湖南理工学院

Hunan Institute of Science and Technology

“炫通杯”大学生电子设计竞赛设计报告

题　　目：	B-2 俄罗斯方块游戏
院　　系：	信息科学与工程学院
专　　业：	软件工程
学生姓名：	陈琳雍 彭飞 张竣华

二〇二三年五月十一日

摘要

本程序类型为游戏，游戏名称为俄罗斯方块。本游戏通过键盘，结合方块移动旋转模块使方块移动、旋转，到达预期的位置。当方块完整拼出一条或几条时，通过消除模块对方块进行消除并获取相应得分。本游戏还可以选择双人对战模式，通过 socket 编程，在局域网中实现客户端和服务端之间实现 PK 对战，当一方结束游戏，分数高者赢得比赛，而双方的分数同时会录入排行榜中，以增加游戏的趣味性和竞争性。

关键词：游戏、方块移动旋转模块、消除模块、socket 编程

目录

一、前言

1.1 选题目的及意义

1.1.1 选题目的

为丰富我们学生的课外实践活动，培养我们的工程实践能力、创新能力和协作精神，我们参加了此次“炫通杯”电子设计竞赛。同时因为我们队伍都是软件工程专业学生，所以我们选择了 B 题设计题——俄罗斯方块游戏。这是使用 C 语言编程来实现的项目游戏，具有一定的难度。我们想勇于挑战自我，提高自我的编程水平。体验设计游戏的快乐。

1.1.2 意义

- 提高自我编程能力。更加熟练掌握 C 语言这门计算机程序设计语言。
- 体验游戏设计的乐趣。学习 C 语言已久，从未真正去写出过一个游戏，这次题目给我们很好的机会去展现自我。
- 丰富我们大学生的文化生活。让我们沉浸在编程的快乐中。

1.2 研究内容及要求

1.2.1 游戏任务

设计俄罗斯方块游戏程序。通过键盘移动需要的方块，然后通过

旋转、移动使得方块到达预期的位置。当在屏幕底部拼出完整的一条或几条时，这些完整的横条会随即消失，给新落下来的方块腾出空间，与此同时，玩家得到分数奖励。而没有被消除掉的方块将不断堆积起来， 一旦堆到屏幕顶端，则游戏结束。

1.2.2 游戏要求

1. 基本要求

1)操作功能

① 能正常进行方块移动、旋转、加速、消除。

② 能正常判断结束状态。

③ 游戏难度（即方块下落速度）多档可调。

2)显示功能

① 有一个完善的菜单系统。

② 能实时显示得分状态。

3)代码要求：良好的代码结构，具有一定可读性。

2. 发挥部分

(1) 具有双人在线同时玩功能（出来的方块顺序一致，最后 PK 出谁得分最高）。

(2) 使用图形库完成，有较美观的界面。

(3) 加入网络排行榜功能。

二、系统分析

2.1 游戏任务分析

首先明确需要实现单人模式的图形用户界面，在项目进行初期，我们成员内部就对此图像用户界面展开了激烈的讨论。目前主流的图形库有 BGI 图形库，GRX 图形库，dislin 库，allegro 图形库 ，opengl 和 sdl 库。

BGI 图形库：即 turbo c 所带的图形库，又叫 easyx 图形库，安装上手简单，功能强大适合 C 语言开发。其他图形库较为繁琐，且安装上手复杂。

总的来说，easyx 图形库是非常适合这次程序设计。因此我们选择了 easy-X 图像库进行用户界面设计和游戏的设计与实现。

“游戏需要随机生成方块”这个需求点，其中包括几个关键点--如何实现随机生成不同类型方块、如何实现预告下一方块是什么类型的方块的函数。

“通过旋转、移动使得方块到达预期的位置”这个需求点，我们结合任务参考资料分析，得到需要去实现的关键点：方块的旋转、移动可以通过不同按键实现不同功能，并且要确定不同类型的方块旋转下会有什么形态。

“当在屏幕底部拼出完整的一条或几条时，这些完整的横条会随即消失”需求点,这就涉及到方块的消除问题，涉及对横向方块的判断。

“与此同时，玩家得到分数奖励”这个需求点。这个需要在每次消除后，调用文本显示的函数，将分数更新显示。

“没有被消除掉的方块将不断堆积起来， 一旦堆到屏幕顶端，则游戏结束”这个需求点，我们需要考虑如何实现当方块堆积到屏幕顶端时游戏结束的函数。

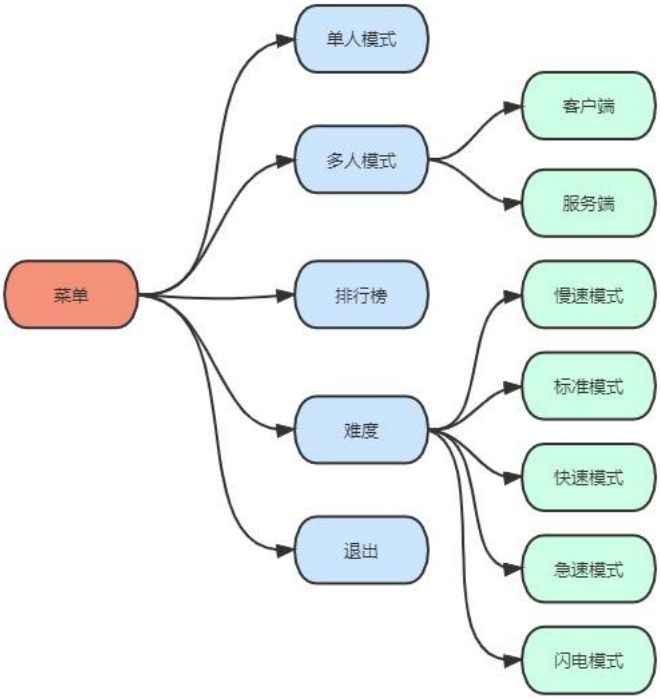
“双人对战根据双方最终得分高低得到优胜方”这个附加需求点，需要我们实现双人联机对战，同时实时传递双方的得分信息，在一方结束时退出游戏进程对比双方成绩判断优胜方并且将分数传入网络排行榜中。

2.2 软硬件环境

编程语言：C 语言

集成开发环境：Microsoft Visual Studio 2022

游戏界面结构图如下：



图一　游戏界面结构图

三、系统概要设计

3.1 内核设计思路

程序采用命令模式结构，将方块操作封装，由客户端（玩家方块下落线程）执行操作来控制方块走向，直到判定游戏失败，终止客户端一切操作

3.2 系统功能模块设计

在我们编写的程序中，主要有包括五大模块（随机生成、方块旋转、方块移动、双人联机、消除判断）。这几大模块基本囊括了所有的需求功能。

3.2.1 随机生成模块

程序先以 start 函数入口，传入一个特定的 seed 值，用于生成唯一的、随机的方块序列，以便于和可能的对手同步比赛。为了确保方块类型、颜色在同一 seed 的中具有整体唯一的、方块间完全随机的序列，采用线性加法算法来实现，同时也能预见性地给出未来下落方块，可方便给玩家预览。

算法如下，表示为第 $x+y$ 号方块的序号，A、B、C 均为常量。

$$f(x,y)=\left(\frac{x+y}{A}+B\right)\bmod C$$

3.2.2 方块旋转模块

方块有 I，J，L，S，Z，O，T 七种形态，故旋转时按照当前方块分发给不同函数执行，他们以 rotateXXXBlock 格式命名。

在现代俄罗斯方块中，不再考虑旋转时非占用空间，因此，该旋转也能在狭窄的空间里旋转自如——只要其满足旋转后不会与其他方块冲突。

设计思路是先记录好旋转后排列模式，需要时直接拷贝一份即可。

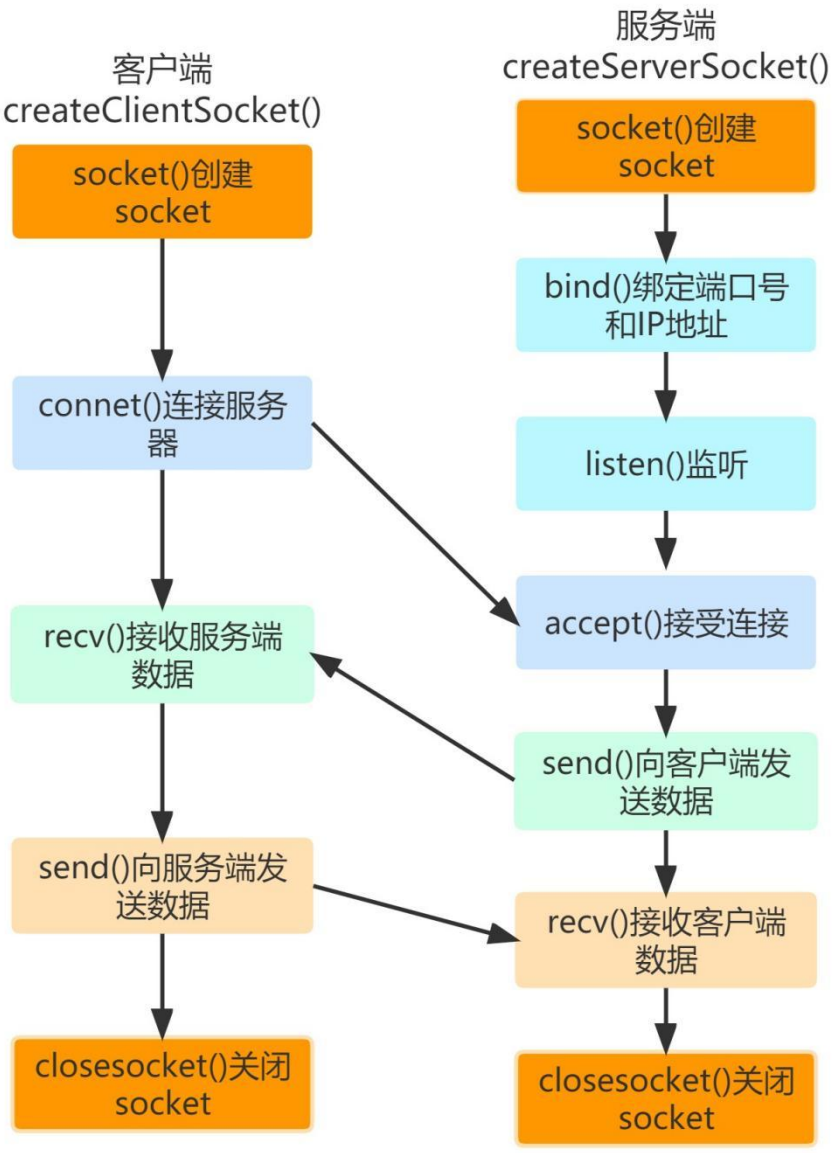
3.2.3 方块移动模块

作为请求者，其由 startMove()、endMove()和 bottomMove()对应了向左、向右以及向下移动。

如果判断能移动，则会对方块坐标进行赋值、修改，以实现目标。

该模块由 endMove()和 startMove()这个两个函数完成通过改变方块的 x 坐标数据实现方块的横线移动，而 bottomMove()则使方块加速下落

3.2.4 双人联机模块



图二 双人联机模块流程图

双人联机模块实现函数为 createClientSocket()和 createServerSocket()，利用 Socket 编程在 TCP 协议下来建立服务端和客户端实现两者之间信息互传。首先由服务端设定端口号以及 IP 地址，再由客户端通过设定的端口号以及 IP 地址，通过 acceprt()函数实现在局域网下与服务端的连接，当两者连接成功后，利用 send()和 recv()函数进行收发信息进而实现客户端与服务端之间信息互传，达到双人联机对战的要求。

3.2.5 消除判断模块

消除判断模块则是以 removeData()来判定。在方块完成下落并产生新方块前，由最近的四行判断是否有填充(isFilledRow(row))，有则消除(overWrite(row))，返回消除的行数，最后以平方后的分数来统计。

四、系统详细设计

4.1 数据存储的设计与描述

```
// save basic block data.
struct Block {
    // the width & height
    int property[2];
    // Take the lower left corner of the game as the origin.
    int data[4][2];
    // color
    enum BLOCK_TYPE type ;
};

const int Length = 7;
/* ... */
static struct Block mBlocks[Length] =
{
        {
            {1, 4},
            {0, 0, 0, 1, 0, 2, 0, 3},
            BLUE_BLOCK_TYPE
            },
        {
            {2, 3},
            {0, 0, 1, 0, 1, 1, 1, 2},
            BLUE_BLOCK_TYPE
```

图三 存储不同类型变量的结构体

结构体是 C 语言中很重要的一种存储结构，能够简化编程，封装变量。程序中共定义了一个结构体数组，用于存储方块的基本数据如方块宽、高、方块相对位置、颜色等信息。

五、系统实现

5.1 主要功能界面

这一部分我们对我们程序运行的界面和功能做一个简单的展示。包括

我们游戏的初始界面，方块生成界面，方块旋转变化界面、游戏得分显示界面、双人游戏界面、网络排行榜界面、游戏结束界面等等。由于方块移动、方块相除是一个动态过程，所以并没有用截图展示效果。

1.游戏开始界面

图四 游戏开始界面（game start screen）

2.初始生成随机方块界面

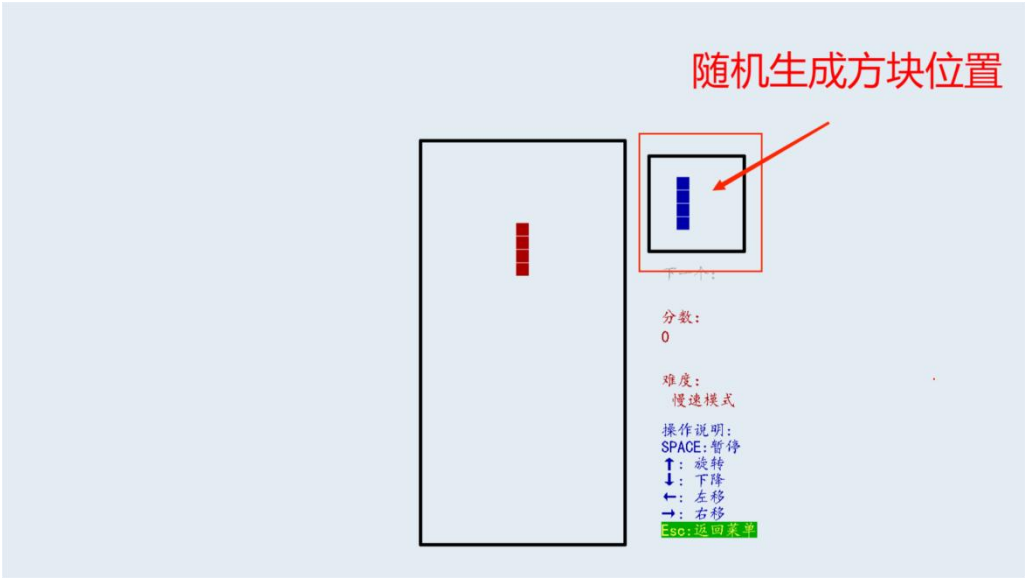


图 5 初始生成随机方块界面（Initial checkerboard block interface）

3.方块旋转界面

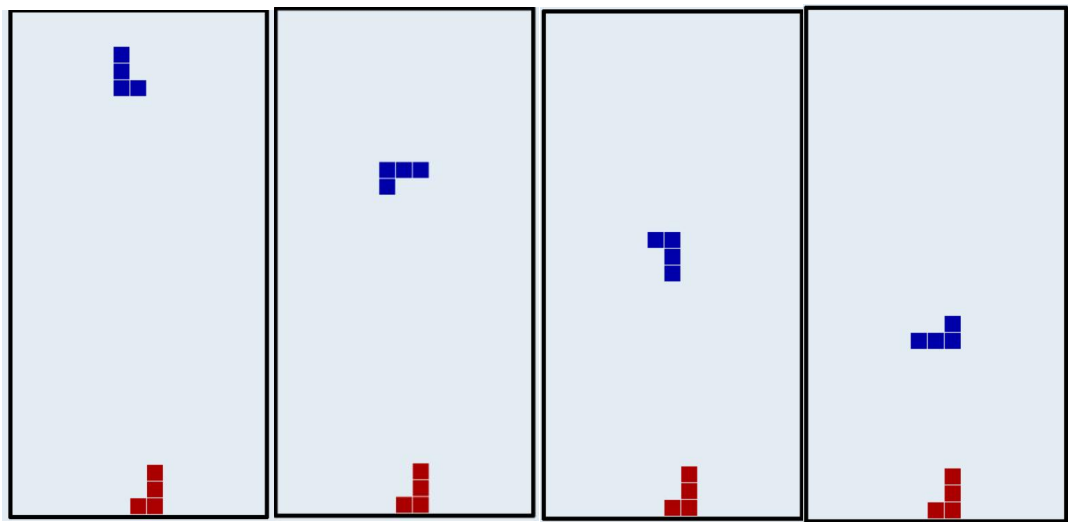


图 6 方块旋转（Block rotation）

4.游戏得分显示界面

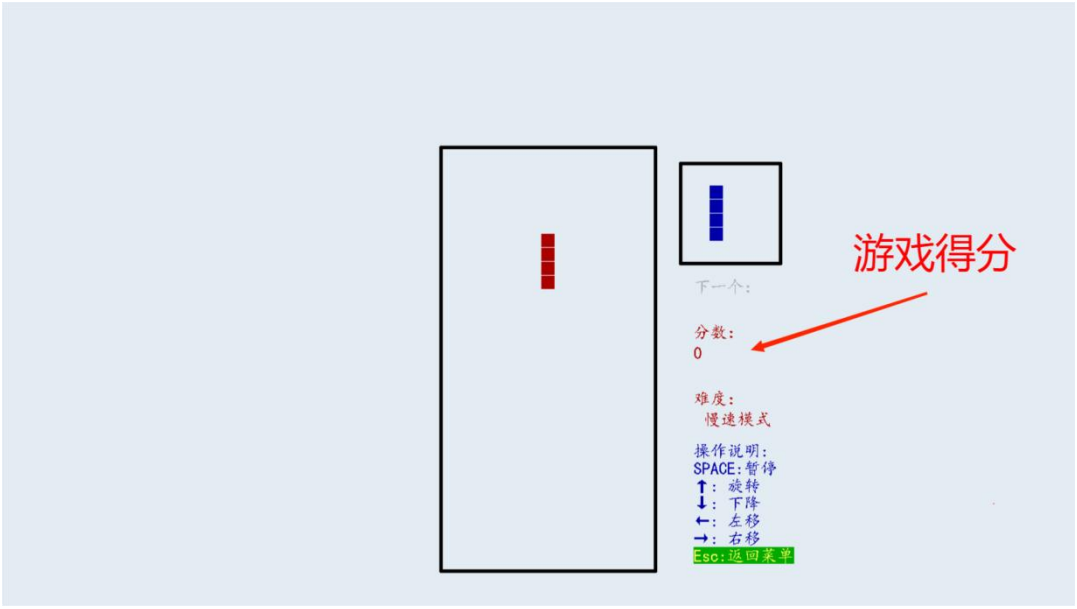


图 7 游戏得分（game score）

5.双人游戏对战界面

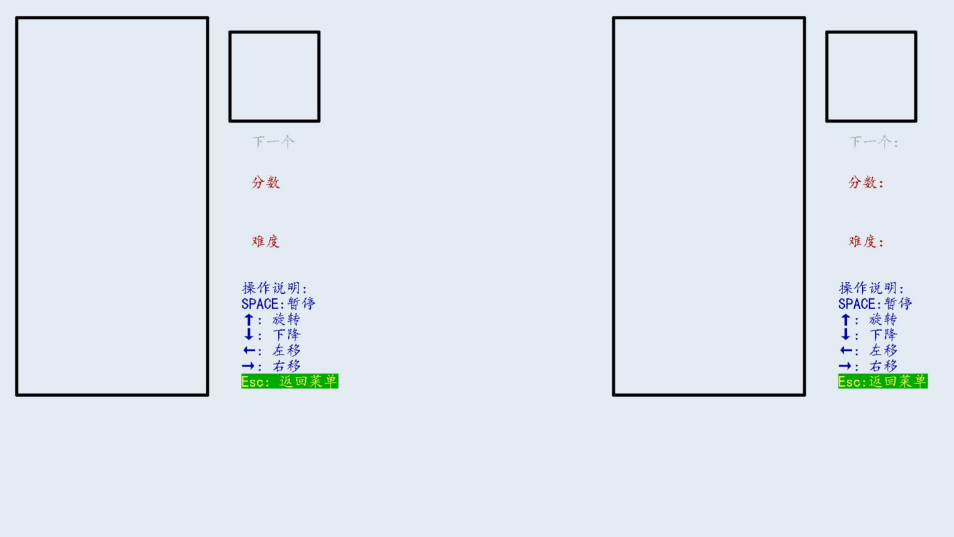


图 8 双人对战（Two-player battle）

6.网络排行榜界面

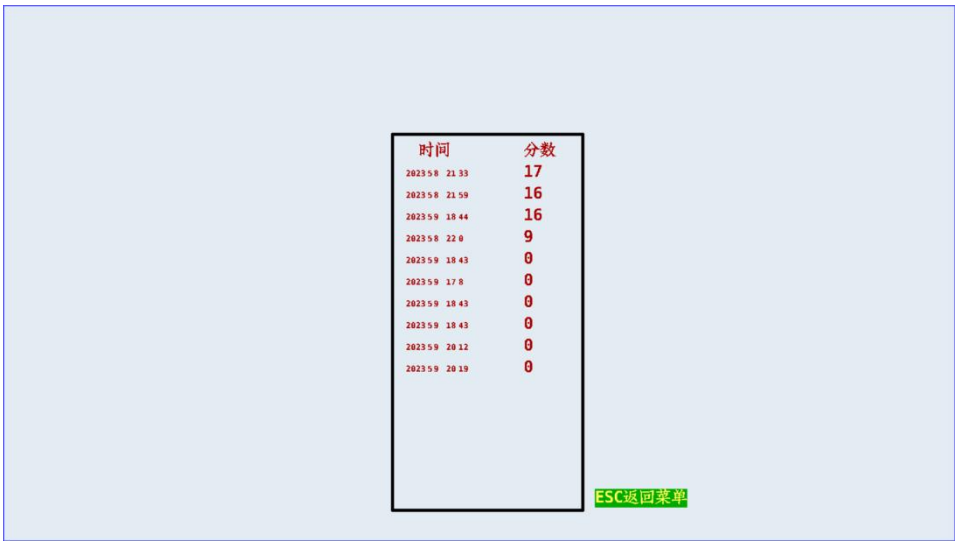


图 9 网络排行榜（Network ranking）

7.游戏结束界面

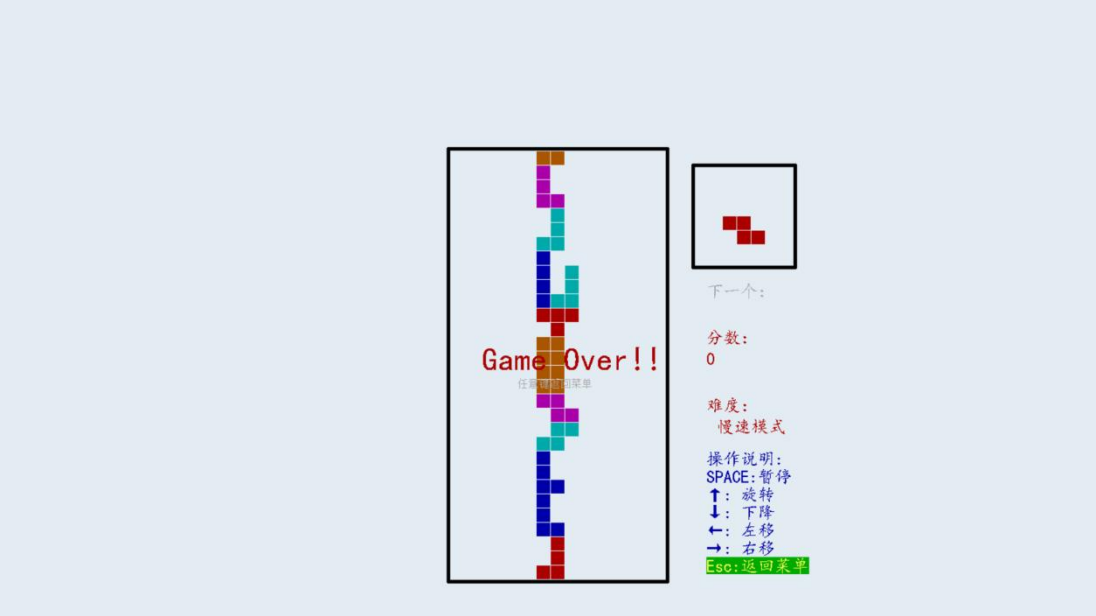


图 10 游戏结束（game over）

5.2 源代码

由于代码过长，我们将在最后附录部分给出所有代码。

六、收获及体会

6.1 收获

1.深入了解和学习了 easy-X 图像库以及 socket 编程。

2.对操作系统、网络编程有了更深一步的了解。

3.对 C 语言更加熟练掌握。

6.2 难点讨论

在我们对俄罗斯方块游戏进行设计与分析时，讨论过这些问题：

当利用 easyx 图形库绘制的连续运动的图片时每次贴完图片时都要进行清屏，这时就会出现闪屏现象，通过查询众多资料得知，可以通过双缓冲去解决，首先将要输出的数据写在缓冲区一（写的过程中显示的是缓冲区二的内容），之后显示缓冲区一的内容，而将要输出的数据则写在缓冲区二（写的过程中显示的是缓冲区一的内容）最后显示缓冲区二的内容 ，重复开始的步骤，这样就能够很好的解决运行中闪屏现象。

其次在实现双人对战过程中，由于玩家、系统控制方块操作有冲突的可能，使用两个线程同时对加互斥锁函数进行访问，避免了玩家一直按住导致方块不会自动下落的问题。

6.3 不足之处

对于 socket 编程掌握还不是很充足，本次双人对战只是由服务端和客户端进行的信息互传，而没有实现以服务器与多个客户端建立起的信息通信功能，进而导致双人对战功能不是很完备。

七、附录

Appendix 1

game.h

```
#ifndef _GAME_H_
#define _GAME_H_
#define _CRT_SECURE_NO_WARNINGS

#define WIN32_LEAN_AND_MEAN
#include<windows.h>
#include<time.h>
#include<stdlib.h>
#include<string.h>
#include<conio.h>


#include"initgreen1.h"
#include"initgreen2.h"
#include"paihang.h"
#include "degree.h"


#define BLOCK_COUNT 7
#define BLOCK_WIDTH 5
#define BLOCK_HEIGHT 5
#define UNIT_SIZE 20


#define R 960   //左右距离
#define SingleX 600
#define SingleY 180

#define ORANGE  0xFF8C00



//typedef unsigned int useconds_t;

; extern int speed; //方块降落速度
extern int degree;   //游戏难度

//用于读取文件
extern int rank ;
extern int score ;
extern int color[7];
//对应位置颜色
extern int OffsetX;         //偏移量
extern int OffsetY;
extern int score_game;


#endif
```

Appendix 2

Datainput.h

```
/*
 * Copyright (C) 2023 Frank Miles - Frms
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 */

/**
 *
 * @author Frms(Frank Miles)
 * @email FrankMiles@qq.com
 * @time 2023/04/23 下午 08:51
 */
#ifndef CLIONPROJECT_DATALEXER_H
#define CLIONPROJECT_DATALEXER_H

#include <memory>

extern void start(int value);
extern void suspend();
extern void reload();
extern void exit();
extern void setSpeed(int speed);

#endif //CLIONPROJECT_DATALEXER_H
```

```
/*
 * Copyright (C) 2023 Frank Miles - Frms
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
```

```
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 */

/**
 * 数据操作
 * @author Frms(Frank Miles)
 * @email FrankMiles@qq.com
 * @time 2023/04/23 下午 08:51
 */
#include "DataInput.h"
#include "R.h"
#include "DataOptions.h"
#include "GameDataLexer.h"
#include "../pf/activities/MainActivity.h"
#include <pthread.h>
#include <unistd.h>
#include <conio.h>
#include <synchapi.h>
#include "../pf/core/paihang.h"


typedef unsigned int useconds_t;

pthread_mutex_t lock;

pthread_t systemInput;

extern void input(enum INPUT_TYPE type);

void* systemRunnable(void* args);

useconds_t dropSpeed = 1000; // 1s
/**
 * 线程状态
 */
enum THREAD_STATE state = STOPPED;


void create(int value)
{
    newGameDataLexer(value);

    state = RUNNING;
    pthread_mutex_init(&lock, NULL);
    pthread_create(&systemInput, NULL, systemRunnable, NULL);

}

void* systemRunnable(void* args)
{
    while (true)
    {
        if(state == RUNNING) {
            input(BOTTOM);
            Sleep(dropSpeed);
        } else if(state == STOPPED) {
```

```
            return NULL;
        }
    }
    return NULL;
}

void input(enum INPUT_TYPE type)
{
    pthread_mutex_lock(&lock);
    submitOption(type);
    pthread_mutex_unlock(&lock);
}

void destory()
{
    exit();
    pthread_join(systemInput, NULL);
    pthread_mutex_destroy(&lock);
    stop();
}


bool  translateState = true;

/**
 * 创建操作数,
 * 传入当前方块序号，以及唯一值[0, 255]，会生成统一方块。
 * @param value 唯一值，用于生成同步块
 */
void start(int value)
{
    create(value);
    while (true)
    {
        char c = _getch();

        switch (c)
        {
            case 'a':
            case 'A':
                input(START);
                break;
            case 'd':
            case 'D':
                input(END);
                break;
            case 'w':
            case 'W':
                input(TOP);
                break;
            case 's':
            case 'S':
                input(BOTTOM);
                break;
            case 32:
                if(translateState) {
                    translateState= false;
                    suspend();
```

```
                }else
                {
                    translateState= true;
                    reload();
                }

                break;
            case 27:
                RandList(score_game);
                destory();
                onCreate();
                return;
        }
    }

}

/**
 * 阻塞线程，此时不会允许操作，如果对弈则无效。
 */
void suspend() {
    state = SUSPEND;
}

/**
 * 阻塞后可回复操作，如果对弈则无效。
 */
void reload() {
    state = RUNNING;
}

/**
 * 终止线程
 */
void exit() {
    state = STOPPED;
}

/**
 * 设置下落速度
 * @param speed
 */
void setSpeed(int speed) {
    dropSpeed = speed;
}
```

Appendix 3

DataOption.h

```
/*
 * Copyright (C) 2023 Frank Miles - Frms
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
```

```
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 */

/**
 *
 * @author Frms(Frank Miles)
 * @email FrankMiles@qq.com
 * @time 2023/04/23 下午 08:31
 */
#include "R.h"
#ifndef CLIONPROJECT_DATAOPTIONS_H
#define CLIONPROJECT_DATAOPTIONS_H

extern void init();

extern enum BLOCK_TYPE get(int x, int y);

extern enum SYSTEM_STATE set(int x, int y, enum BLOCK_TYPE val);

extern int checkRange(int x, int y);

extern void overWriteHeightData(int dest, int src);

extern void writeFullLine(int line, enum BLOCK_TYPE type);

extern void stop();
enum BLOCK_TYPE getSafely(int x, int y);

#endif //CLIONPROJECT_DATAOPTIONS_H
```

```
/*
 * Copyright (C) 2023 Frank Miles - Frms
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
```

```
 * limitations under the License.
 */

/**
 *
 * @author Frms(Frank Miles)
 * @email FrankMiles@qq.com
 * @time 2023/04/23 下午 08:31
 */
#include <memory.h>
//#include <ntdef.h>
#include "DataOptions.h"
#include "R.h"
#include "math.h"
#include "../pf/core/initgreen1.h"

enum BLOCK_TYPE array[HEIGHT + 1][WIDTH] = {};



void init() {

    memset(array, EMPTY_BLOCK_TYPE, sizeof(array));
   clearArray();
}

enum BLOCK_TYPE get(int x, int y)
{
    if(checkRange(x, y)) {
        return EMPTY_BLOCK_TYPE;
    }

    return array[y][x];
}

/**
 * get strict value , ignore the over range's data
 * @param x
 * @param y
 * @return
 */
enum BLOCK_TYPE getSafely(int x, int y)
{
    if(checkRange(x, y)) {
```

```
        return OVER_RANGE_BLOCK_TYPE;
    }

    return array[y][x];
}


enum SYSTEM_STATE set(int x, int y, enum BLOCK_TYPE val)
{
    if(checkRange(x, y)) {
//      errorPrint("DataOptions Out of Array Index 2");
        return ERROR_SYS_STA;
    }

    array[y][x] = val;

    return RIGHT_SYS_STA;
}

/**
 * 将src内容覆盖到dest中去
 * @param dest
 * @param src
 */
void overWriteHeightData(int dest, int src)
{
    if(fmin(dest, src) < 0 || fmax(dest, src) > HEIGHT) {
        return;
    }
    memcpy(array[dest], array[src], sizeof(array[dest]));
}


void writeFullLine(int line, enum BLOCK_TYPE type)
{
    if(line < 0 || line > HEIGHT) {
        return;
    }
    memset(array[line], type, sizeof(array[line]));
}


int checkRange(int x, int y) {
    return x < 0 || x >= WIDTH || y < 0 || y >= HEIGHT;
```

```
}
```

Appendix 4

GameDataLexer.cpp

```
/*
 * Copyright (C) 2023 Frank Miles - Frms
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 */

/**
 * 游戏数据处理（线程不安全）
 * 数组坐标轴:
 * 原点位于左下角,
 * @author Frms(Frank Miles)
 * @email FrankMiles@qq.com
 * @time 2023/04/26 下午 11:16
 */
#include <cstdlib>
//#include <ntdef.h>
#include <cmath>
#include "GameDataLexer.h"
#include "DataOptions.h"
#include "../pf/core/game.h"

void startMove();

bool inBlockRange(int x, int y);

void endMove();

void bottomMove();

int removeData(int y, int height);

bool isFilledRow(int y);

void topMove();
```

```
void rotateIBlock();

void rotateJBlock();

void rotateLBlock();

void rotateSBlock();

bool canRotateBlock();

void copyNativeBlockData();

void rotateZBlock();

void rotateTBlock();

void overWrite(int y);

bool isAnyInRow(int y);

void redraw();

struct Block nowBlock {};

/**
 * 下落方块位置，指的是方块相对坐标在绝对坐标位置
 */
int blockX = 0, blockY = 0;

/**
 * 方块类型，I, J, L, S, Z, O, T,
 * 0, 1, ...
 */
int blockType = 0;

/**
 * 方块状态
 */
int blockState = 0;

/**
 * 方块计数器
 */
int blockCounter = 0;
/**
 * 运行状态
 */
bool isRunning = false;

int singleValue = 0;

/**
 * 新建一个类用于实现
 * @param value
 */
void newGameDataLexer(int value)
{
    blockCounter = 0;
```

```
    singleValue  = value;
    init();

    setBlock();
    isRunning = true;
}

void stop()
{
    blockCounter = 0;
    singleValue = 0;
    isRunning = false;
    // it is also mean clear every data.
    init();
    blockX = 0;
    blockY = 0;
    memset(nowBlock.data, 0, sizeof(nowBlock.data));
    memset(nowBlock.property, 0, sizeof(nowBlock.property));
    nowBlock.type = EMPTY_BLOCK_TYPE;
}

/**
 * 生成1-5颜色序号
 * @param count 如果想要下x个，置x
 * @return
 */
int getBlockColor(int count = 0) {
    int res =  1 +(abs(((blockCounter + count) * (singleValue + 2) +
514)/4) % 5);
    return res;
}

/**
 * blocks type
 * I, J, L, S, Z, O, T
 * 0, 1, 2, 3, 4, 5, 6
 * 生成0-6方块序号
 * @param count 如果想要下x个，置x
 * @return
 */
int getBlockType(int count = 0){
    int res = (abs(((blockCounter + count) * (singleValue + 2) +
114)/4) % 7);
    return  res;// res;
}

/**
 * 设置block，顺便会重置位置
 * @return
 */
void setBlock()
{
    blockState = 0;
    blockCounter++;

    nowBlock.type = (BLOCK_TYPE)(getBlockColor());
    blockX = (WIDTH - nowBlock.property[0]) >> 1;
    blockY = HEIGHT;
```

```
    blockType = getBlockType();

    copyNativeBlockData();
}

/**
 * 提交的操作
 * @param type
 */
void submitOption(enum INPUT_TYPE type)
{
    if(!isRunning) {
        return;
    }
    switch (type)
    {
        case START:
            startMove();
            break;
        case END:
            endMove();
            break;
        case TOP:
            topMove();
            break;
        case BOTTOM:
            bottomMove();
            break;
    }
}

/**
 * 旋转方块，只有在能旋转时才有效。
 */
void topMove()
{
    switch(blockType)
    {
        case 0:
            rotateIBlock();
            break;
        case 1:
            rotateJBlock();
            break;
        case 2:
            rotateLBlock();
            break;
        case 3:
            rotateSBlock();
            break;
        case 4:
            rotateZBlock();
            break;
        case 6:
            rotateTBlock();
            break;
    }
}
```

```
void rotateTBlock()
{
   if(blockState == 3)
   {
      if(!canRotateBlock()) {
         return;
      }

      blockState = 0;
      copyNativeBlockData();

   } else
   {

      for(int i=1; i<=4; i++)
      {
         if(
               getSafely(
                     blockX + TBlock[blockState][i][0],
                     blockY + TBlock[blockState][i][1]
               ) != EMPTY_BLOCK_TYPE
         ) {
            return;
         }
      }
      memcpy(nowBlock.property,  TBlock[blockState][0],
sizeof(nowBlock.property));
      memcpy(nowBlock.data, TBlock[blockState][1],
sizeof(nowBlock.data));
      blockState++;

   }


   redraw(blockX, blockX, blockX+4, blockY+4);
}

void rotateZBlock()
{
   if(blockState == 0)
   {
      for(int i=1; i<=4; i++) {
         if(getSafely(ZBlock[i][0]+blockX,
ZBlock[i][1]+blockY) != EMPTY_BLOCK_TYPE) {
            return;
         }
      }
      memcpy(nowBlock.property, ZBlock[0],
sizeof(nowBlock.property));
      memcpy(nowBlock.data, ZBlock[1], sizeof(nowBlock.data));

      blockState = 1;
   } else
   {
      if(!canRotateBlock()) {
         return;
      }
```

```
       copyNativeBlockData();
       blockState = 0;
    }
    redraw(blockX, blockX, blockX+4, blockY+4);
}

void rotateSBlock()
{
    if(blockState == 0)
    {
       for(int i=1; i<=4; i++) {
          if(getSafely(SBlock[i][0]+blockX,
SBlock[i][1]+blockY) != EMPTY_BLOCK_TYPE) {
             return;
          }
       }
       memcpy(nowBlock.property, SBlock[0],
sizeof(nowBlock.property));
       memcpy(nowBlock.data, SBlock[1], sizeof(nowBlock.data));

       blockState = 1;
    } else
    {
       if(!canRotateBlock()) {
          return;
       }
       copyNativeBlockData();
       blockState = 0;
    }
    redraw(blockX, blockX, blockX+4, blockY+4);
}

void rotateLBlock()
{
    if(blockState == 3)
    {
       if(!canRotateBlock()) {
          return;
       }

       copyNativeBlockData();

    } else
    {
       for(int i=1; i<=4; i++)
       {
          if(
                getSafely(
                      blockX + LBlock[blockState][i][0],
                      blockY + LBlock[blockState][i][1]
                ) != EMPTY_BLOCK_TYPE
          ) {
             return;
          }
       }
       memcpy(nowBlock.property,  LBlock[blockState][0],
sizeof(nowBlock.property));
       memcpy(nowBlock.data, LBlock[blockState][1],
```

```
sizeof(nowBlock.data));

    }

    blockState = (blockState + 1) % 4;
    redraw(blockX, blockX, blockX+4, blockY+4);
}

void copyNativeBlockData()
{
    memcpy(nowBlock.property, mBlocks[blockType].property,
sizeof(nowBlock.property));
    memcpy(nowBlock.data, mBlocks[blockType].data,
sizeof(nowBlock.data));
}

void rotateJBlock()
{
    if(blockState == 3)
    {
        if(!canRotateBlock()) {
            return;
        }

        copyNativeBlockData();
    } else
    {
        for(int i=1; i<=4; i++)
        {
            if(
                getSafely(
                        blockX + JBlock[blockState][i][0],
                        blockY + JBlock[blockState][i][1]
                ) != EMPTY_BLOCK_TYPE
            ) {
                return;
            }
        }
        memcpy(nowBlock.property,  JBlock[blockState][0],
sizeof(nowBlock.property));
        memcpy(nowBlock.data, JBlock[blockState][1],
sizeof(nowBlock.data));

    }

    blockState = (blockState + 1) % 4;
    redraw(blockX, blockX, blockX+4, blockY+4);
}

void rotateIBlock()
{
    if(blockState == 0)
    {
        int x = blockX - 2;
        int y = blockY + 2;

        // some time block in top , so do not judge y.
        if(x < 0 || blockX + 1 >= WIDTH) {
```

```
            return;
        }

        for(int i=1; i<=4; i++) {
            if(get(x + IBlock[i][0], y +IBlock[i][1]) !=
EMPTY_BLOCK_TYPE) {
                return;
            }
        }
        memcpy(nowBlock.property, IBlock[0],
sizeof(nowBlock.property));
        // todo : It's really work? from a java programmer.
        memcpy(nowBlock.data, IBlock[1], sizeof(nowBlock.data));
        blockX = x;
        blockY = y;
        blockState = 1;
    } else {
        int x = blockX + 2;
        int y = blockY - 2;

        if(x >= WIDTH || y < 0) {
            return;
        }

        for(int i=0; i<4; i++)
        {
            if(get(mBlocks[blockType].data[i][0] + x,
mBlocks[blockType].data[i][1] + y) != EMPTY_BLOCK_TYPE) {
                return;
            }
        }
        copyNativeBlockData();
        blockX = x;
        blockY = y;
        blockState = 0;
    }

    redraw(blockX, blockX, blockX+4, blockY+4);
}

/**
 * 是否能旋转方块-只能用作初始方块检测
 * @return
 */
bool canRotateBlock()
{
    for(int i=0; i<4; i++)
    {
        if(
                getSafely(
                        blockX + mBlocks[blockType].data[i][0],
                        blockY + mBlocks[blockType].data[i][1]
                ) != EMPTY_BLOCK_TYPE
                ) {
            return false;
        }
    }
    return true;
```

```
}

/**
 * bottom 会固定方块，发生写操作。
 */
void bottomMove()
{
    /*
     * 检查游戏是否失败
     * - 只检查HEIGHT + 1行 是否有方块，有则失败（仅在生成新的方块之前检测有效）。
     */

    if(isAnyInRow(HEIGHT-1)) {
        stateTrigger(-99);
        return;
    }


    // force trigger func
    bool canDown = (blockY >= 1);

    for(int i=0; i<4 && canDown; i++)
    {
        int y = (int)fmin(blockY - 1 + nowBlock.data[i][1], HEIGHT - 1);
        if( getSafely(blockX + nowBlock.data[i][0], y) != EMPTY_BLOCK_TYPE) {
            canDown = false;
            break;
        }
    }
    // no reach
    if(canDown)
    {
        blockY--;
        redraw(blockX, blockY, blockX+1+nowBlock.property[0], blockY + nowBlock.property[1] + 1);
    } else
    {
        // reach
        for(int i=0; i<4; i++) {
            set(blockX + nowBlock.data[i][0], blockY + nowBlock.data[i][1], nowBlock.type);
        }

        // check can remove:
        int power = removeData(blockY, nowBlock.property[1]);
        stateTrigger(power);

        setBlock();
        redraw();
    }

}

/**
 * 某行是否存在一个方块
```

```
 * @param y
 * @return
 */
bool isAnyInRow(int y)
{
    for(int i=0; i<WIDTH; i++) {
        if(get(i, y) != EMPTY_BLOCK_TYPE) {
            return true;
        }
    }
    return false;
}


/**
 * 整块消除
 * @param y
 * @param height
 * @return
 */
int removeData(int y, int height)
{
    int power = 0;
    for(int i=0; i<4; i++)
    {
        if(isFilledRow(y)) {
            overWrite(y);
            power++;
        }
    }

    return power;
}

/**
 * 将y限制出去
 * @param y
 */
void overWrite(int y)
{
    for(int i=y+1; i<=HEIGHT; i++) {
        overWriteHeightData(i-1, i);
    }
    writeFullLine(HEIGHT, EMPTY_BLOCK_TYPE);
}

/**
 * 是否被全部填充
 * @param y
 * @return
 */
bool isFilledRow(int y)
{
    for(int i=0; i<WIDTH; i++) {
        if(getSafely(i, y) == EMPTY_BLOCK_TYPE) {
            return false;
        }
    }
```

```
    return true;
}


/**
 * 向end方向平移
 */
void endMove()
{
    if(blockX+nowBlock.property[0] > WIDTH) {
        return;
    }

    int y;
    for(int i=0; i<4; i++)
    {
        y = blockY + nowBlock.data[i][1];
        // blocked condition
        if( y < HEIGHT && getSafely(blockX + nowBlock.data[i][0] + 1,
y) != EMPTY_BLOCK_TYPE) {
            return;
        }
    }

    blockX++;

    redraw(blockX-1, blockY, blockX+nowBlock.property[0], blockY +
nowBlock.property[1]);
}

/**
 * 向start方向平移
 */
void startMove()
{
    if(blockX == 0) {
        return;
    }

    int y;
    for(int i=0; i<4; i++)
    {
        y = blockY + nowBlock.data[i][1];
        // blocked condition
        if( y<HEIGHT && getSafely(blockX-1 + nowBlock.data[i][0],
y) != EMPTY_BLOCK_TYPE) {
            return;
        }
    }

    blockX--;

    redraw(blockX, blockY, blockX+5, blockY + nowBlock.property[1] +
1);
}


/**
```

```
 * 获取某位置坐标
 * @param x
 * @param y
 * @return
 */
enum BLOCK_TYPE getBlock(int x, int y)
{
    if(checkRange(x, y)) {
        errorPrint("You can't get the position's info.");
        return EMPTY_BLOCK_TYPE;
    }

    if(inBlockRange(x, y)) {
        return nowBlock.type;
    }

    return get(x, y);
}

/**
 * 检查坐是否在范围内。
 * @param x
 * @param y
 * @return
 */
bool inBlockRange(int x, int y)
{
    if(
            x >= blockX      && x <= blockX + 4//nowBlock.property[0]
            && y >= blockY   && y <= blockY + 4//nowBlock.property[1]
    ) {
        for(int i=0; i<4; i++)
        {
            if(x == blockX+nowBlock.data[i][0] && y ==
blockY+nowBlock.data[i][1]) {
                return true;
            }
        }
    }
    return false;
}

/**
 * 显示当前状态
 * @param score 得分，小于0表示结束
 */
void stateTrigger(int pScore)
{
    ScoreTrigger(pScore);
    isRunning = (pScore >= 0);
    if(!isRunning) {
        stop();
    }
    redraw();
}


/**
```

```
 * 全局重绘指令
 */
void redraw() {
    redraw(0, 0, WIDTH-1, HEIGHT-1);
}

/**
 * 区域重绘
 * @param startX
 * @param startY
 * @param endX
 * @param endY
 */
void redraw(int startX, int startY, int endX, int endY)
{
    onDraw(startX, startY, endX, endY);
}
```

Appendix 6

initgreen1.h

```
#ifndef _INITGREEN1_H_
#define _INITGREEN1_H_
#define _CRT_SECURE_NO_WARNINGS

#include "game.h"

#include<graphics.h>
#include<easyx.h>




extern void initGameScreen1();
extern void clearArray();
extern void ScoreTrigger(int pScore);
extern void failCheck();
#endif
```

```
#include "frms/R.h"
#include <graphics.h>
#include "pf/core/degree.h"
#include "frms/GameDataLexer.h"
#include "pf/activities/MainActivity.h"




enum BLOCK_TYPE drawArray[WIDTH][HEIGHT] = {};
```

```
/**************************
 * 功能：初始化单人游戏场景
 * 输入：
 *      无
 * 返回：
 *      无
 **************************/
void initGameScreen1() {
    char str[16];  //存放分数

       //1.清屏
    setbkcolor(RGB(228, 236, 243));
    // 用背景色清空屏幕

    cleardevice();

    //2.画场景
//  setbkcolor(BLACK);
    setlinecolor(BLACK);
    rectangle(627, 207, 936, 815);    //方块降落框外框
    rectangle(629, 209, 934, 813);    //方块降落框内框
    rectangle(970, 230, 1115, 375);   //方块提示外框
    rectangle(972, 232, 1113, 373);   //方块提示内框

    setbkmode(TRANSPARENT);
    setfont(24, 0, "楷体");    //写“下一个”
    settextcolor(LIGHTGRAY);  //灰色
    outtextxy(990, 395, "下一个：");

    settextcolor(RED);                //写分数
    outtextxy(990, 460, "分数：");


    //按指定格式，将 score 写入 str
//  sprintf_s(str, 16, "%d", score);

    //这里设置字符集为多字符，保证 outtextxy 可以写出变量 str
//  outtextxy(955, 490, str);

    outtextxy(990, 555, "难度："); //难度
    degreePrint();
```

```
    settextcolor(BLUE);    //操作说明
    outtextxy(990, 630, "操作说明:");
    outtextxy(990, 655, "SPACE:暂停");
    outtextxy(990, 680, "↑: 旋转");
    outtextxy(990, 705, "↓: 下降");
    outtextxy(990, 730, "←: 左移");
    outtextxy(990, 755, "→: 右移");
    settextcolor(YELLOW);    // 设置颜色
    setbkcolor(GREEN);
    setbkmode(OPAQUE);
    outtextxy(990, 780, "Esc:返回菜单");


    onDraw(0, 0, WIDTH - 1, HEIGHT - 1);

}



void ScoreTrigger(int pScore)
{
    if (pScore < 0) {
        RandList(score_game);
        failCheck();
        exit(123165);
    }
    else {
        score_game += pScore * pScore;
    }
}
char str[16];

/**
 * ������Ç©��
 * include these params.
 * @param startX
 * @param startY
 * @param endX
 * @param endY
 */
void onDraw(int startX, int startY, int endX, int endY)
{
    setbkmode(TRANSPARENT);
```

```
    setfillcolor(RGB(228, 236, 243));
    solidrectangle(390 + OffsetX, 310 + OffsetY, 450 + OffsetX, 340 + OffsetY);
    setfont(24, 0, "楷体");
    settextcolor(RED);                //д����
    sprintf_s(str, 16, "%d", score_game);
    outtextxy(390 + OffsetX, 310 + OffsetY, str);


    BeginBatchDraw();
    setfont(23, 0, "楷体");
    setbkmode(TRANSPARENT);
    setfillcolor(RGB(228, 236, 243));
    solidrectangle(375 + OffsetX, 54 + OffsetY, 510 + OffsetX, 185 + OffsetY);

    for (int i = 0; i < WIDTH; i++) {
        for (int l = 0; l < HEIGHT; l++) {
            drawArray[i][l] = getBlock(i, l);
            // do this....
            int x2 = 30 + OffsetX + i * UNIT_SIZE;
            int y2 = 610 + OffsetY - l * UNIT_SIZE;
            if (drawArray[i][l]) {
                setcolor(color[drawArray[i][l]]);
                outtextxy(x2, y2, "■");
            }
            else {
                setcolor(RGB(228, 236, 243));
                outtextxy(x2, y2, "■");
            }
        }
    }

    int block[4][2] = {};
    memcpy(block, mBlocks[getBlockType(1)].data, sizeof(block));

    for (int i = 0; i < 4; i++) {
        int x2 = 411 + OffsetX + block[i][0] * UNIT_SIZE;
        int y2 = 141 + OffsetY - block[i][1] * UNIT_SIZE;
        setcolor(color[getBlockColor(1)]);
        outtextxy(x2, y2, "■");
    }

    EndBatchDraw();

}
```

```
void clearArray()
{
    memset(drawArray, EMPTY_BLOCK_TYPE, sizeof(drawArray));

}



void failCheck() {


    setcolor(RED);
    settextstyle(45, 0, "楷体");

    outtextxy(75 + 600, 300 + 180, "Game Over!!");

    setcolor(LIGHTGRAY);
    settextstyle(15, 0, "隶体");
    outtextxy(125 + 600, 350 + 180, "任意键返回菜单");
    Sleep(1000);
    while (!_kbhit()) {

    }

    onCreate();

}
```