*"""*

*游戏逻辑控制器，负责处理游戏核心算法．*

*"""*

**from** model **import** DirectionModel

**from** model **import** Location

**import** random

**class** GameCoreController:

**def** \_\_init\_\_(self):

self.\_\_list\_merge = **None**

self.\_\_map = [

[0, 0, 0, 0],

[0, 0, 0, 0],

[0, 0, 0, 0],

[0, 0, 0, 0],

]

self.\_\_list\_empty\_location = []

@property

**def** map(self):

**return** self.\_\_map

**def** \_\_zero\_to\_end(self):

*"""*

*零元素移动到末尾.*

*"""*

**for** i **in** range(-1, -len(self.\_\_list\_merge) - 1, -1):

**if** self.\_\_list\_merge[i] == 0:

**del** self.\_\_list\_merge[i]

self.\_\_list\_merge.append(0)

**def** \_\_merge(self):

*"""*

*合并*

*"""*

self.\_\_zero\_to\_end()

**for** i **in** range(len(self.\_\_list\_merge) - 1):

**if** self.\_\_list\_merge[i] == self.\_\_list\_merge[i + 1]:

self.\_\_list\_merge[i] += self.\_\_list\_merge[i + 1]

**del** self.\_\_list\_merge[i + 1]

self.\_\_list\_merge.append(0)

**def** \_\_move\_left(self):

*"""*

*向左移动*

*"""*

**for** line **in** self.\_\_map:

self.\_\_list\_merge = line

self.\_\_merge()

**def** \_\_move\_right(self):

*"""*

*向右移动*

*"""*

**for** line **in** self.\_\_map:

self.\_\_list\_merge = line[::-1]

self.\_\_merge()

line[::-1] = self.\_\_list\_merge

**def** \_\_move\_up(self):

self.\_\_square\_matrix\_transpose()

self.\_\_move\_left()

self.\_\_square\_matrix\_transpose()

**def** \_\_move\_down(self):

self.\_\_square\_matrix\_transpose()

self.\_\_move\_right()

self.\_\_square\_matrix\_transpose()

**def** \_\_square\_matrix\_transpose(self):

*"""*

*方阵转置*

**:param** *sqr\_matrix: 二维列表类型的方阵*

*"""*

**for** c **in** range(1, len(self.\_\_map)):

**for** r **in** range(c, len(self.\_\_map)):

self.\_\_map[r][c - 1], self.\_\_map[c - 1][r] = self.\_\_map[c - 1][r], self.\_\_map[r][c - 1]

**def** move(self, dir):

*"""*

*移动*

**:param** *dir: 方向,DirectionModel类型*

**:return***:*

*"""*

**if** dir == DirectionModel.UP:

self.\_\_move\_up()

**elif** dir == DirectionModel.DOWN:

self.\_\_move\_down()

**elif** dir == DirectionModel.LEFT:

self.\_\_move\_left()

**elif** dir == DirectionModel.RIGHT:

self.\_\_move\_right()

*# def generate\_new\_number(self):*

*# # 思路:选出所有的空白位置(行／列),再随机挑选一个.*

*# list\_empty\_location = []*

*#*

*# for r in range(len(self.\_\_map)):#0 1 2 3*

*# for c in range(len(self.\_\_map[r])):*

*# if self.\_\_map[r][c] == 0:*

*# # 记录r c --> 元组*

*# list\_empty\_location.append((r,c))*

*# 　　# 确定哪个空白位置1 0*

*# loc = random.choice(list\_empty\_location)*

*# 　 　#　产生随机数*

*# 　　if random.randint(1,10) == 1:*

*# self.\_\_map[loc[0]][loc[1]] = 4*

*# else:*

*# self.\_\_map[loc[0]][loc[1]] = 2*

**def** generate\_new\_number(self):

*"""*

*生成新数字*

*"""*

self.\_\_get\_empty\_location()

**if** len(self.\_\_list\_empty\_location) == 0:

**return**

loc = random.choice(self.\_\_list\_empty\_location)

*# if random.randint(1, 10) == 1:*

*# self.\_\_map[loc.r\_index][loc.c\_index] = 4*

*# else:*

*# self.\_\_map[loc.r\_index][loc.c\_index] = 2*

self.\_\_map[loc.r\_index][loc.c\_index] = self.\_\_select\_random\_number()

*# 因为在该位置生成了新数字，所以该位置就不是空位置了．*

self.\_\_list\_empty\_location.remove(loc)

**def** \_\_select\_random\_number(self):

**return** 4 **if** random.randint(1, 10) == 1 **else** 2

**def** \_\_get\_empty\_location(self):

*# 每次统计空位置，都先清空之前的数据，避免影响本次数据．*

self.\_\_list\_empty\_location.clear()

**for** r **in** range(len(self.\_\_map)):

**for** c **in** range(len(self.\_\_map[r])):

**if** self.\_\_map[r][c] == 0:

self.\_\_list\_empty\_location.append(Location(r, c))

**def** is\_game\_over(self):

*"""*

*游戏是否结束*

**:return***: False表示没有结束* *True 表示结束*

*"""*

*# 是否具有空位置*

**if** len(self.\_\_list\_empty\_location) > 0:

**return False**

*# # 判断横向有没有相同的元素*

*# for r in range(len(self.\_\_map)):*

*# for c in range(len(self.\_\_map[r]) - 1): # 0 1 2*

*# if self.\_\_map[r][c] == self.\_\_map[r][c + 1]:*

*# return False*

*#*

*# # 判断竖向有没有相同的元素*

*# for c in range(4):*

*# for r in range(3):*

*# if self.\_\_map[r][c] == self.\_\_map[r + 1][c]:*

*# return False*

**for** r **in** range(len(self.\_\_map)):*#0*

**for** c **in** range(len(self.\_\_map[r]) - 1): *# 0 1 2*

**if** self.\_\_map[r][c] == self.\_\_map[r][c + 1] **or** self.\_\_map[c][r] == self.\_\_map[c+1][r]:

**return False**

**return True**

*# ---------测试代码---------------*

**if** \_\_name\_\_ == **"\_\_main\_\_"**:

controller = GameCoreController()

*# controller.move\_left()*

*# print(controller.map)*

*# controller.move\_down()*

*# print(controller.map)*

*# controller.move(DirectionModel.LEFT)*

*# print(controller.map)*

*# controller.move(DirectionModel.RIGHT)*

*# print(controller.map)*

controller.generate\_new\_number()

controller.generate\_new\_number()

controller.generate\_new\_number()

controller.generate\_new\_number()

controller.is\_game\_over()

print(controller.map)