课 程 设 计 报 告 2

课程名称 计算机程序设计基础 2

班　　级 无六五

学　　号 2016011109

姓　　名 王春禹

指导助教 无

2017 年　7 月 5 日

游戏名称：俄罗斯方块修改版

目　录

1.系统需求分析

该程序原型是经典的俄罗斯方块小游戏，不同在于增加了将方块变为炸弹的功能，同时为了降低难度，增加了随时切换方块的功能。。目的在于使玩家放松身心的同时，提高玩家的应变能力。玩法是将不同形状的方块下放至合适的位置，某一行充满方块则会消去。该游戏采取积分制，随分数增加，关卡数增加，游戏难度也会上升，即速度加快。共计 8 关，每关 20 分，通关则胜利。

操作需求：

方向键：按“下”会加速方块下落，按“左”“右”方块会左右移动，按“上”使方块

变形。当方块遇到边界或其他方块则不能移动。按“space”键方块直接下落。按”p”游戏暂停，按任意键恢复。随时按“c”可更换方块。按“b”可将当前方块变为炸弹，降落后与其接触的方块会消失，但每用一次分数会减少 5 分

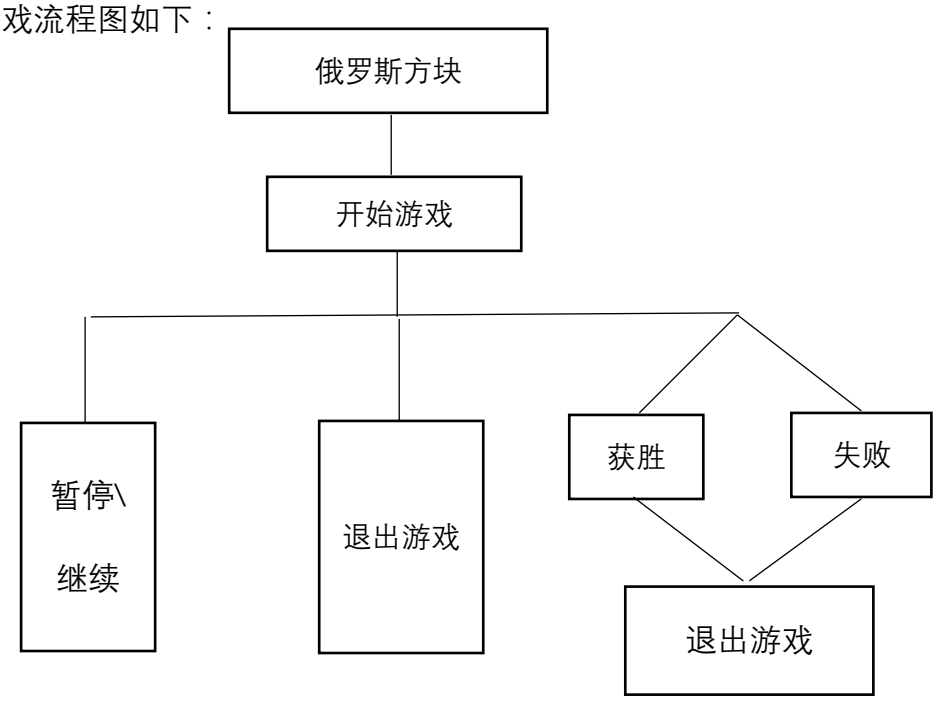
方法需求：

建立 Point 类，表示输出屏幕上的点，派生出 Block 类，私有成员包括 kind（种类）、dir（方向）。方块种类、方向用二维数组 allstate 储存。Block 对象只有两个，一个是当前运动的块，一个是下一个，共有七种状态，每个状态对应一种方块形状，同时每个状态都有四个方向，所以是一个 7x4 的数组。每个方块占有一个 4x4 的面积（用数组表示），用 square 和 space 来填充构成方块形状，且每个方块都用它左上角点的坐标来表示位置。改变方块的方向只要改变 dir，每转四次就会回到原状态。Game 类用来控制整个游戏进程。

2.总体设计

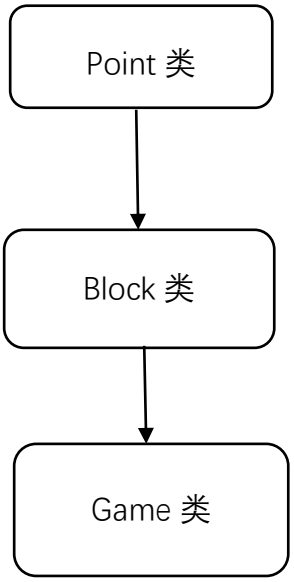
该程序包含游戏界面和信息界面，游戏界面是主面板，显示方块。进入游戏即可开始，直到游戏结束，中途无其他出口。信息界面是副面板，显示 score 和 level，以及下一个出现的方块和操作提示。

游戏流程图如下：

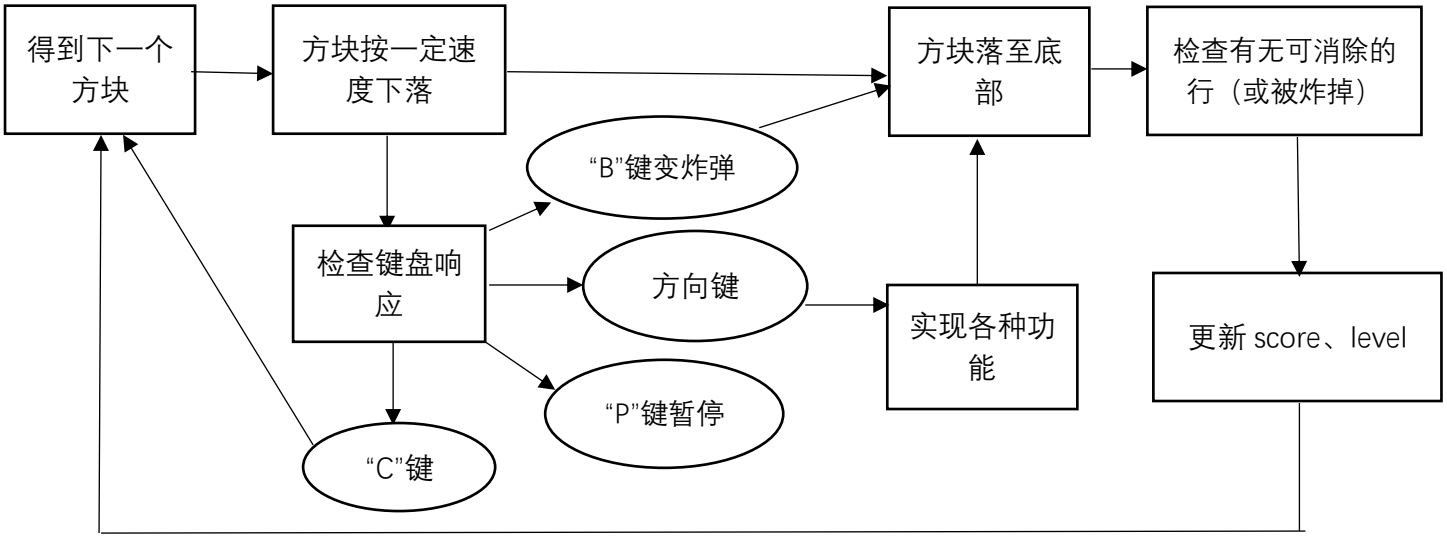


3.详细设计

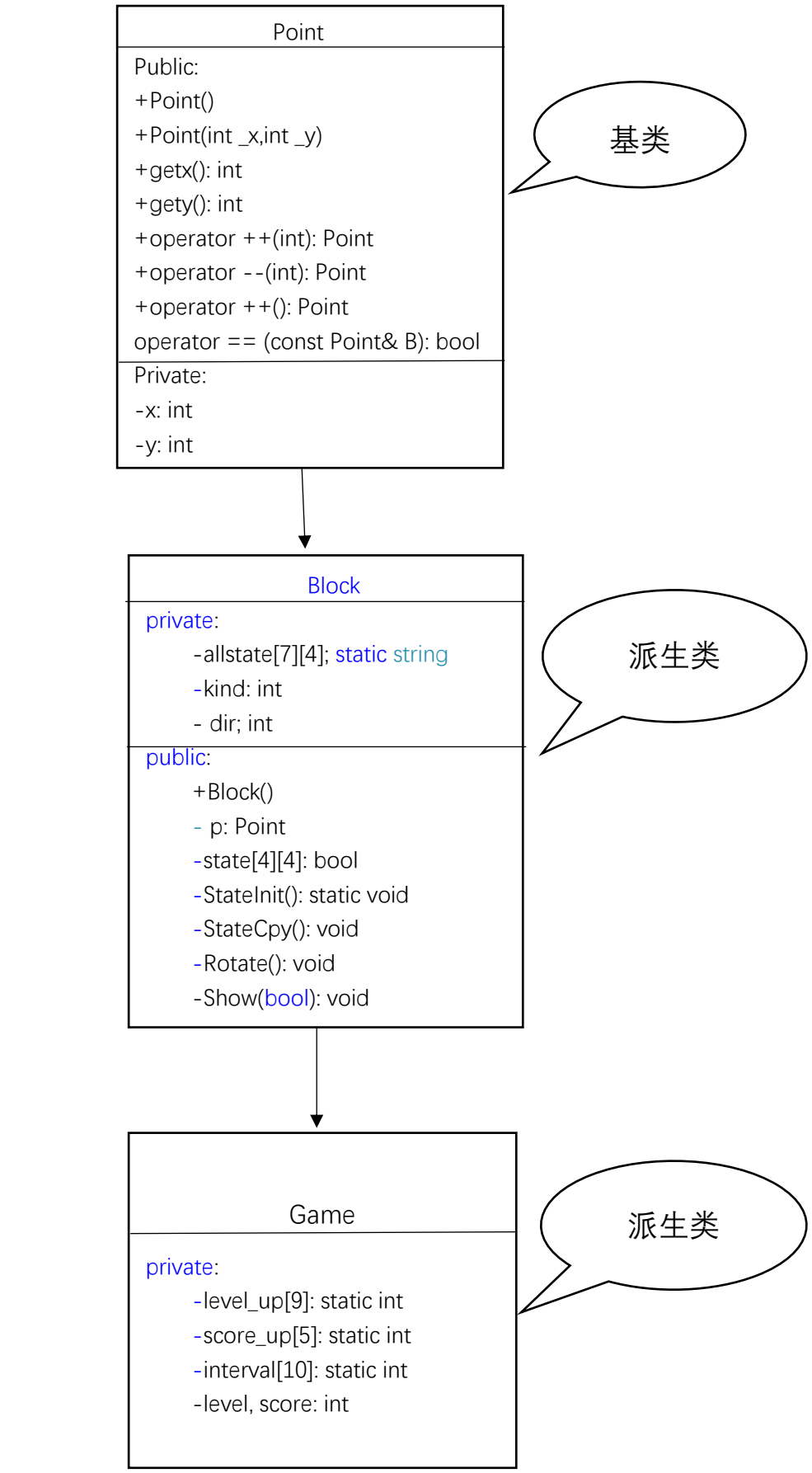
俄罗斯方块的三个类的类层次图为：



功能模块图：



三个类的 UML 图：

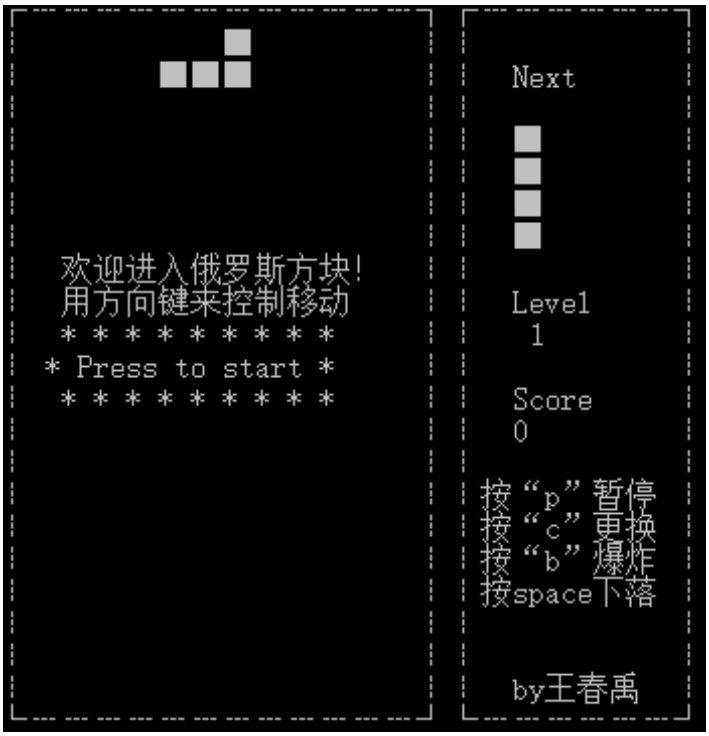

Point
Public:
+Point()
+Point(int _x,int _y)
+getx(): int
+gety(): int
+operator ++(int): Point
+operator --(int): Point
+operator ++(): Point
operator == (const Point& B): bool
Private:
-x: int
-y: int
基类
Block
private:
-allstate[7][4]; static string
-kind: int
- dir; int
public:
+Block()
- p: Point
-state[4][4]: bool
-StateInit(): static void
-StateCpy(): void
-Rotate(): void
-Show(bool): void
派生类
Game
private:
-level_up[9]: static int
-score_up[5]: static int
-interval[10]: static int
-level, score: int
派生类

```
public:
    -runB, nextB; Block
    +Game();
    +GetInterval():int
        +ShowFrame(); void
    +CheckBoard(); void
    +AllSquare(int xx); bool
    +DropDown(int xx); void
    +CheckLine(); void
    +PlaceOn(); void
    +Update(int); void
    +GetNext(); void
    +OutBoard(int, int); bool
    +CanChange(int, int); bool
    +ChangePos(int, int); bool
    +Rotate(); void
    +ChangeBlock(); void
    +Bomb(): void
    +BombOn(): void
    +Start(); void
    +Over(); void
    +Win(); void
```

界面设计：

游戏界面较为简单，包括游戏栏和信息栏，边框由虚线组成，设置光标在特定位置循环输出即构成了边界。不同的 level 有不同的背景、字体颜色

预览图如下：



4.系统调试

（1）第一次完整运行的时候，出现了行数只会闪不会消去的现象，闪的行数好像也不对，在 ChecLine()函数里找了好久，发现是横纵坐标搞反了，在这里，我定义的是行数为 x，列数为 y，与坐标系不同，对应的时候反应总有些迟钝

（2）写完 Block 类运行的时候出现了奇怪的错误，显示“无法解析的外部命令”和“无法解析的外部符号”，向同学求教才发现是 static string allstate[7][4]没有初始化，加上之后即使不赋值也会自动初始化，之前我没注意到这一点

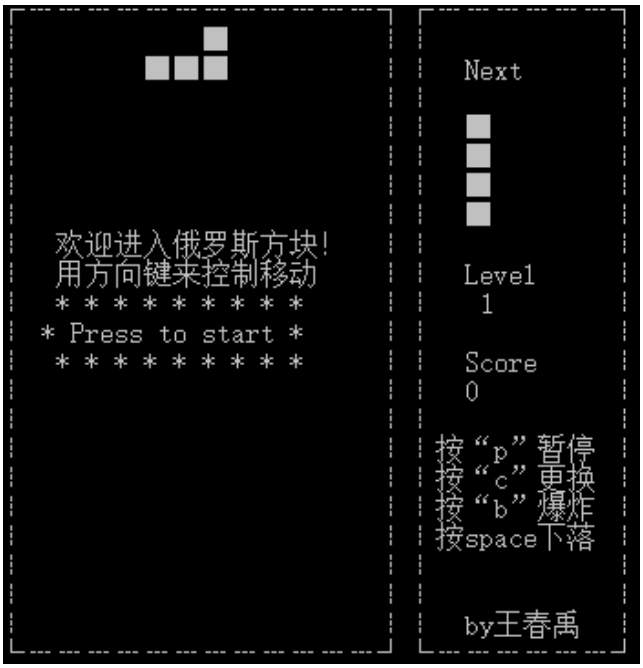
（3）原来没有考虑到碰壁或碰到其他方块时四个方向键是否还有效，导致出现一些不该出现的方块变形，后来加入了检查函数 CanChange()，每次检查键盘响应时，做出反应前都要判断能否发生这样的转变

（4）在增加按空格键直接下落这个功能时，有过好几种想法，开始想让方块在原位置消失（全输出 space），然后直接出现在正下方的方块堆上。但这样就很麻烦，如何让它出现在方块堆上？判断它 4 个 x 坐标下面是否是 space，然后重复执行下降？这样多数情况是不能直接到达它能停留的位置上。后来，我不让它直接“闪现”了，直接改变下落时间间隔就好了，让它加速下落，跟直接下落几乎等效，只要用个全局变量判断下就好了。

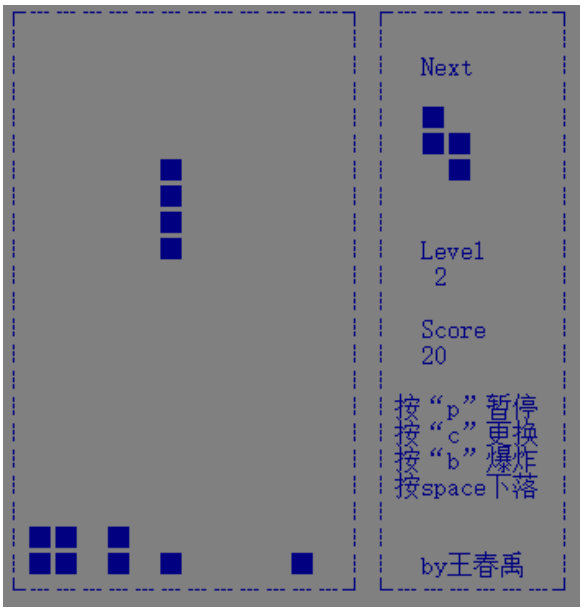
（5）在加入炸弹功能时是最费劲的，先要将当前方块变为”**”，然后迅速下落，接地后将下面、两侧接触的方块变没。这种功能一个函数完成不了，所以在检查键盘响应的时候先将当前方块变为炸弹，调用 Bomb（），之后将方块放置好，再调用 BombOn（）将四周的方块消掉，而且得判断是否有接触。调试的时候不是消失一片，就是不消失但是还可重叠，费很大劲

5．测试结果

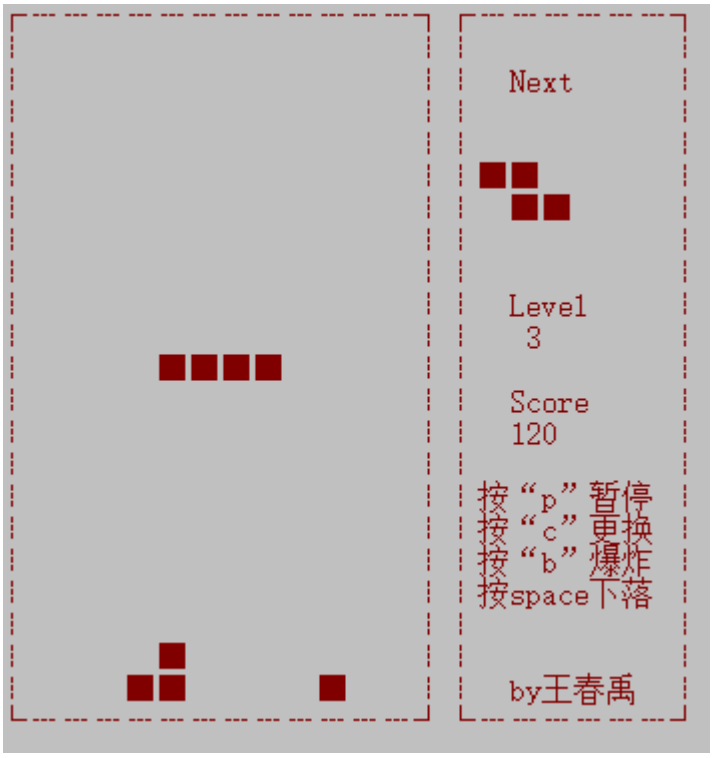
进入界面 & level1：



Level2: 速度加快



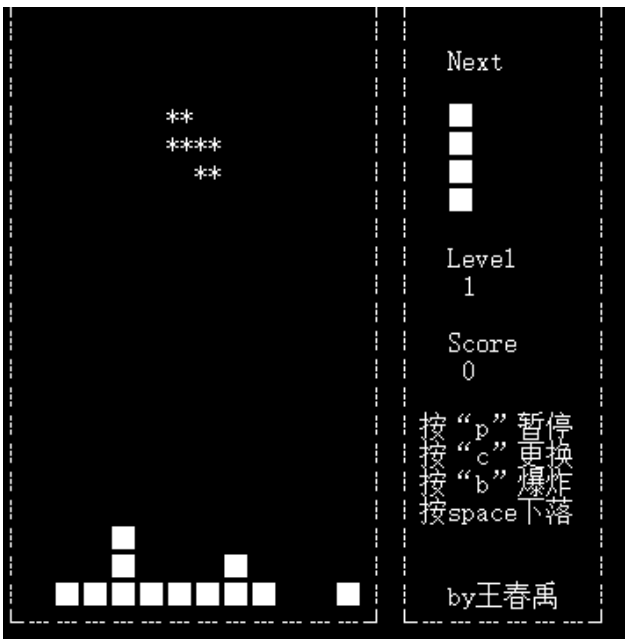
Level3: 速度加快



总共 8 关，就不一一展示了，其中第 5 关、第 8 关速度较快，难度大

按 p 可暂停，按 c 可切换方块种类，按 b 可将当前方块变为炸弹，但每用一次分数减少 5

按 b 变为炸弹



6.总结

这次写完还是蛮有成就感的，类的建立更清晰了，数据结构操作也小有尝试，还用到了类模板、函数多态性、运算符重载等知识点，也新增了些许功能，玩家使用起来还算舒适。

对于 windows 图形界面还是比较生疏，这里只是用了点简单的功能。假期打算学习下 qt。

对于文件格式还是不清楚，就是如何分各个类的头文件和源文件，分完之后编译总有错，上网查讲的不清楚，问同学也不清楚哪里出了 bug，所以干脆把类的定义和函数实现都放在头文件.h 里了

附录 1：源程序清单　　（共计 576 行）

```
#include<vector>
#include<time.h>
#include<string>
#include<conio.h>
#include<iostream>
#include<windows.h>
using namespace std;

const int N = 23;//游戏栏、信息栏的行数
const int M = 14;//游戏栏的列数
const int MM = 8;//信息栏的列数
static int ju = 0,bo=0;

const string square = "■";
const string bomb = "**";
const string space = "  ";
const string line[] = { "---","┆" };
const string corner[] = { "┌","┐","└","┘" };

string g[N][M], gg[N][MM];          //g 为游戏栏,gg 为信息栏

class Point                          //Point 类
```

```
{
public:
	Point() {}
	Point(int _x, int _y) :x(_x), y(_y) {}
	int getx() { return x; }
	int gety() { return y; }
	Point operator++(int)
	{
		y++;
		return (*this);
	}
	Point operator--(int)
	{
		y--;
		return (*this);
	}
	Point operator++()
	{
		x++;
		return (*this);
	}
	bool operator == (const Point& B)
	{
		return x == B.x && y == B.y;
	}
private:
	int x, y;
};

void SetCursor(int x, int y)		//设置光标位置
{
	COORD cd = { x,y };
	SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), cd);
}

void SetCursor(Point &p)		//将数组中的位置映射到屏幕上
{
	SetCursor(2 * p.gety(), p.getx());
}

void HideCursor()				//隐藏光标
{
	HANDLE hConsoleOutput = GetStdHandle(STD_OUTPUT_HANDLE);
	CONSOLE_CURSOR_INFO ConsoleCursorInfo;
```

```
    GetConsoleCursorInfo(hConsoleOutput, &ConsoleCursorInfo);
    ConsoleCursorInfo.bVisible = FALSE;
    SetConsoleCursorInfo(hConsoleOutput, &ConsoleCursorInfo);
}

template<typename T>
void PrintAt(Point &p, T data, int wid = 0)                    //在点 p 处输出 data
{
    SetCursor(p);
    if (wid)
        cout.width(wid);
    cout << data;
}

void   Choose(Point &p, bool flag = false, bool info = false) //在点 p 处输出/擦除方块
{
    if (info || g[p.getx()][p.gety()] == space)
        PrintAt(p, flag ? square : space);
}
void   Choose1(Point &p, bool flag = false, int i = 0)          //输出称赞
{
    if (i || g[p.getx()][p.gety()] == space)
    {
        switch (i)
        {
        case 1:PrintAt(p, flag ? "G o o d！" : space); break;
        case 2:PrintAt(p, flag ? "哎哟，  不错哦！" : space); break;
        case 3:PrintAt(p, flag ? "p e r f e c t！" : space); break;
        case 4:PrintAt(p, flag ? "登 峰 造 极！" : space); break;
        }
    }
}

class Game;

class Block:public Point                        //Block 类
{

private:
    static string allstate[7][4];          //7 种方块各有（至多）4 种状态
    int kind, dir;                              //kind 表示种类,dir 表示方向
public:
    Block();
    Point p;                                    //4*4 数组的左上角坐标
```

```
    bool state[4][4];                    //用 4*4 的数组标记自己的状态
    static void StateInit();          //初始化所有方块的状态
    void StateCpy();                     //将自己的状态与 kind 和 dir 对应
    void Rotate();                       //旋转
    void Show(bool);                     //将方块在屏幕上输出
};

string Block::allstate[7][4];          //初始化静态变量

void Block::StateInit()
{
    //长条形
    allstate[0][0] = allstate[0][2] = "0100 0100 0100 0100";
    allstate[0][1] = allstate[0][3] = "0000 1111 0000 0000";
    //正方形
    allstate[1][0] = allstate[1][1] = allstate[1][2] = allstate[1][3] = "0000 0110 0110 0000";
    //反"Z"形
    allstate[2][0] = allstate[2][2] = "0000 0110 1100 0000";
    allstate[2][1] = allstate[2][3] = "0100 0110 0010 0000";
    //"Z"形
    allstate[3][0] = allstate[3][2] = "0000 1100 0110 0000";
    allstate[3][1] = allstate[3][3] = "0100 1100 1000 0000";
    //"T"形
    allstate[4][0] = "0000 1110 0100 0000";
    allstate[4][1] = "0100 1100 0100 0000";
    allstate[4][2] = "0100 1110 0000 0000";
    allstate[4][3] = "0100 0110 0100 0000";
    //反"L"形
    allstate[5][0] = "0100 0100 1100 0000";
    allstate[5][1] = "1000 1110 0000 0000";
    allstate[5][2] = "1100 1000 1000 0000";
    allstate[5][3] = "0000 1110 0010 0000";
    //"L"形
    allstate[6][0] = "0100 0100 0110 0000";
    allstate[6][1] = "0000 1110 1000 0000";
    allstate[6][2] = "1100 0100 0100 0000";
    allstate[6][3] = "0010 1110 0000 0000";
}

Block::Block()
{
    kind = rand() % 7;                          //随机生成块的种类
    dir = rand() % 4;                           //随机生成块的方向
    p = Point(4, 15);                           //初始化点 p 的位置
```

```
    StateCpy();
}

void Block::StateCpy()
{
    for (int i = 0; i<4; i++)
        for (int j = 0; j<4; j++)
            state[i][j] = allstate[kind][dir][i * 5 + j] == '1';     //给每个方块的状态赋值
}

void Block::Rotate()
{
    dir = (dir + 1) % 4;
    StateCpy();
}

void Block::Show(bool info = false)          //info 表示是否是信息栏,下同
{
    for (int i = 0; i<4; i++)
        for (int j = 0; j<4; j++)
            Choose(Point(p.getx() + i, p.gety() + j), state[i][j], info);
}

class Game:public Block                          //Game 类
{
private:
    static int level_up[9];                    //各 level 对应的分数
    static int score_up[5];                     //消除各行对应要加的分数
    static int interval[10];                    //各 level 对应的时间间隔
    int level, score;
public:
    Block runB, nextB;
    Game();
    int GetInterval() { return interval[level]; }          //得到当前 level 的 interval
    void ShowFrame();                              //输出游戏框架
    void CheckBoard();                             //检查键盘响应
    bool AllSquare(int xx);                       //检查 xx 行是否可消
    void DropDown(int xx);                         //将 xx 行以上的全部下移一格
    void CheckLine();                              //方块安放后检查是否有可消行
    void PlaceOn();                                //方块安放好
    void Update(int);                              //更新信息
    void GetNext();                                //得到下一个方块
    bool OutBoard(int, int);                      //检查坐标是否出界
    bool CanChange(int, int);                      //检查是否可以发生这样的改变
```

```
    bool ChangePos(int, int);               //检查并改变下落方块的位置
    void Rotate();                          //旋转
    void ChangeBlock();                     //改变当前块
    void Bomb();                            //将当前块变为炸弹
    void BombOn();                          //炸弹降落
    void Start();                           //游戏欢迎界面
    void Over();                            //游戏结束界面
    void Win();                             //游戏获胜界面
};
int Game::level_up[9] = { 0,20,40,60,80,100,120,140,160 };
int Game::score_up[5] = { 0,10,30,60,100 };
int Game::interval[10] = { 0,50,40,25,15,8,40,20,6 };

Game::Game() :level(1), score(0)
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < M; j++)
            g[i][j] = "  ";
        for (int j = 0; j < MM; j++)
            gg[i][j] = "  ";
    }
    for (int i = 1; i < M; i++)
        g[0][i] = g[N - 1][i] = line[0];
    for (int i = 1; i < MM; i++)
        gg[0][i] = gg[N - 1][i] = line[0];
    for (int i = 1; i < N; i++)
    g[i][0] = g[i][M - 1] =gg[i][0] = gg[i][MM - 1] = line[1];
    g[0][0] = gg[0][0] = corner[0];
    g[0][M - 1] = gg[0][MM - 1] = corner[1];
    g[N - 1][0] = gg[N - 1][0] = corner[2];
    g[N - 1][M - 1] = gg[N - 1][MM - 1] = corner[3];
}

void Game::ShowFrame()
{
    for (int i = 0; i<N; i++)
    {
        for (int j = 0; j<M; j++)
            cout << g[i][j];
        for (int j = 0; j<MM; j++)
            cout << gg[i][j];
        cout << endl;
    }
```

```
        PrintAt(Point(2, 16), "Next");
        PrintAt(Point(9, 16), "Level");
        PrintAt(Point(10, 16), level, 2);
        PrintAt(Point(12, 16), "Score");
        PrintAt(Point(13, 16), score);
        PrintAt(Point(15, 15), "按“p”暂停");
        PrintAt(Point(16, 15), "按“c”更换");
        PrintAt(Point(17, 15), "按“b”爆炸");
        PrintAt(Point(18, 15), "按 space 下落");
        PrintAt(Point(21, 16), "by 王春禹");
}

bool Game::AllSquare(int xx)
{
        for (int j = 1; j<M - 1; j++)
                if (g[xx][j] != square)
                        return false;
        return true;
}

void Game::DropDown(int xx)
{
        for (int i = xx; i>1; i--)
                for (int j = 1; j<M - 1; j++)
                        g[i][j] = g[i - 1][j];
        for (int j = 1; j<M - 1; j++)
                g[1][j] = space;
}

void Game::ChangeBlock()
{
        for (int i = 0; i<4; i++)
                for (int j = 0; j<4; j++)
                        Choose(Point(runB.p.getx() + i,runB.p.gety() + j), false); //擦除原方块
        runB = nextB;                                   //生成新方块
        runB.p = Point(1, 5);
        nextB = Block();
        nextB.Show(true);
        runB.Show();
}

void Game::Update(int flash_line_cnt=0)
{
        score += score_up[flash_line_cnt];
```

```
        PrintAt(Point(13, 16), score,2);
        if (score >= level_up[level])
            PrintAt(Point(10, 16), ++level, 2);
        if(level&&score<level_up[level-1]) PrintAt(Point(10, 16), --level, 2);
            switch (level%7)
            {
            case 0:system("color 81"); break;
            case 1:system("color 0F"); break;
            case 2:system("color 81"); break;
            case 3:system("color 74"); break;
            case 4:system("color 80"); break;
            case 5:system("color 0E"); break;
            case 6:system("color 82"); break;
            default:break;
            }
        if (level >= 9)
        {this->Win(); exit(1);}
}

void Game::CheckLine()
{
        vector<int> flash_line;                         //用来储存要消去的行号
        vector<int>::iterator it;
        int k=0;                                                //可消除的行数
        for (int i = 0; i<4; i++)
            if (AllSquare(i + runB.p.getx()))
            {
                k++; flash_line.push_back(i + runB.p.getx());
            }
        if (flash_line.empty())
            return;
        int flash_times = 5;
        while (flash_times--)                                   //闪
        {
            for (it = flash_line.begin(); it != flash_line.end(); it++)
            {
                for (int j = 1; j < M - 1; j++)
                    Choose(Point(*it, j), flash_times & 2, true);
                Choose1(Point(10, 5), true, k);
                Choose1(Point(9, 5), true, k);
                Sleep(30);
            }
        }
        for (it = flash_line.begin(); it != flash_line.end(); it++)     //消去某行后，将上面的信息向
```

```
下移动
            DropDown(*it);
        it = flash_line.end() - 1;
        for (int i = 1; i <= *it; i++)
            for (int j = 1; j<M - 1; j++)
                Choose(Point(i, j), g[i][j] == square, true);
        Update(flash_line.size());                    //更新要消去的最后一行上方的信息
        flash_line.clear();
}

void Game::PlaceOn()
{
        for (int i = 0; i<4; i++)
            for (int j = 0; j<4; j++)
                if (runB.state[i][j])
                    g[runB.p.getx() + i][runB.p.gety() + j] = square;
}

void Game::GetNext()
{
        runB = nextB;
        runB.p = Point(1, 5);
        nextB = Block();
        nextB.Show(true);
        runB.Show();
}

bool Game::OutBoard(int x, int y)
{
        return x <= 0 || x >= N - 1 || y <= 0 || y >= M - 1;
}

void Game::CheckBoard()
{
            if (_kbhit())                                          //检测键盘是否按下
            {
                    switch (_getch())
                    {
                    case 72:Rotate(); break;                    //上
                    case 80:ChangePos(1, 0); break;             //下
                    case 75:ChangePos(0, -1); break;            //左
                    case 77:ChangePos(0, 1); break;             //右
                    case 'p':_getch(); break;                   //暂停
                    case ' ':ChangePos(0, 0); break;            //下落
```

```
                    case 'c':ChangeBlock(); break;              //更换
                    case 'b':Bomb(); bo = 1; break;
                    }
                }
}

bool Game::CanChange(int dx = 0, int dy = 0)
{
    const int x = runB.p.getx(), y = runB.p.gety();
    for (int i = 0; i<4; i++)
        for (int j = 0; j<4; j++)
            if (runB.state[i][j] && (OutBoard(i + x + dx, j + y + dy) || g[i + x + dx][j + y + dy]
== square))
                return false;
    return true;
}

void Game::Rotate()
{
    runB.Rotate();
    if (!CanChange())
        for (int i = 0; i<3; i++)
            runB.Rotate();
    runB.Show();
}

bool Game::ChangePos(int dx, int dy)
{
    if (!CanChange(dx, dy))
        return false;
    const int x = runB.p.getx(), y = runB.p.gety();
    if (dx)                                     //处理最后那行的显示
    {
        for (int j = 0; j<4; j++)
            Choose(Point(x, y + j));
        ++runB.p;
    }
    else if (dy == 1)                           //向右
    {
        for (int i = 0; i<4; i++)
            Choose(Point(x + i, y));
        runB.p++;
    }
    else if (dy == -1)                          //向左
```

```
    {
        for (int i = 0; i<4; i++)
            Choose(Point(x + i, y + 3));
        runB.p--;
    }
    else if (dy == 0)                                   //直接下落
    {
         ju = 1; return true;
    }
    runB.Show();
    return true;
}

void Game::Bomb()
{   if(score)
    score -= 5;
    PrintAt(Point(13, 16), score,2);
    Update();
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            if (runB.state[i][j])
            {
                int flash_times = 4;
                while (flash_times--)                                   //闪
                {
                    PrintAt(Point(runB.p.getx() + i, runB.p.gety() + j), bomb);
                    Sleep(5);
                }
            }
    bo = 1;
}

void Game::BombOn()
{
    PlaceOn();
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
        {
            if (runB.state[i][j])
            {
                g[runB.p.getx() + i][runB.p.gety() + j] = space;
                    PrintAt(Point(runB.p.getx() + i, runB.p.gety() + j), space);
                if (g[runB.p.getx() + i + 1][runB.p.gety() + j] == square)
                {
```

```
                        g[runB.p.getx() + i + 1][runB.p.gety() + j] = space;
                        PrintAt(Point(runB.p.getx() + i + 1, runB.p.gety() + j), bomb);
                        PrintAt(Point(runB.p.getx() + i + 1, runB.p.gety() + j), space);
                    }
if (g[runB.p.getx() + i][runB.p.gety() + j-1] == square)
                    {
                        g[runB.p.getx() + i][runB.p.gety() + j-1] = space;
                        PrintAt(Point(runB.p.getx() + i , runB.p.gety() + j-1), bomb);
                        PrintAt(Point(runB.p.getx() + i , runB.p.gety() + j-1), space);
                    }
                    if (g[runB.p.getx() + i][runB.p.gety() + j + 1] == square)
                    {
                        g[runB.p.getx() + i][runB.p.gety() + j + 1] = space;
                        PrintAt(Point(runB.p.getx() + i, runB.p.gety() + j + 1), bomb);
                        PrintAt(Point(runB.p.getx() + i, runB.p.gety() + j + 1), space);
                    }
                }
            }
}
void Game::Start()
{
    ShowFrame();
    nextB = Block();
    GetNext();
    int flash_times = 1000;
    while (!_kbhit() && flash_times--)
    {
        PrintAt(Point(8, 2), "欢迎进入俄罗斯方块！");
        PrintAt(Point(9, 2), "用方向键来控制移动");
        PrintAt(Point(10, 1), flash_times & 1 ? "                                    " : "    * * * * * * * *
* ");
        PrintAt(Point(11, 1), flash_times & 1 ? "      Press to start       " : " * Press to start * ");
        PrintAt(Point(12, 1), flash_times & 1 ? "                                    " : "    * * * * * * * *
* ");
        Sleep(100);
    }
    _getch();
    PrintAt(Point(8, 2), "                                  ");
    PrintAt(Point(9, 1), "                                   ");
    PrintAt(Point(10, 1), "                                    ");
    PrintAt(Point(11, 1), "                                    ");
    PrintAt(Point(12, 1), "                                    ");
}
```

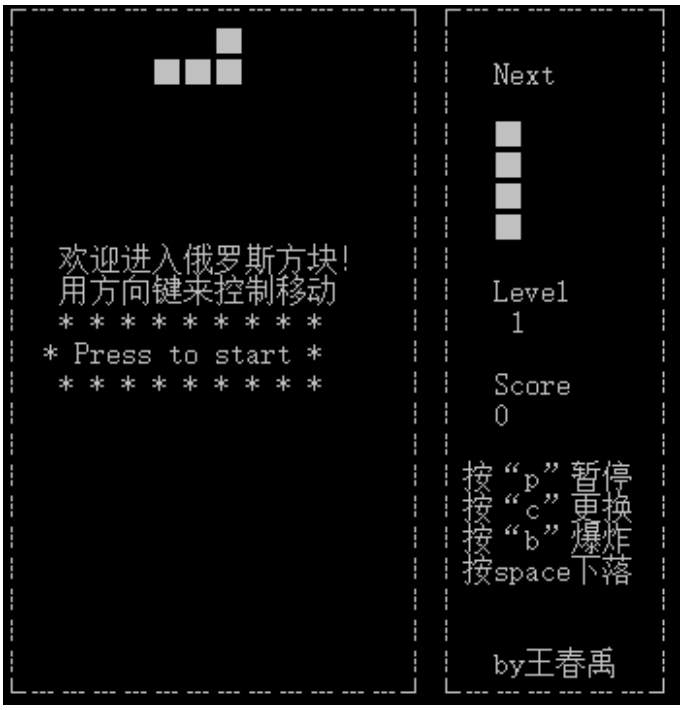
```
void Game::Over()
{
	int flash_times = 12;
	while (flash_times--)
	{
		PrintAt(Point(9, 3), flash_times & 1 ? "                          " : "    ******    ");
		PrintAt(Point(10, 3), flash_times & 1 ? "      Game over      " : " * Game over * ");
		PrintAt(Point(11, 3), flash_times & 1 ? "                          " : "    ******    ");
		Sleep(150);
	}
	PrintAt(Point(25, 0), "");                              //把光标移到最后
}
void Game::Win()
{
	system("color 82");
	for (int i = 0; i < 24; i++)
	{
		for (int j = 0; j < 22; j++)
			PrintAt(Point(i, j), square);
		cout << endl;
	}
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_INTENSITY |
		FOREGROUND_RED);
		PrintAt(Point(5, 3), "**你刚刚挑战完成了最快的 Tetrix**");
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_INTENSITY |
		FOREGROUND_BLUE);
		PrintAt(Point(7, 3), "***我跟不上你风驰电掣的速度！***");
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_INTENSITY |
			FOREGROUND_GREEN);
		PrintAt(Point(9,3),"**********是在下输了！**********");
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_INTENSITY |
			FOREGROUND_RED);
		PrintAt(Point(11,3), "**********恭喜你获胜！**********");
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
FOREGROUND_INTENSITY |
			FOREGROUND_GREEN);
		PrintAt(Point(13, 4), "******EE 出品，必属精品！******");
		system("pause");
}
int main()
```

```
{
    HideCursor();
    srand(time(NULL));
    Block::StateInit();
    Game* pG = new Game();
    pG->Start();
    while (1)
    {
        int tick_times = 0;
        while (++tick_times < pG->GetInterval())      //控制下落时间间隔以及控制输出
        {
            pG->CheckBoard();
            if (bo || ju)   Sleep(0.1);
            else Sleep(7);

        }
        if (!pG->ChangePos(1, 0))
        {
            if (pG->runB.p == Point(1, 5))          //如果一开始就下不去,game over
            {
                pG->Over();
                break;
            }
            if (!bo)
            {
                pG->PlaceOn(); pG->CheckLine();
             }
            else   pG->BombOn();
            pG->GetNext();
            ju = 0; bo = 0;
        }
    }
    delete pG;
    return 0;
}
```

附录 2：使用说明书

进入游戏欢迎界面，如下：



左侧为游戏栏，右侧为信息栏，Next 为下一个出现的方块，level 为关卡数，Score 为当前积分，其中，积分与关卡对应关系为：

Level1: 20	level5: 100
Level2: 40	level6: 120
Level3: 60	level7: 140
Level4: 80	level8: 160

按下任意键即可开始游戏

开始游戏后，可通过按键对方块进行相应操控

上方向键：改变当前方块方向

下方向键：加速下落

左/右方向键：方块左/右移

空格键：方块直接下落

C 键：当前方块变为下一个方块

P 键：暂停

B 键：将当前方块变为炸弹落下，与其接触的方块会消失，但每用一次分数会减少 5

每用方块填满一行，该行就会消去，得到 10 分，随积分增长，关卡数会增加，通过 8 关即可获得胜利。

附录 3：评分表

项　　目	评　　价	
选题报告与设计说明书	**1**	包括选题报告与结题报告两方面
程序基本要求涵盖情况	**4**	包括基本要求与基本工作量（500 行左右）
程序扩展要求与创新	**3**	包括扩展要求与同学自己附加的工作
程序代码编写素养情况	**2**	代码结构与风格
设计与运行结果	**4**	运行情况与鲁棒性
综合成绩	**15**	总分