Project1开发文档

**一、结构设计与分析**

**1.**

类体规定了当前方块的静态二维数组（4×4）、下一方块的静态二维数组（4×4）、提示下落位置的静态二维数组（4×4）、各类不同方块的静态二维数组（4×4）、主界面的String静态二维数组（16×13）、提示下一方块的界面的String静态二维数组（4×4）、方块初始位置的横纵坐标（静态）、分数值（静态）。

注：方块数组中的0、1为信号，0代表此位置无方块，1代表此位置有方块。

**2.**

主方法主体由一个while循环组成。循环外的部分为初始化主界面、初始化提示下一方块的界面、获得当前方块的类型、获得下一个方块的类型、打印游戏开始界面；

循环内，用over（）方法判断当次游戏是否结束，如果当次游戏结束，则回到游戏开始界面（这一步用到了循环时的标签），供玩家选择是否进行下一次游戏。

**3.**

游戏过程中，首先运用clear（）方法与cleradown（）方法清除上一次操作时留下的的方块痕迹；

之后玩家用command（）方法对玩家输入的相应指令控制当前方块的移动与变换；

如果出现可以消除行的现象，则调用clearline（）方法消除行，同时记录相应的分数；

方块在移动时，有下落位置的提示，需要getdownblock（）方法，获得下落位置在主界面中的位置；

之后用gettable（）方法，将当前主界面赋值给已经初始化的主界面的静态二维数组（16×13）；

提示下一方块需要用到getsmalltable（）方块，将下一方块的信息赋值给已经初始化的提示下一方块界面的静态二维数组（4×4）；

最后用一系列打印方法：printsmalltable（）、printScore()、Printtable()将提示下一方块的界面、分数与主界面打印出来，一次循环完成！

如果方块达到了停止运动的条件，即stop（）方法返回值为true，则将下一方块nextblock赋值给当前方块block，并且清空提示下一方块的界面，再产生随机产生下一个方块的提示。同时把方块位置的横纵坐标重新调回初始值。重复循环。

**二、主要函数功能**

**1.gettable（）方法：**

遍历当前方块block的4×4的数组，若发现信号“1”，则将该信号反映到主界面中，类似于物理中的伽利略坐标变换，table[centerX + i][centerY + j]代表主界面坐标与当前方块坐标的叠加。结果是在主界面中，这个位置被赋予一个“\* ”（星号，代表此处有方块）注：getsmalltable（）方法与此类似，只不过是nextblock与smalltable之间的关系，用于提示下一方块的界面。

**2.random（）与nextrandom（）方法：**

用随机数方法取0-6之间的任意整数，将各类型方块与数字对应起来，然后分别将随机对应所得方块赋值给block与nextblock。

**3. initialize()与initializesmall()方法：**

将table数组与smalltable数组分别初始化，若不初始化，则数组内全为null。初始化为无方块的界面

**4.clear（）与cleardown（）方法：**

前者遍历当前方块block数组，若检测到信号“1”，则将table上的对应位置变成“ ”（空格），起到了将主界面原来为方块的位置清除的效果；cleardown（）方法，遍历整个主界面，当搜寻到“+ ”时，将该处变为“ ”（空格），起到了将下落位置提示清除的效果。

**5.spin（）方法**：

实现旋转。首先判断方块类型是否为长条，若是，则引入tempI数组，旋转中心为（1.5，0），运用数学中旋转90度的坐标变换，将当前方块block数组的0和1做相应调整；若不是，则旋转中心为（1,0），同理。注：若不能旋转，即到达边界或者左右有方块，则不能旋转。，

**6.command（）方法：**

读取控制台上的输入的字符，相应地做出变换，每次移动、旋转操作，中心横坐标均加一，即向下移动一格。其中，在判断是否旋转时，首先判断当前方块是否为正方形方块，若是，则旋转无变化，若不是，则调用spin（）方法。

**7.over（）方法：**

判断当次游戏结束。引入一维boolean类型的数组over【15】，检测每i行是否有“\* ”，若有，则over【i】为true，当且仅当整个over数组的每个元素均为true时，代表游戏结束。

**8.stop（）方法：**

判断方块停止运动。检测到当前方块最下端为下边界或者还有方块时，方块停止运动。判断当前方块最下端还有没有方块时，需要调用thelastdig（j）方法，thelastdig（j）方法的返回值是当前方块最下端位于主界面上的行数，即横坐标。该方法需要传入参数j，即block数组中的的第几列，随后遍历当前方块block数组的每一行，若有信号“1”，则返回这个信号所在主界面的行数。由此，*table*[*thelastdig*(j)+ 1 ][*centerY* + j]代表了主界面上当前方块的最下端再下一行所在的位置，再用equals方法判断此位置是否为“\* ”或边界，以判断是否停止。

**9.judgement（）方法**：

返回值为-1或者1，用于“逆操作”：当方块无法再向左移时，会返回1；当方块无法再向右移时，会返回-1。若无法左移了，玩家还输入a，则在command（）方法中有centerY+=judgemendt（），相当于不移动。右移同理

**10.** **leftboundary（）方法**：

判断当前方块能否向左移动。检测到方块最左端到达左边界或者还有方块时，该方法返回true，供judgement（）方法选择返回1。theleftdig（i）方法的返回值是当前方块最左端方块位于主界面上的列数，即纵坐标。此方法需要传入参数i，即block数组中的第几行，随后遍历随后遍历当前方块block数组的每一列，若有信号“1”，则返回这个信号所在主界面的列数。由此table[centerX+i][theleftdig(i) - 1] 代表了主界面上当前方块的最左端再左一行所在的位置，再用equals方法判断此位置是否为“\* ”或边界。

**11.rightboundary（）方法**：

判断当前方块能否向右移动。原理同leftboundary（），这里不解释。

**12.fall（）方法：**

返回当前方块离停止向下运动还需的的行数。运用while循环，调用stop（）方法判断是否停止运动，将当前位置的centerX保存在beforecenterX中，引入整数fall，若没有停下，则每次centerX加一，fall也加一。循环结束后，将beforecenterX赋值给centerX，相当于回到原位，然后返回fall，fall即当前方块离停止向下运动还需的的行数。此方法既可以实现方块的快速到达底部，又可以用于方块下落位置的提示。

**三、遇到的问题及解决办法**

**问题1：**

如何判断方块左边或右边有其他方块，尝试了table[ i ][ j + 1] == “\* ”，发现无论怎样都是false的。

解决方案：

学习了字符串之后，了解到了equals方法，明白了 == 与equals方法的不同，运用equals方法解决了该问题。

**问题2：**

初学数组时，总是遇到数组溢出问题，并且不知道问题所在。

解决方案：

查看书本上的例题，发现数组的“差一错误”，将数组的大小扩大了，解决了问题。

**问题3：**

判定当次游戏结束时，以前的方法是检测到最上面一行，若有“\* ”时，则进入下一次循环，发现当一个新的方块出现时，当次游戏就会结束，不符合要求。

解决方案：

引入一维boolean类型的数组over【15】，检测每i行是否有“\* ”，若有，则over【i】为true，当且仅当整个over数组的每个元素均为true时，代表游戏结束。

**问题4：**

刚刚开始无法实现下坠，因为每次一下坠，下面的方块就会重合，导致错误。

解决方案：

在大佬的提示下引入fall（）方法，由于fall（）方法中调用了stop（）方法，使方块停止在何处有了正确的判断。