3.7 按键控制俄罗斯方块_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.7 按键控制俄罗斯方块涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

```
void key_control()
{
    int ch;
    while(1){
            ch = getch();
            if(ch == 'q' || ch == 'Q'){
                    break;
            }else if(ch == '\r'){

                fall_down();
            }else if(ch == '\33'){
                ch = getch();
                if(ch == '['){
                    ch = getch();
                    switch(ch){
                            case 'A':
                                    change_shape();
                                    break;
                            case 'B':
                                    move_down(dynamic_num,dynamic_mode);
                                    break;
                            case 'C':
                                    move_right(dynamic_num,dynamic_mode);
                                    break;
                            case 'D':
                                    move_left(dynamic_num,dynamic_mode);
                                    break;
                            default:
                                    break;
                    }
                }
        }
    }

    game_over();
}



int change_shape()
{
        int m = (dynamic_mode+1)%4;

        if(dynamic_x+2*(4-shape[dynamic_num][m][16])-1 > 39)
                return 1;

        if(dynamic_y+(4-shape[dynamic_num][m][17])-1 > 29)
                return 1;

        erase_last_shape(dynamic_num,dynamic_mode,dynamic_x,dynamic_y);
        dynamic_mode = m;
        print_mode_shape(dynamic_num,dynamic_mode,dynamic_x,dynamic_y,dynamic_color);
        return 0;
}



int move_left(int n,int m)
{

        if(dynamic_x <= 12){
                return 1;
        }

        if(judge_shape(n,m,dynamic_x-2,dynamic_y))
                return 1;
        erase_last_shape(n,m,dynamic_x,dynamic_y);
        dynamic_x -= 2;
        print_mode_shape(n,m,dynamic_x,dynamic_y,dynamic_color);

        return 0;
}



int move_right(int n,int m)
{

        if(dynamic_x+2*(4-shape[n][m][16])-1 >= 39)
                return 1;
        if(judge_shape(n,m,dynamic_x+2,dynamic_y))
                return 1;


        erase_last_shape(n,m,dynamic_x,dynamic_y);
        dynamic_x += 2;
        print_mode_shape(n,m,dynamic_x,dynamic_y,dynamic_color);

        return 0;
}



 struct termios tm_old;


int getch()
{
        struct termios tm;

        tcgetattr(0,&tm_old);

        cfmakeraw(&tm);

        tcsetattr(0,0,&tm);

        int ch = getchar();

        tcsetattr(0,0,&tm_old);

        return ch;
}


void recover_attribute()
{

        tcsetattr(0,0,&tm_old);
}


void sig_handler(int signum)
{

        move_down(dynamic_num,dynamic_mode);

        if(judge_end_game() == 1)
        {
                game_over();

                recover_attribute();
                exit(0);
        }

}



#include "user_print.h"
#include <termios.h>

int next_num = 0;
int next_mode = 0;
int next_color = 0;


int init_x = 24;
int init_y = 6;


int next_x = 46;
int next_y = 8;
int dynamic_x = 0;
int dynamic_y = 0;


int dynamic_num = 0;
int dynamic_mode = 0;
int dynamic_color = 0;


int tm = 800000;


int score_x = 45;
int score_y = 18;
int level_x = 45;
int level_y = 22;


int matrix[24][28] = {0};

struct termios tm_old;


int shape[7][4][18] =
{
    {
            {1,1,0,0, 1,1,0,0, 0,0,0,0, 0,0,0,0, 2,2},
            {1,1,0,0, 1,1,0,0, 0,0,0,0, 0,0,0,0, 2,2},
            {1,1,0,0, 1,1,0,0, 0,0,0,0, 0,0,0,0, 2,2},
            {1,1,0,0, 1,1,0,0, 0,0,0,0, 0,0,0,0, 2,2},
    },
    {
            {1,0,0,0, 1,0,0,0, 1,0,0,0, 1,0,0,0, 3,0},
            {1,1,1,1, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,3},
            {1,0,0,0, 1,0,0,0, 1,0,0,0, 1,0,0,0, 3,0},
            {1,1,1,1, 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,3},
    },
    {
            {0,1,0,0, 1,1,1,0, 0,0,0,0, 0,0,0,0, 1,2},
            {1,0,0,0, 1,1,0,0, 1,0,0,0, 0,0,0,0, 2,1},
            {1,1,1,0, 0,1,0,0, 0,0,0,0, 0,0,0,0, 1,2},
            {0,1,0,0, 1,1,0,0, 0,1,0,0, 0,0,0,0, 2,1}
    },
    {
            {1,1,0,0, 0,1,1,0, 0,0,0,0, 0,0,0,0, 1,2},
            {0,1,0,0, 1,1,0,0, 1,0,0,0, 0,0,0,0, 2,1},
            {1,1,0,0, 0,1,1,0, 0,0,0,0, 0,0,0,0, 1,2},
            {0,1,0,0, 1,1,0,0, 1,0,0,0, 0,0,0,0, 2,1},
    },
    {
            {0,1,1,0, 1,1,0,0, 0,0,0,0, 0,0,0,0, 1,2},
            {1,0,0,0, 1,1,0,0, 0,1,0,0, 0,0,0,0, 2,1},
            {0,1,1,0, 1,1,0,0, 0,0,0,0, 0,0,0,0, 1,2},
            {1,0,0,0, 1,1,0,0, 0,1,0,0, 0,0,0,0, 2,1},
    },
    {
            {0,0,1,0, 1,1,1,0, 0,0,0,0, 0,0,0,0, 1,2},
            {1,0,0,0, 1,0,0,0, 1,1,0,0, 0,0,0,0, 2,1},
            {1,1,1,0, 1,0,0,0, 0,0,0,0, 0,0,0,0, 1,2},
            {1,1,0,0, 0,1,0,0, 0,1,0,0, 0,0,0,0, 2,1}
    },
    {
            {1,0,0,0, 1,1,1,0, 0,0,0,0, 0,0,0,0, 1,2},
            {1,1,0,0, 1,0,0,0, 1,0,0,0, 0,0,0,0, 2,1},
            {1,1,1,0, 0,0,1,0, 0,0,0,0, 0,0,0,0, 1,2},
            {0,1,0,0, 0,1,0,0, 1,1,0,0, 0,0,0,0, 2,1}},
};



#ifndef _USER_CONTROL_H_
#define _USER_CONTROL_H_

#include <stdio.h>
#include <termios.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <stdlib.h>


extern int getch();
extern void alarm_us(int n);

extern int dynamic_x;
extern int dynamic_y;
extern struct termios tm_old;


extern int dynamic_num;
extern int dynamic_mode;
extern int dynamic_color;
extern void key_control();
extern int change_shape();
extern int move_right(int n,int m);
extern int move_left(int n,int m);
extern void fall_down();
extern void game_over();
extern void recover_attribute();
#endif



#include "user_control.h"
#include "user_print.h"


int getch()
{
        struct termios tm;

        tcgetattr(0,&tm_old);

        cfmakeraw(&tm);

        tcsetattr(0,0,&tm);

        int ch = getchar();

        tcsetattr(0,0,&tm_old);

        return ch;
}


void recover_attribute()
{

        tcsetattr(0,0,&tm_old);
}


void alarm_us(int n)
{
        struct itimerval value;


        value.it_value.tv_sec = 0;
        value.it_value.tv_usec = n;


        value.it_interval.tv_sec = 0;
        value.it_interval.tv_usec = n;

        setitimer(ITIMER_REAL,&value,NULL);
}


void fall_down()
{
        int ret;
        while(1)
        {

                ret = move_down(dynamic_num,dynamic_mode);
                if(ret == 1)
                        return;
        }
}

int move_left(int n,int m)
{

        if(dynamic_x <= 12){
                return 1;
        }

        if(judge_shape(n,m,dynamic_x-2,dynamic_y))
                return 1;


        erase_last_shape(n,m,dynamic_x,dynamic_y);
        dynamic_x -= 2;
        print_mode_shape(n,m,dynamic_x,dynamic_y,dynamic_color);

        return 0;
}

int move_right(int n,int m)
{

        if(dynamic_x+2*(4-shape[n][m][16])-1 >= 39)
                return 1;
        if(judge_shape(n,m,dynamic_x+2,dynamic_y))
                return 1;


        erase_last_shape(n,m,dynamic_x,dynamic_y);
        dynamic_x += 2;
        print_mode_shape(n,m,dynamic_x,dynamic_y,dynamic_color);

        return 0;
}


int change_shape()
{
        int m = (dynamic_mode+1)%4;

        if(dynamic_x+2*(4-shape[dynamic_num][m][16])-1 > 39)
                return 1;

        if(dynamic_y+(4-shape[dynamic_num][m][17])-1 > 29)
                return 1;

        erase_last_shape(dynamic_num,dynamic_mode,dynamic_x,dynamic_y);
        dynamic_mode = m;
        print_mode_shape(dynamic_num,dynamic_mode,dynamic_x,dynamic_y,dynamic_color);
        return 0;
}

void game_over()
{
        printf("\33[32;9H**********  Game Over  ********\33[0m");

        printf("\33[?25h");


        printf("\n\n");
}

void key_control()
{
    int ch;
    while(1){
            ch = getch();
            if(ch == 'q' || ch == 'Q'){
                    break;
            }else if(ch == '\r'){

                    fall_down();
            }else if(ch == '\33'){
                    ch = getch();
                    if(ch == '['){
                            ch = getch();
                            switch(ch){
                            case 'A':
                                    change_shape();
                                    break;
                            case 'B':
                                    move_down(dynamic_num,dynamic_mode);
                                    break;
                            case 'C':
                                    move_right(dynamic_num,dynamic_mode);
                                    break;
                            case 'D':
                                    move_left(dynamic_num,dynamic_mode);
                                    break;
                            default:
                                    break;
                            }
                    }
            }
    }
    game_over();
    return ;
}
```

```
#ifndef _USER_PRINT_H_
#define _USER_PRINT_H_

extern int next_num;
extern int next_mode;
extern int next_color;

extern int next_num;
extern int next_mode;
extern int next_color;


extern int next_x;
extern int next_y;


extern int init_x;
extern int init_y;


extern  int dynamic_x;
extern  int dynamic_y;


extern  int dynamic_num;
extern  int dynamic_mode;
extern  int dynamic_color;


extern int shape[7][4][18];


extern int matrix[24][28];

extern void print_mode_shape(int n,int m,int x,int y,int c);
extern void print_next_shape();
extern void erase_last_shape(int n,int m,int a,int b);
extern int move_down(int num,int mode);
extern void store_current_shape();
extern void init_new_shape();
extern int judge_shape(int num,int mode,int x,int y);
#endif



#include <stdio.h>
#include <sys/time.h>
#include <stdlib.h>
#include <signal.h>
#include "user_print.h"






void print_mode_shape(int n,int m,int x,int y,int c)
{
    int i = 0;
    int xx = x;
    int yy = y;
    for(i = 0;i < 16;i++)
    {

        if(i != 0 && i%4 == 0)
        {
            yy += 1;
            xx = x;
        }

        if(shape[n][m][i] == 1){
            printf("\033[%d;%dH\033[%dm[]\033[0m",yy,xx,c);
        }
        xx += 2;
    }
    fflush(NULL);
}






void erase_last_shape(int n,int m,int a,int b)
{
    int i = 0;
    int xx = a;
    int yy = b;

    for(i = 0;i < 16;i++){
        if(i != 0 && i%4 == 0){
            yy++;
            xx = a;
        }
        if(shape[n][m][i] == 1){
            printf("\033[%d;%dH  \033[0m",yy,xx);
        }
        xx += 2;
    }
    fflush(NULL);
}


void print_next_shape()
{

    erase_last_shape(next_num,next_mode,next_x,next_y);

    next_num = random()%7;
    next_mode = random()%4;
    next_color = random()%7 + 40;


    print_mode_shape(next_num,next_mode,next_x,next_y,next_color);

    fflush(NULL);
}


void store_current_shape()
{
    int m_line = dynamic_y - 6;
    int m_column = dynamic_x - 12;
    int i = 0;

    for(i = 0;i < 16;i++)
    {

        if(i != 0 && i % 4 == 0)
        {
            m_line++;
            m_column = dynamic_x - 12;
        }


        if(shape[dynamic_num][dynamic_mode][i] == 1)
        {
            matrix[m_line][m_column] = dynamic_color;
            matrix[m_line][m_column + 1] = dynamic_color;
        }
        m_column += 2;

    }

}

void init_new_shape()
{
    dynamic_num = next_num;
    dynamic_mode = next_mode;
    dynamic_color = next_color;

    dynamic_x = init_x;
    dynamic_y = init_y;

    print_mode_shape(next_num,next_mode,dynamic_x,dynamic_y,dynamic_color);
}


int judge_shape(int num,int mode,int x,int y)
{
    int m_line = y - 6;
    int m_column = x - 12;

    int i = 0;

    for(;i < 16;i++){

        if(i != 0 && i%4 == 0)
        {
            m_line++;
            m_column = x-12;
        }
        if(shape[num][mode][i] == 1){
            if(matrix[m_line][m_column] != 0){
                return 1;
            }
        }
        m_column += 2;
    }
    return 0;
}

int move_down(int num,int mode)
{
    if((dynamic_y + (4 - shape[num][mode][17]) - 1 >= 29) || judge_shape(num,mode,dynamic_x,dynami
    {

        store_current_shape();


        init_new_shape();


        print_next_shape();

        return 1;
    }

    erase_last_shape(num,mode,dynamic_x,dynamic_y);
    dynamic_y++;
    print_mode_shape(num,mode,dynamic_x,dynamic_y,dynamic_color);

    return 0;
}





#include <stdio.h>
#include <termios.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <stdlib.h>
#include "user_print.h"
#include "user_control.h"


extern int tm;

extern int score_x;
extern int score_y;
extern int level_x;
extern int level_y;

void print_start_ui()
{

        printf("\33[2J");
        int i;

        for(i = 0;i < 47;i++){
                printf("\33[%d;%dH\33[43m \33[0m",5,i+10);
                printf("\33[%d;%dH\33[43m \33[0m",30,i+10);
        }

        for(i = 0;i < 26;i++){
                printf("\33[%d;%dH\33[43m  \33[0m",i+5,10);
                printf("\33[%d;%dH\33[43m  \33[0m",i+5,40);
                printf("\33[%d;%dH\33[43m  \33[0m",
                                i+5,56);
        }

        for(i=0;i < 17;i++){
                printf("\33[%d;%dH\33[43m \33[0m",12,40+i);
        }



        printf("\33[%d;%dH分数:\33[0m",score_y,score_x);

        printf("\33[%d;%dH等级:\33[0m",level_y,level_x);

        fflush(NULL);
}

void init_game_ui()
```

```
{

        print_start_ui();


        getch();


        srand(time(NULL));

        dynamic_num = random()%7;
        dynamic_mode = random()%4;
        dynamic_color = random()%7+40;

        dynamic_x = init_x;
        dynamic_y = init_y;


        print_mode_shape(dynamic_num,dynamic_mode,dynamic_x,dynamic_y,dynamic_color);

        print_next_shape();
        printf("\33[?25l");
}


int get_matrix_result(int n_line)
{
        int i = 0;

        if(n_line < 0)
        {
                return 1;
        }


        for(i = 0;i<28;i++)
        {
                if(matrix[n_line][i] != 0)
                {
                        return 1;
                }
        }

        return 0;
}


int judge_end_game()
{
        int n_line = 23;
        int n_count = 0;
        int i = 0;
        for(i = 0;i<24;i++)
        {
                int no_zero = get_matrix_result(n_line);
                if(no_zero != 0)
                {
                        n_line--;
                }
                else
                {
                        return 0;
                }
        }
        return 1;
}



void sig_handler(int signum)
{

        move_down(dynamic_num,dynamic_mode);

        if(judge_end_game() == 1)
        {
                game_over();

                recover_attribute();
                exit(0);
        }

}


int main()
{
        init_game_ui();


        signal(SIGALRM,sig_handler);


        alarm_us(tm);

        key_control();

        return 0;
}
```