C 语言实现推箱子

邓凯旋

（华北科技学院，河北 廊坊 065201）

摘　要：C 语言作为一种高级语言，贴近底层、程序执行的效率高，多应用于嵌入式开发，也可以用来开发应用软件、游戏等。推箱子是一款非常经典的小游戏，玩法多样化、内涵丰富。为了探究软件工程的开发逻辑，查阅了相关资料、搜索了图片素材，了解了图形化编程，借助 MySQL 数据库、Easyx 图形库，利用 VS2019 编写了推箱子游戏。

关键词：C 语言；Eacyx；推箱子；数据库

1　编程原理

所使用的的背景图片尺寸为 1190*800 像素 、其余均为 61*61 像素。使用 MySQL 数据库设计用户表和关卡等级表，且事先插入用户信息和关卡地图。在控制台界面登录时验证用户身份信息，获取当前所在关卡，进入游戏。

根据游戏规则：通过键盘按键推箱子，箱子只能推动而不能拉动；如果箱子前一格是地板或箱子目的地，则可以推动一个箱子往前走一格，如果箱子已经在箱子目的地则不能再推动；所有箱子都成功推到箱子目的地，并且移动步数在规定步数之内，本关卡结束，连接到数据库，更新数据库表中的当前关卡字段，读取下一关地图，直到通关。

2　项目实现

2.1　道具表示

墙: 0，地板: 1，箱子目的地: 2, 小人: 3, 箱子: 4, 箱子命中目标: 并把代表数字存放在二位数组里。不同地图行列不同，此处 LINE 和 COLUMN 表示允许的最大值，具体还要根据从数据库中读取的行列初始化地图。

```
int map[LINE][COLUMN];
```

2.2　地图初始化

初始化“画布”的尺寸和背景图片一样，“画布”太小，会造成图片显示不全；太大，会有黑色阴影。之后再加载图片并放到“画布”上。

```
    initgraph(960,768);
        loadimage(&bg_image,_T("blackground.bmp"),
960,768,true);//true 代表是否拉伸图片
    putimage(0,0,&bg_image);
loadimage (&images [WALL],_T ("wall.bmp"),RATIO,RA-
TIO,true);
    loadimage (&images [FL OOR],_T ("floor.bmp"),RA-
TIO,RATIO,true);
    loadimage (&images [BOX_DES],_T ("des.bmp"),
RATIO,RATIO,true);
    loadimage (&images [MAN],_T ("man.bmp"),RATIO,
RATIO,true);
    loadimage (&images [BOX],_T("box.bmp"),RATIO,
RATIO,true);
    loadimage (&images [HIT],_T ("box.bmp"),RATIO,
RATIO,true);
for(int i = 0; i< levlel.map_row; i++){
        for(int j = 0; j<level.map_column; j++){
            if(map[i][j] == MAN){
                man.x = i;
                man.y = j;
            }
            putimage (100 +j*RATIO,150 +i*RATIO,
&images[map[i][j]]);
        }
    }
```

2.3　获取用户信息，验证后登录

```
bool fetch_level_info(levelinfo& level, int level_id) {
    MYSQL mysql;
    MYSQL_RES* res; //查询结果集
    MYSQL_ROW row;  //记录结构体
    char sql[256];
bool ret = false;
    //1.连接到数据库
    if (connect_db(mysql) == false) {
        return false;
    }
    //2.根据关卡 id 查询数据库获取关卡地图信息
```

作者简介：邓凯旋（1995–），男，硕士，研究方向：安全工程。

```
        snprintf (sql, 256, "select name, map_row, map_column, map_data, next_level_id from levels where id=%d;", level_id);
    ret = mysql_query(&mysql, sql); //成功返回 0
    if (ret) {
        printf (" 数据库查询出错,%s 错误原因：%s\n", sql, mysql_error(&mysql));
        mysql_close(&mysql);
        return false;
    }
    //3.获取结果
    res = mysql_store_result(&mysql);
    row = mysql_fetch_row(res);
    if (row == NULL) {//没有查找到记录
        mysql_free_result(res);
        mysql_close(&mysql);
        return false;
    }
    level.id = level_id;
    level.name = row[0];
    level.map_row = atoi(row[1]);
    level.map_column = atoi(row[2]);
    level.map_data = row[3];
    level.next_level = atoi(row[4]);
    if (debug) printf("level id: %d name: %s map row: %d map column: %d map data: %s next level:%d\n",level.id, level.name.c_str (), level.map_row, level.map_column,level.map_data.c_str(), level.next_level);
    //释放结果集
    mysql_free_result(res);
    //关闭数据库
    mysql_close(&mysql);
    return true;
}
```

2.4 更新关卡信息

```
bool update_user_level (userinfo& user, int next_level_id) {
    //根据用户 id 更新下一关的 level_id
     snprintf(sql, 256, "update users set level_id = %d where id=%d;", next_level_id, user.id);
}
```

2.5 热键控制

Conio.h 库函数是 Console Input/Output（控制台输入输出）的简写，其中定义了通过控制台进行数据输入和数据输出的函数，主要是一些用户通过按键盘产生的对应操作。

Kbhit() 函数检查当前是否有键盘输入，若有则返回一个非 0 值，否则返回 0。

getch() 函数，从控制台读取一个字符，但不显示在屏幕上。

上下左右退出分别用‘w’、‘a’、‘s’、‘d’、‘q’键控制。

```
#include <conio.h>
#define KEY_UP ´w´ //char ´a´
#define KEY_LEFT ´a´
#define KEY_RIGHT ´d´
#define KEY_DOWN ´s´
#define KEY_QUIT ´q´
//游戏环节
bool quit = false;
do {
    if(_kbhit()){ //玩家按键
    char ch = _getch();
    if(ch == KEY_UP){
    gameControl(UP);
    }else if(ch == KEY_DOWN){
    gameControl(DOWN);
    }else if(ch == KEY_LEFT){
    gameControl(LEFT);
    }else if(ch == KEY_RIGHT){
    gameControl(RIGHT);
    }else if(ch == KEY_QUIT){
    quit = true;
    }
    }
Sleep(100);
}while(quit==false);
```

由于推箱子游戏完全由方向键控制，所以应检测玩家按下的按键，并产生一系列动作。而每一系列动作模式基本一样，只是具体的动作不同，所以定义了函数 void gameControl (enum _DIRECT direct) 来实现。实参就是具体的方向，为枚举类型。

该函数包括：因为推箱子时需考虑下个位置是地板还是墙壁，甚至下个位置的下个位置也要考虑，所以实例化 3 个结构体。以“小人”当前位置为基准，当按下方向键时，就可以确定下个位置的地点。

接下来具体移动要分几种场景。

当前位置是“小人”，下个位置是地板，为正常移动。当前位置是“小人”，下个位置是箱子，若下下个

位置是目的地，则实现把箱子推到目的地上。 推完后人离开，恢复“现场”。

```
void gameControl(enum _DIRECT direct){
    int x = man.x;
    int y = man.y;
    struct _POS next_pos  = man;
    struct _POS next_next_pos = man;
    struct _POS last_pos = man;
    switch (direct){
    case UP:
        next_pos.x--;
        next_next_pos.x -=2;
        break;
    case DOWN:
        next_pos.x++;
        next_next_pos.x +=2;
        break;
    case LEFT:
        next_pos.y--;
        next_next_pos.y -=2;
        break;
    case RIGHT:
        next_pos.y++;
        next_next_pos.y +=2;
        break;
    }
    if (isValid (next_pos) &&map [man.x][man.y] ==
MAN &&map[next_pos.x][next_pos.y] == FLOOR ){
    changeMap( &next_pos,MAN);
        changeMap(&man,FLOOR);
        man = next_pos;
        count++;
            }  else if(isValid(next_next_pos) &&map
[next_pos.x][next_pos.y] == BOX  ){
          if( map[next_next_pos.x][next_next_pos.y]
== FLOOR){
changeMap( &next_pos,MAN);
                    changeMap(&man,FLOOR);
                changeMap( &next_next_pos,BOX);
                    man = next_pos;
                    count++;
            }
                    else if ( map[next_next_pos.x]
[next_next_pos.y] == BOX_DES ){
                    changeMap(&man,FLOOR);
                    changeMap( &next_pos,MAN);
                changeMap( &next_next_pos,BOX);
                    man = next_pos;
                    count++;
                }
        }
 else if ( isValid (next_pos) && map [man.x][man.y] ==
BOX_DES &&map[next_pos.x][next_pos.y]== FLOOR ){
            changeMap( &next_pos,MAN);
            changeMap(&man,BOX_DES);
            man = next_pos;
            count++;
        }
}
```

2.6 游戏结束函数说明

本关卡结束判断函数：bool isGameOver()。“小人”将箱子推到目的地上，在程序上是通过把目的地改成箱子来实现的，所以行列遍历判断地图上是否还有目的地标号“1”，若没有，则完成任务，游戏结束。

闯关成功的画面实现函数：void gameoverSceen (IMAGE *bg_image)，参数是背景图片变量的指针，实现过程是先显示背景图片来刷新控制台界面，然后再设置字体格式，在背景图片水平和竖直方向正中画一个矩形，在矩形中显示闯关成功即可。

同理，闯关失败函数 void failSceen（IMAGE *bg_image）实现步骤与之相同。

3 结语

利用数据库实现了推箱子游戏，对存储用户信息更加安全可靠，对数据的增删改查更加方便快捷。包含了初始化场景、键盘控制、关卡切换和结束场景显示等模块。深入分析了实现的逻辑过程，对于开发这类的软件有很高的借鉴作用。

参考文献

[1] 谢玉庚. 推箱子游戏目标间遮掩关系的分析 [J] . 电脑编程技巧与维护，2018，(04)：4-7.

[2] 潘惠勇，夏敏捷. Java 实现 2.5D 推箱子游戏 [J] . 电脑编程技巧与维护，2014，(19)：28-31+56.

[3] 董航，饶世钧，洪俊. 基于 MySQL 的雷达目标信息数据库构建 [J] . 科技创新与应用，2020，(28)：80-83.

[4] 彭剑，刘艳松，唐闻. MySQL 主从服务器数据库同步的实现 [J] . 福建电脑，2020，36 (07)：118-119.