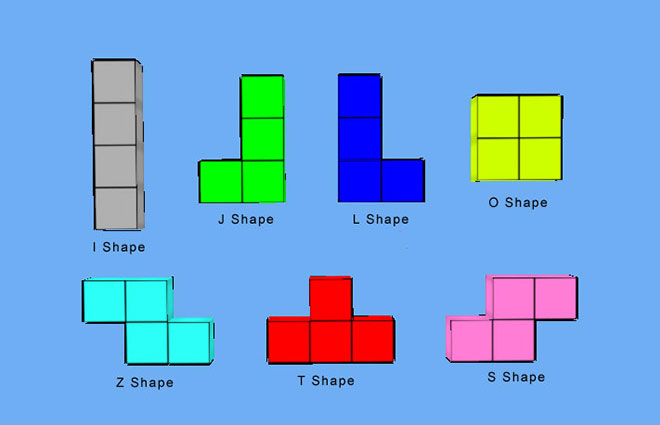
实验报告

———俄罗斯方块

1. 游戏介绍

俄罗斯方块由7个不同的小方块组成



这7个方块通过每次旋转90度都能得到一个新的图形，所以俄罗斯方块玩耍的过程中是有19个图形（有的旋转后重复），然后就变成了

**游戏规则：**

由小方块组成的不同形状的板块陆续从屏幕上方落下来，玩家通过调整板块的位置和方向，使它们在屏幕底部拼出完整的一条或几条。这些完整的横条会随即消失，给新落下来的板块腾出空间，与此同时，玩家得到分数奖励。没有被消除掉的方块不断堆积起来，一旦堆到屏幕顶端，玩家便告输，游戏结束。

1. 游戏设计

**游戏窗口的设计：**

**#include<Windows>的头**

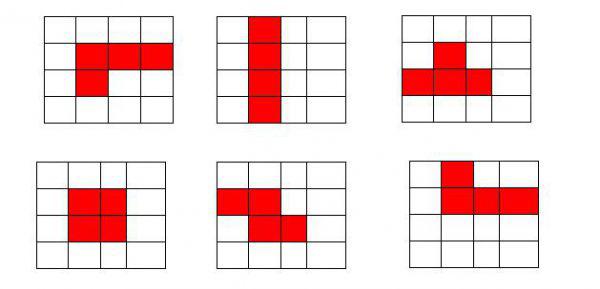
**定义合适的刷子和笔**

**画出游戏时的边框和游戏窗口旁边的游戏数据以及游戏规则**

**详情可见ui.c**

**方块的设计：**

**通过观察我们可以发现方块中的图形都可以在一个4\*4的方块中体现出来**



**这样我们就能把所有的图像都存储为4\*4图像中的坐标**

**我们可以定义结构体**

struct B

{

POINT pt[4];

}Blocks[][4] =

{

//正7

0, 0, 1, 0, 1, 1, 1, 2, 2, 0, 0, 1, 1, 1, 2, 1, 1, 0, 1, 1, 1, 2, 2, 2, 0, 1, 1, 1, 2, 1, 0, 2,

//反7

1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 1, 1, 2, 1, 2, 2, 1, 0, 1, 1, 0, 2, 1, 2, 0, 0, 0, 1, 1, 1, 2, 1,

//长方体

1, 0, 1, 1, 1, 2, 1, 3, 0, 0, 1, 0, 2, 0, 3, 0, 1, 0, 1, 1, 1, 2, 1, 3, 0, 0, 1, 0, 2, 0, 3, 0,

//正Z

0, 0, 1, 0, 1, 1, 2, 1, 2, 0, 1, 1, 2, 1, 1, 2, 0, 0, 1, 0, 1, 1, 2, 1, 2, 0, 1, 1, 2, 1, 1, 2,

//反Z

1, 0, 2, 0, 0, 1, 1, 1, 1, 0, 1, 1, 2, 1, 2, 2, 1, 0, 2, 0, 0, 1, 1, 1, 1, 0, 1, 1, 2, 1, 2, 2,

//正方体

0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1,

//凸

1, 0, 0, 1, 1, 1, 2, 1, 0, 0, 0, 1, 1, 1, 0, 2, 0, 0, 1, 0, 2, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 2

}

**那么这些数字到底表达了了什么意思呢？**

**例如（0, 0, 1, 0, 1, 1, 1, 2 ），这是其中的一小组的数据，这组数据表示了一个图形，这个图形存在于4\*4的方格中，数字代表了坐标（0,0）（1,0）（1,1）（1,2）**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **0** |
| **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **0** |

**这样我们把1的地方认为是有颜色的就可以画出不同形状的方块了，而我的每一行就是一个方块和他旋转后的方块的总和，然后总共有7个方块，所以构成了这样的结构体。然后变形的时候实际上就是一个循环，如果到了一排中最后一个图形，下一个就转到第一个。**

**游戏操作：**

**游戏的操作函数有**

//随机数函数定制版，用于随机出现的方块

unsigned Random(int n);

**用这个随机函数的定义，可以生成随机数，这样可以调用上面结构体中的数据，就可以随机生成方块，而且能够随机判断下落的位置。**

//判断是否可以下落，可以则返回true

bool CanDown(POINT pt[]);

**先假设这个方块是可以下落的，然后下落了之后如果碰到了边界或者是下落后的地方已经有了方块，则判断这个方块是否还能再下落。**

//实现下落的过程

void Down(POINT pt[]);

**将方块下落一个单位。**

//判断是否可以左移

bool CanLeft(POINT pt[]);

**先假设这个方块是可以左移的，然后左移了之后如果碰到了边界或者是左移后的地方已经有了方块，则判断这个方块是否还能再左移。**

//实现左移的过程

void Left(POINT pt[]);

**将方块左移一个单位。**

//判断是否可以右移

bool CanRight(POINT pt[]);

**先假设这个方块是可以右移的，然后右移了之后如果碰到了边界或者是右移后的地方已经有了方块，则判断这个方块是否还能再右移。**

//实现右移的过程

void Right(POINT pt[]);

**将方块右移一个单位。**

//判断是否可以变形

bool CanChange(POINT pt[]);

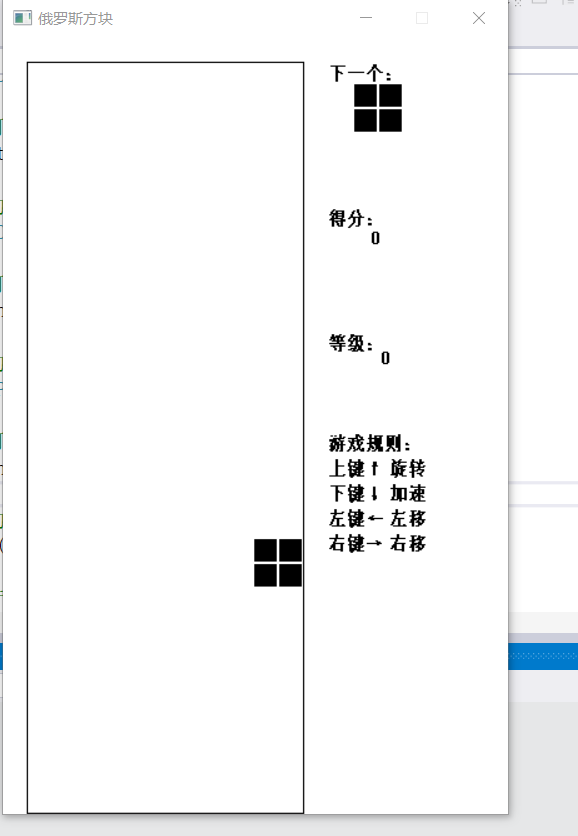
**先假设这个方块是可以变形的，然后变形了之后如果碰到了边界或者是变形后的地方已经有了方块，则判断这个方块是否还能再变形。**

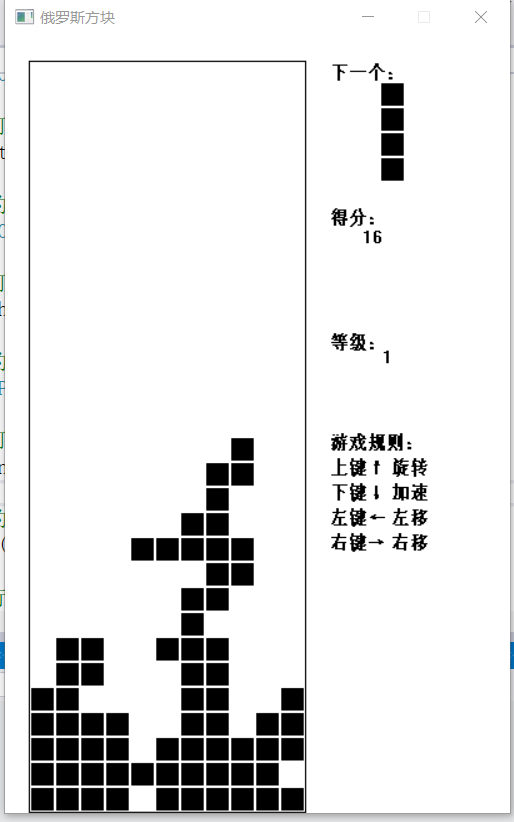
//实现变形的过程

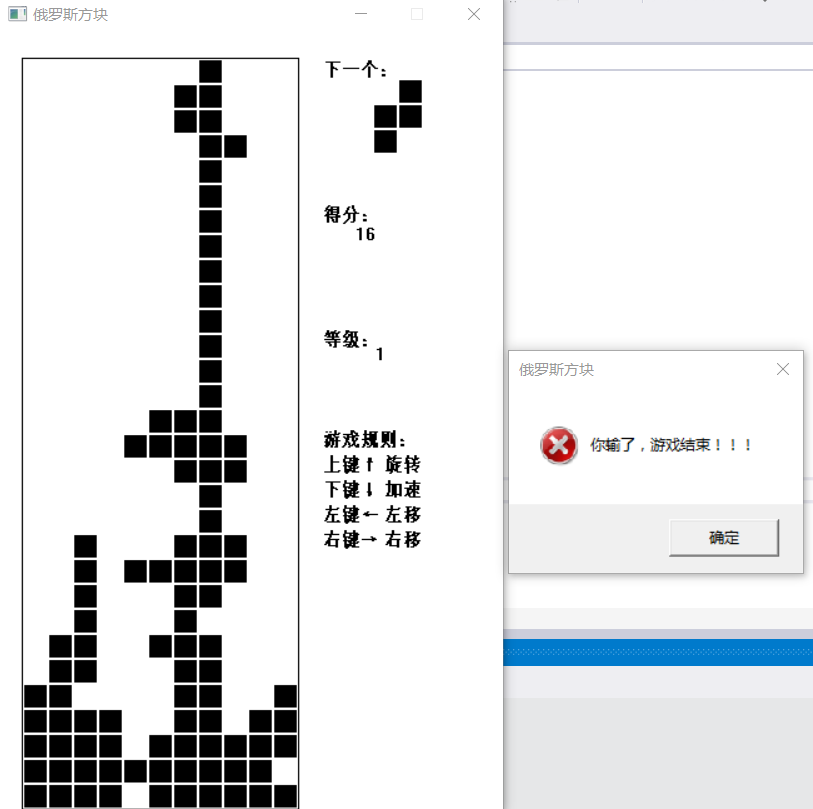
void Change(POINT pt[]);

**将方块变形。**

1. 游戏效果

  **随机的方块会从随机的位置落下，右边的边框会显示下一个将要落下的方块的形状，然后每消除一行就会得一分，我设定10分为一个等级，每增加10分就会升一个等级。**





**如果方块在某一位置不能够再落下，那么出现一个窗口，点击确定以后程序自动结束。**

1. **感想**

**C语言的结构严谨，我很开心写出游戏来，但是我的游戏还是存在一个bug，在进行转换时，会闪一下，有些方块不能及时的画出来。通过小学期的写游戏经历，不再把游戏视为游戏，而是把从游戏中又发现了一个新的世界。**

**第一次自己写，加上对于数据结构的了解不是那么透彻，所以借鉴了网上的一些东西，还向同学请教了了很多，虽然写的艰难困苦，但不得不说这是一次有趣的经历。**