**Федеральное государственное образовательное**

**бюджетное учреждение высшего образования**

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ**

**ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**ИНСТИТУТ ЗАОЧНОГО И ОТКРЫТОГО ОБРАЗОВАНИЯ**

**ДЕПАРТАМЕНТ АНАЛИЗА ДАННЫХ И МАШИННОГО ОБУЧЕНИЯ**

**ПРАКТИЧЕСКАЯ РАБОТА № 1**

**по дисциплине «Практикум по программированию»**

**Вариант №10. Игра «Пятнашки»**

Направление подготовки - 09.03.03 «Прикладная информатика»

Студент(ка) **Белоусов А.А.**

(Ф.И.О.)

Курс2 № группы **ЗБ-ПИ19-1**

Преподаватель **Горохова Р.И.**

(Ф.И.О.)

Москва 2020

Задание

Реализовать программу, с которой можно играть в игру «Пятнашки». Правила игры следующие.

Головоломка представляет собой 15 квадратных костяшек с числами от 1 до 15. Все костяшки заключены в квадратную коробку (поле) размером 4 на 4. При размещении костяшек в коробке остается одно пустое место, которое можно использовать для перемещения костяшек внутри коробки. Цель игры - упорядочить размещение чисел в коробке, разместив их по возрастанию слева направо и сверху вниз, начиная с костяшки с номером 1 в левом верхнем углу и заканчивая пустым местом в правом нижнем углу коробки.

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 4 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна считать количество сделанных ходов, уметь автоматически определять недопустимые ходы, окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается упорядочить костяшки с целью выиграть игру.

Код программы

Код файла main.py

import pygame  
from puzzle\_game.constants import WIDTH, HEIGHT, SQUARE\_SIZE, BORDEROUT, BORDERIN, FOOTER, BTNWIDTH, BTNHEIGHT  
from puzzle\_game.game import Game  
  
pygame.init()  
  
FPS = 60  
  
WIN = pygame.display.set\_mode((WIDTH, HEIGHT))  
pygame.display.set\_caption(**'Пятнашки'**)  
  
  
def pos\_on\_btn(pos):  
 x, y = pos  
 if x >= BORDEROUT and x <= BTNWIDTH + BORDEROUT and y >= HEIGHT - BTNHEIGHT - BORDEROUT and y <= HEIGHT - BORDEROUT:  
 return True  
 return False  
  
  
def pos\_on\_board(pos):  
 on = True  
 x, y = pos  
 if x >= BORDEROUT + BORDERIN and x <= WIDTH - BORDEROUT and y >= BORDEROUT + BORDERIN and y <= HEIGHT - FOOTER:  
 row, col = get\_row\_col\_from\_mouse((x - BORDEROUT - BORDERIN, y - BORDEROUT - BORDERIN))  
  
 if (x - BORDEROUT - BORDERIN - col \* (SQUARE\_SIZE + BORDERIN) < SQUARE\_SIZE) and (  
 y - BORDEROUT - BORDERIN - row \* (SQUARE\_SIZE + BORDERIN) < SQUARE\_SIZE):  
 return (True, row, col)  
 return (False, 0, 0)  
  
  
def get\_row\_col\_from\_mouse(pos):  
 x, y = pos  
 row = y // (SQUARE\_SIZE + BORDERIN)  
 col = x // (SQUARE\_SIZE + BORDERIN)  
 return row, col  
  
  
def main():  
 run = True  
 clock = pygame.time.Clock()  
 game = Game(WIN)  
  
 while run:  
 clock.tick(FPS)  
  
 winner = game.winner()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 run = False  
  
 if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:  
 pos = pygame.mouse.get\_pos()  
 on, row, col = pos\_on\_board(pos)  
 if on:  
 if not winner:  
 game.select(row, col)  
 elif pos\_on\_btn(pos):  
 game.start()  
  
 game.update()  
  
 pygame.quit()  
  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

Код файла constants.py

ROWS, COLS = 4, 4  
SQUARE\_SIZE = 100  
BORDEROUT = 20  
BORDERIN = 2  
  
BTNWIDTH = 100  
BTNHEIGHT = 50  
FOOTER = BTNHEIGHT + BORDEROUT\*2  
WIDTH, HEIGHT = ROWS \* SQUARE\_SIZE + BORDEROUT \* 2 + BORDERIN \* (  
 ROWS + 1), COLS \* SQUARE\_SIZE + BORDEROUT + BORDERIN \* (COLS + 1) + FOOTER  
*# rgb*RED = (255, 0, 0)  
WHITE = (255, 255, 255)  
BLACK = (0, 0, 0)  
BLUE = (0, 0, 255)  
GREY = (128, 128, 128)  
  
DARKGREEN = (38, 70, 83)  
GREEN = (42, 157, 143)  
YELLOW = (233, 196, 106)  
ORANGE = (244, 162, 97)  
TOMATO = (231, 111, 81)

Код файла game.py

import pygame  
from .board import Board  
import time  
  
  
class Game:  
 def \_\_init\_\_(self, win):  
 self.\_init()  
 self.win = win  
 self.run = False  
  
 def \_init(self):  
 self.board = Board()  
 self.win\_flag = False  
  
 def select(self, row, col):  
 self.board.move(row, col)  
  
 def start(self):  
 if self.run:  
 self.\_init()  
 self.run = True  
 self.start\_time = time.time()  
 self.board.change\_btn\_text(**"Заново"**)  
  
 def update(self):  
 self.board.draw\_board(self.win)  
 if self.win\_flag:  
 self.board.draw\_win(self.win)  
 elif self.run:  
 self.board.draw(self.win)  
 time\_diff = round(time.time() - self.start\_time)  
 if time\_diff >= 10000:  
 self.start\_time = time\_diff = 0  
 self.board.change\_time(time\_diff)  
  
 pygame.display.update()  
  
 def winner(self):  
 self.win\_flag = self.board.winner()  
 if (self.win\_flag):  
 self.board.change\_btn\_text(**"Начать"**)  
 return self.win\_flag  
  
 def reset(self):  
 self.\_init()

Код файла board.py

import pygame  
from .constants import ROWS, SQUARE\_SIZE, COLS, BTNHEIGHT, BTNWIDTH, GREEN, ORANGE, TOMATO, BORDERIN, \  
 BORDEROUT, WIDTH, HEIGHT, FOOTER, DARKGREEN  
from .piece import Piece  
import random  
  
  
class Board:  
 def \_\_init\_\_(self):  
 self.board = []  
 self.font = None  
 self.font\_size = 32  
 self.btn\_text = **"Начать"** self.time = 0  
 self.move\_count = 0  
 self.create\_board()  
  
 def change\_btn\_text(self, text):  
 self.btn\_text = text  
  
 def change\_time(self, value):  
 self.time = value  
  
 def draw\_win(self, win):  
 pygame.draw.rect(win, TOMATO, (  
 BORDEROUT, BORDEROUT, WIDTH - BORDEROUT \* 2, HEIGHT - BORDEROUT - FOOTER))  
 f = pygame.font.Font(self.font, self.font\_size \* 2)  
 win\_text = f.render(**"Вы выиграли!"**, 1, DARKGREEN)  
 win.blit(win\_text,  
 win\_text.get\_rect(center=(WIDTH // 2, BORDEROUT + (HEIGHT - BORDEROUT - FOOTER) // 2)))  
  
 def draw\_board(self, win):  
 win.fill(ORANGE)  
 pygame.draw.rect(win, TOMATO, (  
 BORDEROUT, BORDEROUT, WIDTH - BORDEROUT \* 2, HEIGHT - BORDEROUT - FOOTER))  
  
 for row in range(ROWS):  
 for col in range(COLS):  
 pygame.draw.rect(win, GREEN, (  
 BORDEROUT + col \* SQUARE\_SIZE + (col + 1) \* BORDERIN,  
 BORDEROUT + row \* SQUARE\_SIZE + (row + 1) \* BORDERIN, SQUARE\_SIZE,  
 SQUARE\_SIZE))  
  
 f = pygame.font.Font(self.font, self.font\_size)  
 *# draw btn play* pygame.draw.rect(win, TOMATO, (BORDEROUT, HEIGHT - BORDEROUT - BTNHEIGHT, BTNWIDTH, BTNHEIGHT))  
  
 btn\_text = f.render(self.btn\_text, 1, GREEN)  
 win.blit(btn\_text,  
 btn\_text.get\_rect(center=(BORDEROUT + BTNWIDTH // 2, HEIGHT - FOOTER + BORDEROUT + BTNHEIGHT // 2)))  
  
 *# draw move count* move\_text = f.render(**f"Ходы:** {self.move\_count}**"**, 1, GREEN)  
 win.blit(move\_text, move\_text.get\_rect(  
 center=(WIDTH - BORDEROUT - move\_text.get\_rect().width // 2, HEIGHT - FOOTER + BORDEROUT + BTNHEIGHT // 2)))  
  
 *# draw time* time\_text = f.render(**f"Время:** {self.time}**"**, 1, GREEN)  
 win.blit(time\_text, time\_text.get\_rect(center=(  
 BORDEROUT + BTNWIDTH + (WIDTH - BORDEROUT - BTNWIDTH - move\_text.get\_rect().width) // 2,  
 HEIGHT - FOOTER + BORDEROUT + BTNHEIGHT // 2)))  
  
 def getTopBox(self, box):  
 if (box.row == 0): return None  
 return self.board[box.row - 1][box.col]  
  
 def getBottomBox(self, box):  
 if (box.row == COLS - 1): return None  
 return self.board[box.row + 1][box.col]  
  
 def getLeftBox(self, box):  
 if (box.col == 0): return None  
 return self.board[box.row][box.col - 1]  
  
 def getRightBox(self, box):  
 if (box.col == COLS - 1): return None  
 return self.board[box.row][box.col + 1]  
  
 def getNextBoxes(self, box):  
 return list(filter(lambda x: x != None,  
 [self.getTopBox(box), self.getLeftBox(box), self.getBottomBox(box), self.getRightBox(box)]))  
  
 def getRandomNextBox(self, box):  
 next = self.getNextBoxes(box)  
 return next[random.randint(0, len(next) - 1)]  
  
 def swapBoxes(self, box1, box2):  
 box1.value, box2.value = box2.value, box1.value  
  
 def create\_board(self):  
  
 for row in range(ROWS):  
 self.board.append([])  
 for col in range(COLS):  
 self.board[row].append(Piece(row, col, (row \* ROWS + col + 1) % 16))  
  
 *# Shuffle* while True:  
 blankBox = self.board[-1][-1]  
 for i in range(1000):  
 nextBox = self.getRandomNextBox(blankBox)  
 self.swapBoxes(blankBox, nextBox)  
 blankBox = nextBox  
 if not self.winner():  
 break  
  
 def move(self, row, col):  
 next = self.getNextBoxes(self.board[row][col])  
 blankBox = [x for x in next if x.value == 0]  
 if blankBox:  
 self.swapBoxes(self.board[row][col], blankBox[0])  
 if self.move\_count >= 1000:  
 self.move\_count = 0  
 self.move\_count += 1  
  
 def draw(self, win):  
 for row in range(ROWS):  
 for col in range(COLS):  
 self.board[row][col].draw(win)  
  
 def winner(self):  
 for row in range(ROWS):  
 for col in range(COLS):  
 if self.board[row][col].value != (row \* ROWS + col + 1) % 16:  
 return False  
 return True

Код файла piece.py

from .constants import SQUARE\_SIZE, GREEN, BORDERIN, BORDEROUT, DARKGREEN  
import pygame  
  
  
class Piece:  
 def \_\_init\_\_(self, row, col, value):  
 self.row = row  
 self.col = col  
 self.value = value  
  
 def draw(self, win):  
 if self.value != 0:  
 pygame.draw.rect(win, GREEN, (  
 BORDEROUT + self.col \* SQUARE\_SIZE + (self.col + 1) \* BORDERIN,  
 BORDEROUT + self.row \* SQUARE\_SIZE + (self.row + 1) \* BORDERIN,  
 SQUARE\_SIZE,  
 SQUARE\_SIZE))  
 f = pygame.font.Font(None, 48)  
 box\_text = f.render(str(self.value), 1, DARKGREEN)  
 win.blit(box\_text, box\_text.get\_rect(  
 center=(BORDEROUT + self.col \* SQUARE\_SIZE + (self.col + 1) \* BORDERIN + (SQUARE\_SIZE // 2),  
 BORDEROUT + self.row \* SQUARE\_SIZE + (self.row + 1) \* BORDERIN + (SQUARE\_SIZE // 2))))

Результат выполнения программы

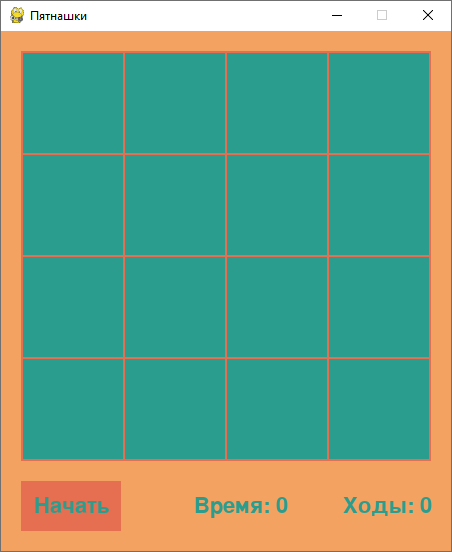


Рис.1 – Стартовый экран игры

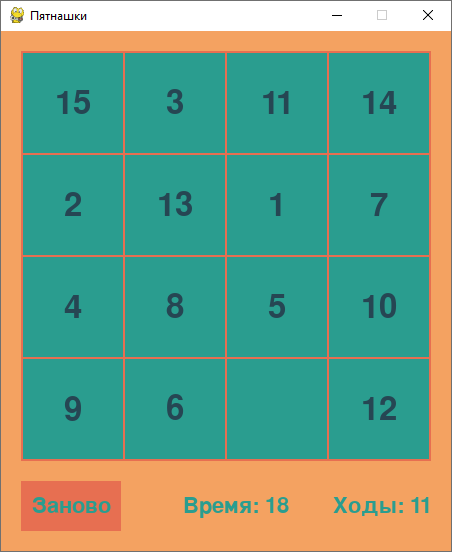


Рис.2 – Игровой интерфейс

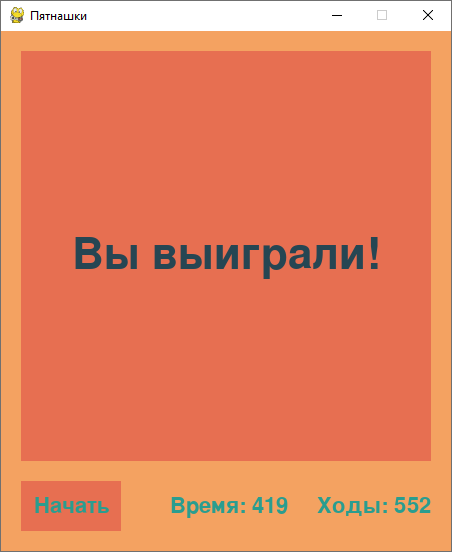


Рис.3 – Итоговый экран