Name: 傅彦钧

ID: 516030910354

关于大作业**Teris(**俄罗斯方块**)**的说明

此大作业为游戏Teris(俄罗斯方块)。此游戏中包含7种基本方块，L型，O型，Z型，I型，S型，T型，J型。每种方块颜色不同。将有一块20\*10的区域，随机产生一种方块，对此方块可以施加旋转90度，向左移动，向右移动，向下加速移动的操作。不施加操作时，方块自动向下落，直到其底部已有方块或到达最下一层为止，此时其颜色变为灰色。当10\*20区域方块已堆到顶部，那么游戏结束。若有一行已经填满方块，那么此行消除，其他到底方块下移。游戏分数加10分。游戏将允许玩家暂停游戏，开始游戏，重新开始游戏，退出游戏，查看游戏帮助，查看“关于”以了解程序编写者信息。游戏结束将有提示，玩家可根据提示选择是否重新开始或退出。选择中途退出也将有提示，供玩家选择是否继续退出。

为方便GUI的构建。此程序选择Tkinter模块，此外还选择了tkMessageBox, time，random模块。此程序以矩阵及矩阵变换作为核心思想和算法，以下为详细解释。

模块teris.py：

一**.** 初始化：

**1.** 创立窗口，加入GUI构件：画布（Canvas）,按键（Pause/Resume:暂停/继续游戏，Start:开始游戏，Restart:重新开始，Quit:退出游戏），标签（Label）来显示当前的分数，菜单（Menu）(New Game:新游戏，Start:游戏开始，Exit:退出，Help:查看游戏帮助和“关于”)。添加键盘操作（a或左键：向左移，w或上键：旋转，s或下键：向下加速）。

**2.** 将20\*10的画布抽象成20行，10列的矩阵，用map表示，此时矩阵中所有元素为0。扩充矩阵，左边添加一列，右边添加一列，底部添加一行，将扩充位置的元素都设为1，即为这个矩阵左，右，下方向添加了边界，方块到边界处将不能移动。

**3.** 对于方块的创建，所有方块都包含四个小方块，我们选取之一为核心块（core），其位置为（0，0），其他三个方块的相对于它的位置可以给出，用字典类型记录相对位置。一共七种方块，编号1～7（self.shapeDict）。建立随机颜色列表，7种颜色（self.color）。

self. shapeDict = {1: [(0,0), (0, -1), (0, -2), (0,1)], # shape I

2: [(0,0), (0, -1), (1, -1), (1,0)], # shape O

3: [(0,0), (-1,0), (0, -1), (1,0)], # shape T

4: [(0,0), (0, -1), (1,0), (2,0)], # shape J

5: [(0,0), (0, -1), (-1,0), (-2,0)], # shape L

6: [(0,0), (0, -1), (-1, -1), (1,0)], # shape Z

7: [(0,0), (-1,0), (0, -1), (1, -1)]} # shape S

self. color = ['red', 'orange', 'yellow', 'purple', 'blue', 'green', 'pink']

**4.** 初始高度（self. height）:20,初始宽度（self. width) :10，初始分数（self. score）：0，核心块初始位置（self. coreLocation）:[4, -2]

**5.** 对于旋转。每个块相对于核心块相对位置已知，那么旋转以后的相对位置变化与原始位置一一对应，将这种一一对应关系用字典self.rotateDict表示，知道原始位置就可以访问到旋转90度以后的位置。如（1，0）到（0，-1）

self. rotateDict = {(0,0) :(0,0), (0,1) :(-1,0), (0,2) :(-2,0), (0, -1) :(1,0),

(0, -2) :(2,0), (1,0) :(0,1), (2,0) :(0,2), (-1,0) :(0, -1),

(-2,0) :(0, -2), (1,1) :(-1,1), (-1,1) :(-1, -1),

(-1, -1) :(1, -1), (1, -1) :(1,1)}

**6.**为方便暂停和加速运动功能使用，加入两个初始值：self. isPause=True, self. isFaster = False。

二**.**获取方块位置：

先将核心块的所在位置在map中的元素设为1，通过self.shapeDict获取其余方块位置，将map中对应元素设为1。

三**.**判断方块是否可继续下移：

判断方块下移一格后对应位置map中的元素是否为一，是，则不可移动，返回False；否，可以移动，返回True。

四**.**画出移动方块：

先用randRange获取1～7中的随机整数，随机到某一整数，那么访问self.shapeDict,获取这种形状方块的核心块及其他方块的相对位置。访问颜色字典，获取此方块的颜色。建立循环，当方块可移动时(while self. canMove():)，且暂停键未被摁下（if isPause:)，核心块纵坐标加一，根据核心块及其他方块对于核心块的相对位置，画出四个方块。用self.getLocation()函数获取方块的位置。

五**.**画出底部方块：

给底部每行中方块都加上标签：bottom + str(j), j代表该块所在行数，每次遍历map，建立对于range（self. height）的for循环，删去每一行，若map什么地方的元素为1，画出这一位置的方块，不断更新。这样可以画出底部方块。

六**.**判断填满并加分：

遍历map每一行的各个元素，若所有元素为1，则标签中score值+10，将此行所有元素改为0，行数map（i,j)=map(i-1,j)（即所有之上的行下移），那么后续画底部方块时，可实现消行。

七**.**判断是否游戏结束：

遍历每一行，若从顶部到底部map每一行都有某一个元素或更多元素为1，那么说明方块以顶到最上端，游戏结束。此处不可以简单判定最上一行是否有元素为1就判定结束，若这样会产生顶部有新的方块产生，然后导致顶部有元素为1，误判为游戏结束。

八**.**方块的旋转：

先判断方块是否可以旋转（针对其靠近边界时）。先将其现在所在位置对应map中的元素改为0，判断其旋转后位置对应map中的元素是否有一，若有，说明其旋转后的位置已经被占，是不能旋转的，返回值为False。否则为可旋转，返回值True。若已判定可以旋转，那么访问self.rotateDict，得出旋转以后所有小块的位置变换，将变换以后的位置对应map的元素设为1，旋转便已完成。

九**.**方块的左移：

先判断是否左移，同样，将方块现在所处位置的map中元素设为0，看其移动后的位置上map的元素是否有1，若有，说明这一位置已被占据或已到边界，不可移动，返回False。若可移动，返回True。按下左键，若可以移动，核心块的横坐标减1，由于我们只讨论其他小块对于核心块的相对位置，所以其他小块的位置自动随核心块的位置移动而移动。将移动过后的位置对应map中的元素设为1。

十**.**方块的右移：

与左移相似，略。

十一**.**暂停功能：

每次一按下暂停键，isPause = not isPause,当isPause = True时，由于之前提到过的if isPause:语句，方块可以移动，游戏运行。当按下暂停键以后，isPause值为False，方块将不可移动。同时，isPause值为False时，暂停键变为开始键，即标签由Pause 改为 Resume,当isPause值为True时，Resume改为Pause。这一功能由if语句实现。

十二**.**方块加速：

初始化中有一self. isFaster 的变量被设为False，当其为False时，程序中的sleep(time)中time的值为0.35，而按下下键，self. isFaster变为True，time变成0.05，通过调整sleep()中变量的大小可以调节方块运动的速度。此功能通过if语句实现。

十三**.**退出：

当玩家选择退出时，将弹出提示框，若得到确认，则窗口(self. root)会self.root.destroy(),窗口关闭，游戏结束。

十四.游戏的运行：

在while True:的控制下，若游戏未结束，画方块，消行，加分。若游戏结束，弹出对话框，询问是否restart，是，重新跑这个程序，所有值全部进行初始化，否，self.root.destroy()。

主程序**TerisPlay.py:**

主函数：引用teris模块，用mainloop()进入主循环，游戏即开始。