == 1)

settextcolor(LIGHTGRAY);

else if (mode == 2)

settextcolor(MAGENTA);

else if (mode == 3)

settextcolor(WHITE);

else if (mode == 4)

settextcolor(GREEN);

switch (choice)

{

case 0: break;

case 1: outtextxy(100, 190, c\_start); break;

case 3: outtextxy(100, 240, c\_help); break;

case 4: outtextxy(100, 290, c\_accomplishment); break;

case 5: outtextxy(100, 340, c\_screen); break;

case 6: outtextxy(100, 390, c\_quit); break;

case 11: outtextxy(100, 390, c\_menu); break;

case 12: outtextxy(30, 390, c\_menu); break;

case 13: outtextxy(870, 430, c\_menu); break;

default: break;

}

if (mode == 2)

Sleep(500);

return 1;

}

void Outtextgame()

{

const char c\_start[] = "开始";

const char c\_pause[] = "帮助";

const char c\_menu[] = "主菜单";

const char c\_1[] = "玩家姓名";

const char c\_2[] = "最高分";

const char c\_3[] = "当前得分";

const char c\_4[] = "倒计时";

LOGFONT font;

gettextstyle(&font);

font.lfHeight = 20;

font.lfWidth = 10;

\_tcscpy\_s(font.lfFaceName, \_T("黑体"));

font.lfQuality = ANTIALIASED\_QUALITY;

settextstyle(&font);

settextcolor(BLACK);

setbkmode(TRANSPARENT);

outtextxy(757, 35, c\_start);

outtextxy(757, 95, c\_pause);

outtextxy(747, 155, c\_menu);

outtextxy(705, 225, c\_1);

outtextxy(705, 285, c\_2);

outtextxy(705, 345, c\_3);

outtextxy(705, 405, c\_4);

}

**//Judge.cpp**

#include"header.h"

bool touch(min min1, min min2) //检测两个金块是否接触

{

int x1 = min1.min\_kind.pic\_1.getwidth();

int x2 = min2.min\_kind.pic\_1.getwidth();

int y1 = min1.min\_kind.pic\_1.getheight();

int y2 = min2.min\_kind.pic\_1.getheight();

if ((abs(min1.x - min2.x) > ((x1 > x2) ? x1 : x2)) || (abs(min2.y - min1.y) > ((x1 > x2) ? x1 : x2)))

{

return false;

}

return true;

}

bool judge(node s, node\*\* H) //遍历链表

{

int i;

for (i = 0; i < 6; i++, (\*H) = (\*H)->next)

{

if (touch((\*H)->min\_s, s.min\_s))

{

return false;

}

}

return true;

}

void c\_min(node\*\* H, kind kinds[])

{

srand((unsigned)time(NULL));

(\*H)->min\_s.x = 854;

(\*H)->min\_s.y = 480;

node s;

do

{

s.min\_s.x = (abs(rand()\*rand()) % 500) + 50;

s.min\_s.y = (abs(rand()\*rand()) % 200) + 200;

s.min\_s.min\_kind = kinds[rand()\*rand() % 6];

} while (!if\_right(s, H));

putimage(s.min\_s.x, s.min\_s.y, &(s.min\_s.min\_kind.pic\_2), SRCAND);

putimage(s.min\_s.x, s.min\_s.y, &(s.min\_s.min\_kind.pic\_1), SRCPAINT);

(\*H)->min\_s = s.min\_s;

}

bool if\_right(node s, node\*\* H)

{

int i;

for (i = 0; i < 6; i++, (\*H) = (\*H)->next)

{

if (touch((\*H)->min\_s, s.min\_s))

{

return false;

}

}

return true;

}

bool caught(min g, int \*m1, int \*n)

{

int x1 = \*m1 + 9 + 20;

int x2 = \*m1 + 9 + 30;

int y1, y2;

y1 = \*n + 35 +135;

y2 = y1;

SetWorkingImage(&(g.min\_kind.pic\_2));

int x1x, y1x, x2x, y2x;

x1x = x1 - g.x;

x2x = x2 - g.x;

y1x = y1 - g.y;

y2x = y2 - g.y;

if ((getpixel(x1x, y1x) == 0x000000) || (getpixel(x2x, y2x) == 0x000000))

{

SetWorkingImage();

return true;

}

SetWorkingImage();

return false;

}

**//Menu.cpp**

#include"header.h"

void Menu(struct player \* point\_player, int \*m1, int \*n,bool dec, kind kinds[], node\*\* H, int\* score, node\*h, node\*s)

{

StartBkMusic(2);

char ch;

MOUSEMSG m = GetMouseMsg();

DisplayBackground(2);

OuttextScript();

while (true)

{

if (\_kbhit()) { ch = \_getch(); if (ch == 27) Quit(); }

while (MouseHit())

{

m = GetMouseMsg();

if (100 < m.x && m.x < 100 + 130 && 190 < m.y && m.y < 440)

{

if (m.uMsg == WM\_LBUTTONUP)

{

if (100 < m.x && m.x < 100 + 130 && 190 < m.y && m.y < 240)

{

OuttextChoice(2, 1);

Game(point\_player,m1, n,dec,kinds,H,score,h,s);

}

else if (100 < m.x && m.x < 100 + 70 && 240 < m.y && m.y < 290)

{

OuttextChoice(2, 3);

DisplayBackground(3);

WorkSpace(RETURNTOMENU, point\_player, m1, n,dec,kinds,H,score, h, s);

}

else if (100 < m.x && m.x < 100 + 70 && 290 < m.y && m.y < 340)

{

OuttextChoice(2, 4);

DisplayBackground(4);

put\_rank();

WorkSpace(RETURNTOMENU, point\_player, m1, n,dec,kinds,H,score, h, s);

}

else if (100 < m.x && m.x < 100 + 70 && 340 < m.y && m.y < 390)

{

OuttextChoice(2, 5);

DisplayBackground(6);

LoadAlbum();

WorkSpace(RETURNTOMENU, point\_player, m1, n, dec, kinds, H, score, h, s);

}

else if (100 < m.x && m.x < 100 + 70 && 390 < m.y && m.y < 440)

{

OuttextChoice(2, 6);

Quit();

}

}

}

}

}

}

**//Motion.cpp**

# #include"header.h"

int LorR(int Orientation, int \* m1, int \*time\_begin)

{

IMAGE temp;

IMAGE bk;

IMAGE img;

IMAGE img\_mask;

IMAGE img\_2;

IMAGE img\_mask\_2;

int time\_end, time\_cnt;

time\_end = GetTickCount();

time\_cnt = time\_end - \*time\_begin;

renew\_time(time\_cnt);

loadimage(&img, "images/miner.jpg");

loadimage(&img\_mask, "images/miner\_b.jpg");

loadimage(&img\_2, "images/cleek.jpg");

loadimage(&img\_mask\_2, "images/cleek\_b.jpg");

if (\*m1 >= 0 && \*m1 <= 623)

{

loadimage(&bk, "images/game\_bg.jpg");

BeginBatchDraw();

clearrectangle(0, 50, 700, 135);

SetWorkingImage(&bk);

getimage(&temp, 0, 50, 700, 135);

SetWorkingImage();

putimage(0, 50, &temp);

\*m1 += (10 \* Orientation);

clearrectangle(32 + \*m1, 120, \*m1 + 36, 137);

putimage(0 + \*m1, 55, &img\_mask, SRCAND);

putimage(0 + \*m1, 55, &img, SRCPAINT);

putimage(9 + \*m1, 135, &img\_mask\_2, SRCAND);

putimage(9 + \*m1, 135, &img\_2, SRCPAINT);

EndBatchDraw();

}

else if(\*m1<0)

{

\*m1 += (10 \* Orientation);

}

else if (\*m1>623)

{

\*m1 -= (10 \* Orientation);

}

Sleep(10);

return 1;

}

int Go\_down(int \*m1, int \*n, int \*time\_begin, node\*\* H)

{

int i;

node h;

IMAGE temp;

IMAGE temp\_2;

IMAGE bk;

IMAGE img;

IMAGE img\_mask;

IMAGE img\_2;

IMAGE img\_mask\_2;

int time\_end, time\_cnt;

loadimage(&img, "images/miner.jpg");

loadimage(&img\_mask, "images/miner\_b.jpg");

loadimage(&img\_2, "images/cleek.jpg");

loadimage(&img\_mask\_2, "images/cleek\_b.jpg");

SetWorkingImage();

getimage(&temp, 9 + \*m1, 170, 50, 405);

do

{

if (\*m1 >= 0 && \*m1 <= 630)

{

BeginBatchDraw();

loadimage(&bk, "images/game\_bg.jpg");

clearrectangle(9 + \*m1 + 2, 135, \*m1 + 50, 165);

SetWorkingImage(&bk);

getimage(&temp\_2, 9 + \*m1, 135, 50, 35);

SetWorkingImage();

putimage(9 + \*m1, 135, &temp\_2);

time\_end = GetTickCount();

time\_cnt = time\_end - \*time\_begin;

renew\_time(time\_cnt);

\*n += 10;

putimage(9 + \*m1, 170, &temp);

putimage(9 + \*m1, 135 + \*n, &img\_mask\_2, SRCAND);

putimage(9 + \*m1, 135 + \*n, &img\_2, SRCPAINT);

setlinecolor(RGB(222, 75, 168));

setfillcolor(RGB(222, 75, 168));

solidrectangle(\*m1 + 9 + 23, 135, \*m1 + 9 + 27, \*n + 135);

EndBatchDraw();

for (i = 0; i < 6; i++,(\*H)= (\*H)->next)

{

h = \*\*H;

if (caught(h.min\_s, m1, n))

return false;

}

}

} while (\*n <= 320);

return true;

}

int Null\_up(int \*m1, int \*n, int \*time\_begin)

{

IMAGE temp\_1;

IMAGE temp\_2;

IMAGE temp;

IMAGE bk;

IMAGE img;

IMAGE img\_mask;

IMAGE img\_2;

IMAGE img\_mask\_2;

int time\_end, time\_cnt;

loadimage(&img, "images/miner.jpg");

loadimage(&img\_mask, "images/miner\_b.jpg");

loadimage(&img\_2, "images/cleek.jpg");

loadimage(&img\_mask\_2, "images/cleek\_b.jpg");

loadimage(&bk, "images/game\_bg.jpg");

BeginBatchDraw();

clearrectangle(\*m1 + 9 + 23, 135, \*m1 + 9 + 27, \*n + 135);

clearrectangle(\*m1 + 9, \*n + 135, \*m1 + 50, \*n + 170);

SetWorkingImage(&bk);

getimage(&temp\_1, \*m1 + 9 +23, 135, 6, 380);

getimage(&temp\_2, \*m1 + 9, \*n + 135, 50, 35);

SetWorkingImage();

putimage(\*m1 + 9 +23, 135, &temp\_1);

putimage(\*m1 + 9, \*n+135, &temp\_2);

getimage(&temp, \*m1 + 9, 135, 50, 380);

EndBatchDraw();

do

{

time\_end = GetTickCount();

time\_cnt = time\_end - \*time\_begin;

renew\_time(time\_cnt);

BeginBatchDraw();

\*n -= 10;

clearrectangle(9 + \*m1, 135 + \*n, \*m1 + 50, 165 + \*n);

putimage(9 + \*m1, 135, &temp);

putimage(9 + \*m1, 135 + \*n, &img\_mask\_2, SRCAND);

putimage(9 + \*m1, 135 + \*n, &img\_2, SRCPAINT);

setlinecolor(RGB(222, 75, 168));

setfillcolor(RGB(222, 75, 168));

solidrectangle(\*m1 + 9 + 23, 135, \*m1 + 9 + 27, \*n + 135);

EndBatchDraw();

Sleep(1);

} while (\*n >5);

return 1;

}

void Go\_up(node\*\* H, kind kinds[], int\* m1, int\* n, int\* time\_begin, int \*score)

{

node h;

IMAGE temp;

IMAGE temp1;

IMAGE bk;

IMAGE img;

IMAGE img\_mask;

IMAGE img\_2;

IMAGE img\_mask\_2;

int time\_end, time\_cnt;

time\_end = GetTickCount();

time\_cnt = time\_end - \*time\_begin;

renew\_time(time\_cnt);

loadimage(&img, "images/miner.jpg");

loadimage(&img\_mask, "images/miner\_b.jpg");

loadimage(&img\_2, "images/cleek.jpg");

loadimage(&img\_mask\_2, "images/cleek\_b.jpg");

loadimage(&bk, "images/game\_bg.jpg");

loadimage(&bk, "images/game\_bg.jpg");

BeginBatchDraw();

clearrectangle(\*m1 + 9, 135 + \*n, 50 + \*m1, 165 + \*n);

clearrectangle((\*H)->min\_s.x, (\*H)->min\_s.y, (\*H)->min\_s.x + (\*H)->min\_s.min\_kind.pic\_1.getwidth(), (\*H)->min\_s.y + (\*H)->min\_s.min\_kind.pic\_1.getheight());

SetWorkingImage(&bk);

getimage(&temp, 9 + \*m1, 135 + \*n, 50, 35);

getimage(&temp1, (\*H)->min\_s.x, (\*H)->min\_s.y, (\*H)->min\_s.min\_kind.pic\_1.getwidth(), (\*H)->min\_s.min\_kind.pic\_1.getheight());

SetWorkingImage();

int left, right;

left = ((\*H)->min\_s.x) < (\*m1) ? (\*H)->min\_s.x : \*m1;

right = ((\*H)->min\_s.x + (\*H)->min\_s.min\_kind.pic\_1.getwidth()) >(\*m1 + 50) ? ((\*H)->min\_s.x + (\*H)->min\_s.min\_kind.pic\_1.getwidth()) : (\*m1 + 50);

putimage(\*m1 + 9, \*n + 135, &temp);

putimage((\*H)->min\_s.x, (\*H)->min\_s.y, &temp1);

getimage(&temp, left, 135, right - left + 1, (\*H)->min\_s.min\_kind.pic\_1.getheight() + 1);

EndBatchDraw();

while (\*n >= 0)

{

BeginBatchDraw();

clearrectangle(9 + \*m1, 135 + \*n, \*m1 + 50, 165 + \*n);

clearrectangle((\*H)->min\_s.x, (\*H)->min\_s.y, (\*H)->min\_s.x + (\*H)->min\_s.min\_kind.pic\_1.getwidth(), (\*H)->min\_s.y + (\*H)->min\_s.min\_kind.pic\_1.getheight());

SetWorkingImage(&bk);

getimage(&temp, 9 + \*m1, 135 + \*n, 50, 35);

getimage(&temp1, (\*H)->min\_s.x, (\*H)->min\_s.y, (\*H)->min\_s.min\_kind.pic\_1.getwidth() + 2, (\*H)->min\_s.min\_kind.pic\_1.getheight() + 2);

SetWorkingImage();

putimage(9 + \*m1, 135 + \*n, &temp);

putimage((\*H)->min\_s.x, (\*H)->min\_s.y, &temp1);

\*n -= 10;

(\*H)->min\_s.y -= 10;

putimage((\*H)->min\_s.x, (\*H)->min\_s.y, &((\*H)->min\_s.min\_kind.pic\_2), SRCAND);

putimage((\*H)->min\_s.x, (\*H)->min\_s.y, &((\*H)->min\_s.min\_kind.pic\_1), SRCPAINT);

putimage(\*m1 + 9, \*n + 135, &img\_mask\_2, SRCAND);

putimage(9 + \*m1, 135 + \*n, &img\_2, SRCPAINT);

time\_end = GetTickCount();

time\_cnt = time\_end - \*time\_begin;

renew\_time(time\_cnt);

EndBatchDraw();

Sleep((\*H)->min\_s.min\_kind.score);

}

\*score += (\*H)->min\_s.min\_kind.score;

BeginBatchDraw();

clearrectangle(\*m1 + 9, 135 + \*n, 50 + \*m1, 165 + \*n);

clearrectangle((\*H)->min\_s.x, (\*H)->min\_s.y, (\*H)->min\_s.x + (\*H)->min\_s.min\_kind.pic\_1.getwidth(), (\*H)->min\_s.y + (\*H)->min\_s.min\_kind.pic\_1.getheight());

SetWorkingImage(&bk);

getimage(&temp, 9 + \*m1, 135 + \*n, 50, 35);

getimage(&temp1, (\*H)->min\_s.x, (\*H)->min\_s.y, (\*H)->min\_s.min\_kind.pic\_1.getwidth() + 2, (\*H)->min\_s.min\_kind.pic\_1.getheight() + 2);

SetWorkingImage();

putimage((\*H)->min\_s.x, (\*H)->min\_s.y, &temp1);

putimage(9 + \*m1, 135 + \*n, &temp);

clearrectangle(32 + \*m1, 120, \*m1 + 36, 137);

putimage(9 + \*m1, 135, &img\_mask\_2, SRCAND);

putimage(9 + \*m1, 135, &img\_2, SRCPAINT);

EndBatchDraw();

}

**//Quit.cpp**

#include"header.h"

int Quit(void)

{

HWND ifexit = GetHWnd();

if ((MessageBoxA(ifexit, "是否退出当前游戏？", "退出游戏", MB\_YESNO | MB\_ICONQUESTION)) == 6)

exit(0);

return 1;

}

**//Score.cpp**

#include"header.h"

void a\_score(int \*score)

{

char s[100];

sprintf(s, "%d", \*score);

outtextxy(705, 370, s);

}

void sort(player players[], int n)

{

int i, j;

player temp;

for (i = 0; i <n-1; i++)

for (j = i+1; j < n; j++)

if (players[i].score<players[j].score)

{

temp = players[j];

players[j] = players[i];

players[i] = temp;

}

}

void sort\_file()

{

player players[10];

FILE\* fp = NULL;

int n = 0;

fp = fopen("record.txt", "r");

for(n=0;n<10;n++)

{

fscanf(fp, "%s %d", players[n].playername, &players[n].score);

}

sort(players, 10);

fclose(fp);

fp = fopen("record.txt", "w");

for (n = 0; n < 10; n++)

{

fprintf(fp, "%s %d\n", players[n].playername, players[n].score);

}

fclose(fp);

}

void put\_rank()

{

FILE\* fp = NULL;

sort\_file();

fp = fopen("record.txt", "r");

int i;

char name[30];

int score = 0;

settextstyle(30, 0, \_T("宋体"));

char s[60];

for (i = 0; i < 2; i++)

{

fscanf(fp, "%s %d", name, &score);

sprintf(s, "%-10s%7d", name, score);

outtextxy(150+100\*i, 250 + 125 \* i, s);

}

fclose(fp);

}

**//Time.cpp**

#include"header.h"

void renew\_time(int time)

{

setbkmode(OPAQUE);

setbkcolor(0xffffff);

settextcolor(0x000000);

if (180000 > time)

{

char s[100];

sprintf\_s(s, "%02d:%02d:%02d", (180000 - time) / 60 / 1000,(180000 - time) / 1000 % 60, (180000 - time) % 1000 / 10);

outtextxy(705, 435, s);

}

else

{

outtextxy(705, 435, "00:00:00");

}

}

**//Workspace.cpp**

#include"header.h"

int WorkSpace(int branch, struct player \* point\_player, int \*m1, int\* n,bool dec, kind kinds[], node\*\* H, int\* score, node\*h, node\*s)

{

char ch;

MOUSEMSG m;

switch (branch)

{

case RETURNTOMENU: OuttextChoice(1, 11); break;

}

while (true)

{

if (\_kbhit())

{

ch = \_getch();

switch (branch)

{

case RETURNTOMENU:

default:

if (ch == 27) { Sleep(500); FlushMouseMsgBuffer(); return 1; }

}

}

while (MouseHit())

{

m = GetMouseMsg();

if (branch == RETURNTOMENU && 100 < m.x && m.x < 100 + 95 && 390 < m.y && m.y < 420) OuttextChoice(4, 11);

if (m.uMsg == WM\_LBUTTONUP)

{

switch (branch)

{

case RETURNTOMENU:

if (100 < m.x && m.x < 100 + 95 && 390 < m.y && m.y < 420) { OuttextChoice(2, 11); Menu(point\_player, m1, n,dec,kinds,H,score, h, s); }

break;

case EXITTOMENU:

if (737 < m.x&&m.x < 817 && 150 < m.y&&m.y < 180) { Menu(point\_player, m1, n,dec,kinds,H,score, h, s); }

break;

case HELP:

if (737 < m.x&&m.x < 817 && 90 < m.y&&m.y < 120)

{

DisplayBackground(3);

WorkSpace(RETURNTOMENU, point\_player, m1, n, dec, kinds, H, score, h, s);

}

}

}

else

{

switch (branch)

{

case RETURNTOMENU: OuttextChoice(1, 11); break;

}

}

}

}

return 1;

}

**//Music.cpp**

#include"header.h"

int SoundEffect(int branch)

{

switch (branch)

{

case 1: PlaySound(TEXT("music\\m\_click.wav"), 0, SND\_FILENAME); break;

case 2: {

mciSendString(\_T("close all"), NULL, 0, NULL);

mciSendString(\_T("open \"music\\m\_applauds.mp3\" alias bkmusic"), NULL, 0, NULL);

mciSendString(\_T("play bkmusic repeat"), NULL, 0, NULL);

break;

}

}

return 1;

}

int StartBkMusic(int branch)

{

switch (branch)

{

case 1:

{

mciSendString(\_T("close all"), NULL, 0, NULL);

mciSendString(\_T("open \"music\\m\_gamebackground.mp3\" alias bkmusic"), NULL, 0, NULL); // Open the gameinterface background music 打开游戏界面背景音乐

mciSendString(\_T("play bkmusic repeat"), NULL, 0, NULL); // Stop the gameinterface background music 开始循环播放游戏界面背景音乐

break;

}

// Play the background music for the menu 播放菜单背景音乐

case 2:

{

mciSendString(\_T("close all"), NULL, 0, NULL); // Close the music 关闭音乐文件

mciSendString(\_T("open \"music\\m\_menubackground.mp3\" alias bkmusic"), NULL, 0, NULL); // Open the menu background music 打开主菜单背景音乐

mciSendString(\_T("play bkmusic repeat"), NULL, 0, NULL); // Stop the menu background music 开始循环播放主菜单背景音乐

break;

}

}

return 1;

}

int StopBkMusic(void)

{

mciSendString(\_T("stop bkmusic"), NULL, 0, NULL); // Stop the gameinterface background music 停止播放游戏界面背景音乐

mciSendString(\_T("close all"), NULL, 0, NULL); // Close the music 关闭音乐文件

return 1;

}

**//Screen.cpp**

#include"header.h"

int ScreenShot()

{

char c\_image\_screen\_shot[50];

IMAGE screen\_shot;

static int tag = 0;

getimage(&screen\_shot, 0, 0, 854, 490);

sprintf(c\_image\_screen\_shot, "screenshots/screen\_shot.jpg", 1);

saveimage(c\_image\_screen\_shot, &screen\_shot);

FlushConsoleInputBuffer(GetStdHandle(STD\_INPUT\_HANDLE));

return 1;

}

int LoadAlbum()

{

char c\_pic[50];

IMAGE album;

sprintf(c\_pic, "screenshots/screen\_shot.jpg", 1);

loadimage(&album, c\_pic);

putimage(0, 0, &album);

return 1;

}