Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по Домашнему заданию «Разработка игры на Pygame»

Выполнил:
студент группы ИУ5-34Б
Волков К.И.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2024 г.

Описание задания

Разработать игру на языке Python с использованием библиотеки Pygame. Для реализации проекта была выбрана игра Pacman.

Цели:

1. Создать графический интерфейс
2. Создать игру в стиле Pacman

Текст программы

Main.py

```
import pygame
import pygame_menu
import random
pygame.font.init()


class Game:
    def __init__(self):
        self.is_on = True
        self.music_is_on = -1

    def game_over(self):
        if field.pacman.lives == 0:
            field.pacman.records(field.pacman.score)
            self.is_on = False

    def music(self):
        pygame.mixer.music.load("tetris.mp3")
        pygame.mixer.music.play(-1)

    def start_game(self):
        clock = pygame.time.Clock()
        while self.is_on:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.is_on = False
                field.pacman.process_events(event)
            sc.fill((255, 255, 255))
            field.pacman.records(field.pacman.score)
            field.pacman.pacman_process_movements()
            field.ghost.ghost_process_movements()
            field.draw(sc)
            pygame.display.flip()
            clock.tick(5)
            self.game_over()


class Menu:
    def __init__(self):
        self.game = Game()
        self.is_over = 0

    def start_the_game(self):
        self.game.start_game()

    def menu(self):
        surface = pygame.display.set_mode((field.width * 30,
                                           field.height * 30))
```

```
        menu = pygame_menu.Menu('Pacman', field.width * 30,
                                field.height * 30,
                                  theme=pygame_menu.themes.THEME_DARK)
        menu.add.text_input('Name :', default='John Doe')
        menu.add.button('Play', self.start_the_game)
        menu.add.button('Quit', pygame_menu.events.EXIT)
        menu.mainloop(surface)


class Pacman:
    def __init__(self, field, x, y):
        self.moving_way = "None"
        self.px = x
        self.py = y
        self.last_known_pos_x = 0
        self.last_known_pos_y = 0
        self.field = field
        self.score = 0
        self.best_score = 0
        self.lives = 3

    def records(self, score):
        if score >= self.best_score:
            self.best_score = score
        else:
            self.best_score = self.best_score

    def restart(self):
        self.field[self.py][self.px] = 0
        self.px = 8
        self.py = 10

    def process_events(self, event):
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_w:
                self.moving_way = "Up"
            if event.key == pygame.K_s:
                self.moving_way = "Down"
            if event.key == pygame.K_d:
                self.moving_way = "Right"
            if event.key == pygame.K_a:
                self.moving_way = "Left"

    def pacman_process_movements(self):
        if self.moving_way == "Up":
            if self.field[self.py - 1][self.px] != 1:
                if self.field[self.py - 1][self.px] == 2:
                    self.field[self.py][self.px] = 0
                    self.py = self.py - 1
                    self.score += 1
                    self.field[self.py][self.px] = 4
                else:
                    self.field[self.py][self.px] = 0
                    self.py = self.py - 1
                    self.field[self.py][self.px] = 4
            else:
                self.moving_way = "None"
        if self.moving_way == "Down":
            if self.field[self.py + 1][self.px] != 1:
                if self.field[self.py + 1][self.px] == 2:
                    self.field[self.py][self.px] = 0
                    self.py = self.py + 1
                    self.score += 1
                    self.field[self.py][self.px] = 4
```

```
                else:
                    self.field[self.py][self.px] = 0
                    self.py = self.py + 1
                    self.field[self.py][self.px] = 4
            else:
                self.moving_way = "None"
        if self.moving_way == "Right":
            if self.field[self.py][self.px + 1] != 1:
                if self.field[self.py][self.px + 1] == 2:
                    self.field[self.py][self.px] = 0
                    self.px = self.px + 1
                    self.score += 1
                    self.field[self.py][self.px] = 4
                else:
                    self.field[self.py][self.px] = 0
                    self.px = self.px + 1
                    self.field[self.py][self.px] = 4
            else:
                self.moving_way = "None"
        if self.moving_way == "Left":
            if self.field[self.py][self.px - 1] != 1:
                if self.field[self.py][self.px - 1] == 2:
                    self.field[self.py][self.px] = 0
                    self.px = self.px - 1
                    self.score += 1
                    self.field[self.py][self.px] = 4
                else:
                    self.field[self.py][self.px] = 0
                    self.px = self.px - 1
                    self.field[self.py][self.px] = 4
            else:
                self.moving_way = "None"


class Ghost:
    def __init__(self, field, x, y, pacman):
        self.dirs = {"up" : False,
                     "down" : False,
                     "left" : False,
                     "right" : False}
        self.field = field
        self.moving_way = {"up": False,
                     "down": False,
                     "left": False,
                     "right": False}
        self.gx = x
        self.gy = y
        self.pacman = pacman

    def restart(self):
        self.field[self.gy][self.gx] = 0
        self.gx = 9
        self.gy = 20

    def ghost_possible_dirs(self):
        if self.field[self.gy - 1][self.gx] != 1:
            self.dirs["up"] = True
        else:
            self.dirs["up"] = False
        if self.field[self.gy + 1][self.gx] != 1:
            self.dirs["down"] = True
        else:
            self.dirs["down"] = False
        if self.field[self.gy][self.gx - 1] != 1:
```

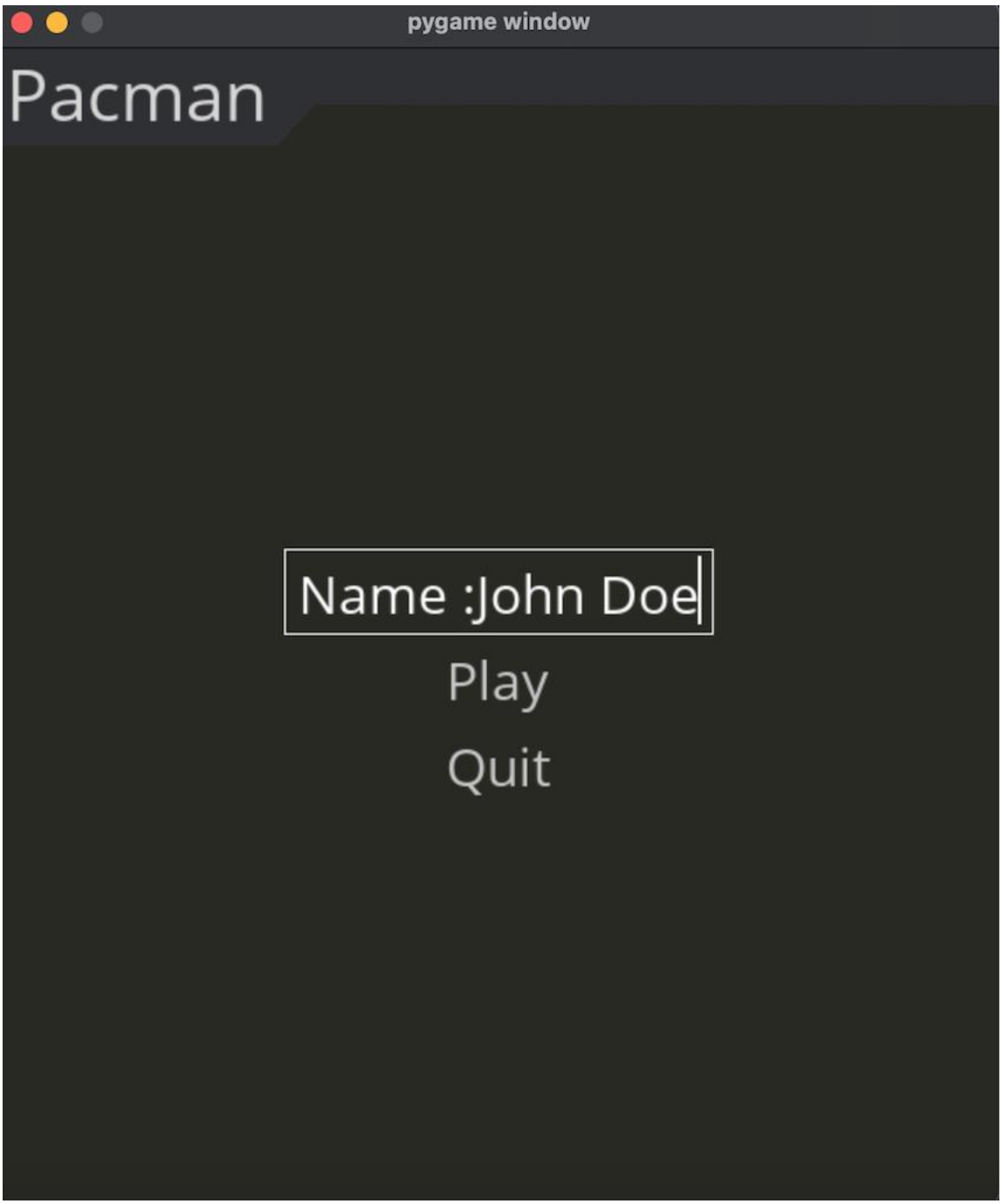
```
            self.dirs["left"] = True
        else:
            self.dirs["left"] = False
        if self.field[self.gy][self.gx + 1] != 1:
            self.dirs["right"] = True
        else:
            self.dirs["right"] = False

    def guess(self):
        self.ghost_possible_dirs()
        a = []
        if self.dirs["up"]:
            min_dir_u = round(((self.pacman.px - self.gx)**2 +
(self.pacman.py - (self.gy - 1))**2)**0.5)
            a.append(min_dir_u)
        else:
            min_dir_u = 999
            a.append(min_dir_u)
        if self.dirs["down"]:
            min_dir_d = round(((self.pacman.px - self.gx) ** 2 +
(self.pacman.py - (self.gy + 1)) ** 2)**0.5)
            a.append(min_dir_d)
        else:
            min_dir_d = 999
            a.append(min_dir_d)
        if self.dirs["left"]:
            min_dir_l = round(((self.pacman.px - (self.gx - 1)) ** 2 +
(self.pacman.py - self.gy) ** 2)**0.5)
            a.append(min_dir_l)
        else:
            min_dir_l = 999
            a.append(min_dir_l)
        if self.dirs["right"]:
            min_dir_r = round(((self.pacman.px - (self.gx + 1)) ** 2 +
(self.pacman.py - self.gy) ** 2)**0.5)
            a.append(min_dir_r)
        else:
            min_dir_r = 999
            a.append(min_dir_r)

        print(a)
        print(min(a))
        dir = min(a)
        rand_plus = random.randint(1, 2)

        if dir == min_dir_r and dir == min_dir_l:
            if rand_plus == 1:
                min_dir_r += 1
            else:
                min_dir_l += 1
        if dir == min_dir_u and dir == min_dir_d:
            if rand_plus == 1:
                min_dir_u += 1
            else:
                min_dir_d += 1

        if dir == min_dir_u:
            self.moving_way["up"] = True
        else:
            self.moving_way["up"] = False
        if dir == min_dir_d:
            self.moving_way["down"] = True
        else:
            self.moving_way["down"] = False
```

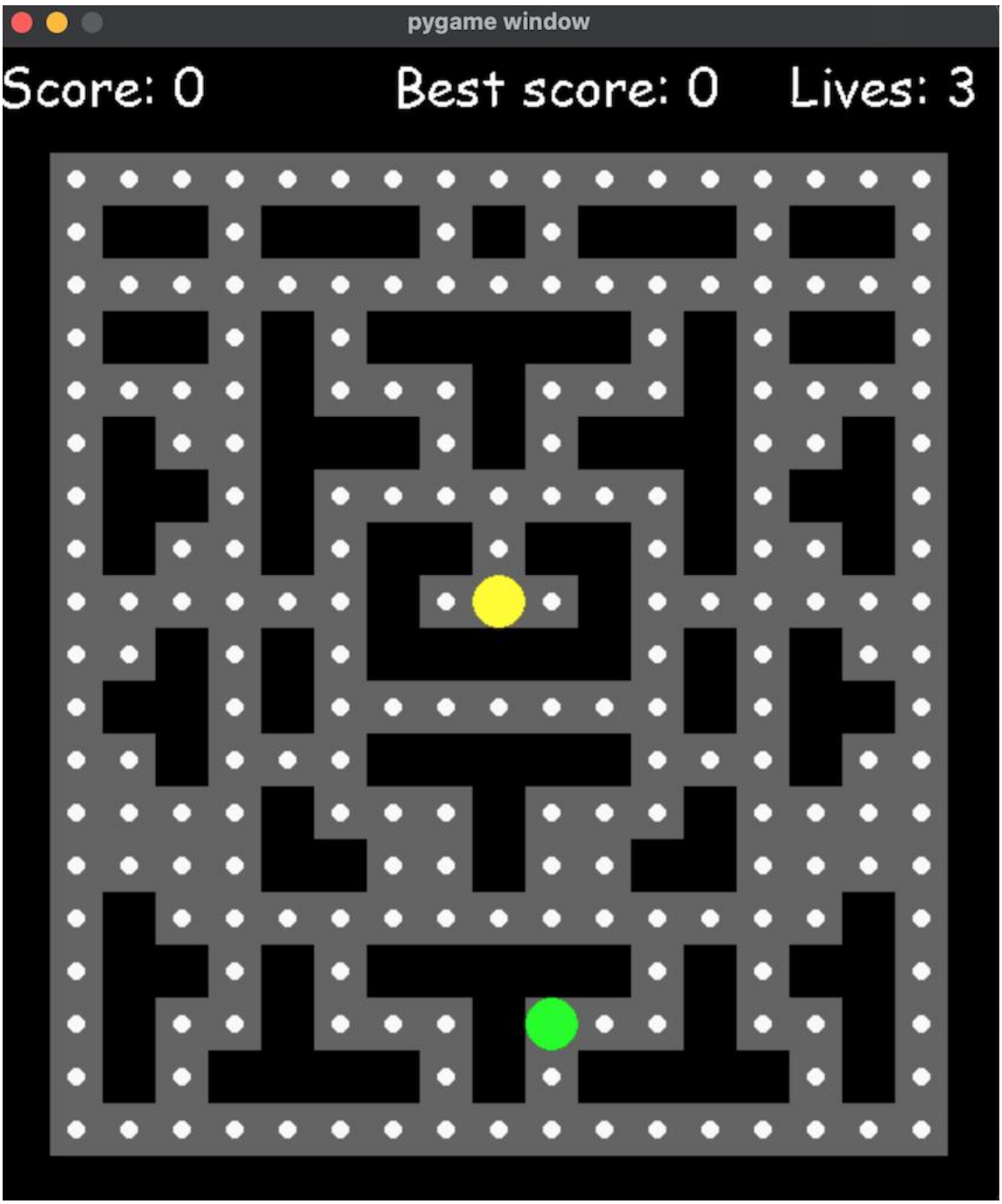
```
        if dir == min_dir_l:
            self.moving_way["left"] = True
        else:
            self.moving_way["left"] = False
        if dir == min_dir_r:
            self.moving_way["right"] = True
        else:
            self.moving_way["right"] = False

    def ghost_process_movements(self):
        # Ghost movement
        self.guess()
        if self.moving_way["up"]:
            if self.field[self.gy - 1][self.gx] != 1:
                if self.field[self.gy - 1][self.gx] == 2:
                    self.field[self.gy][self.gx] = 2
                    self.gy = self.gy - 1
                    self.field[self.gy][self.gx] = 3
                elif self.field[self.gy - 1][self.gx] == 4:
                    self.field[self.gy][self.gx] = 0
                    self.pacman.lives -= 1
                    self.pacman.restart()
                    self.restart()
                else:
                    self.field[self.gy][self.gx] = 0
                    self.gy = self.gy - 1
                    self.field[self.gy][self.gx] = 3
            else:
                self.moving_way["up"] = False
        if self.moving_way["down"]:
            if self.field[self.gy + 1][self.gx] != 1:
                if self.field[self.gy + 1][self.gx] == 2:
                    self.field[self.gy][self.gx] = 2
                    self.gy = self.gy + 1
                    self.field[self.gy][self.gx] = 3
                elif self.field[self.gy + 1][self.gx] == 4:
                    self.field[self.gy][self.gx] = 0
                    self.pacman.lives -= 1
                    self.pacman.restart()
                    self.restart()
                else:
                    self.field[self.gy][self.gx] = 0
                    self.gy = self.gy + 1
                    self.field[self.gy][self.gx] = 3
            else:
                self.moving_way["down"] = False
        if self.moving_way["right"]:
            if self.field[self.gy][self.gx + 1] != 1:
                if self.field[self.gy][self.gx + 1] == 2:
                    self.field[self.gy][self.gx] = 2
                    self.gx = self.gx + 1
                    self.field[self.gy][self.gx] = 3
                elif self.field[self.gy][self.gx + 1] == 4:
                    self.field[self.gy][self.gx] = 0
                    self.pacman.lives -= 1
                    self.pacman.restart()
                    self.restart()
                else:
                    self.field[self.gy][self.gx] = 0
                    self.gx = self.gx + 1
                    self.field[self.gy][self.gx] = 3
            else:
                self.moving_way["right"] = False
        if self.moving_way["left"]:
```

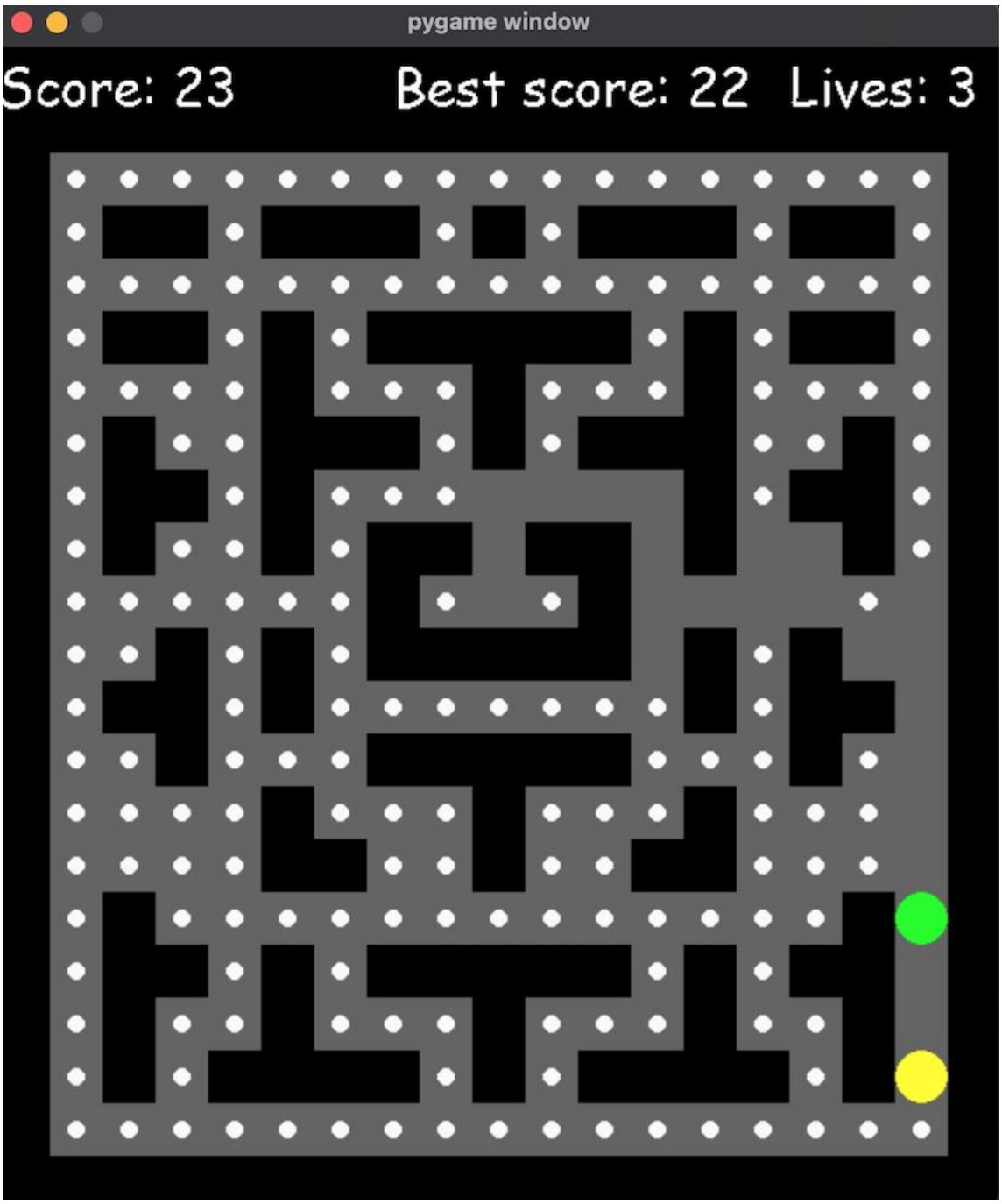
```
            if self.field[self.gy][self.gx - 1] != 1:
                if self.field[self.gy][self.gx - 1] == 2:
                    self.field[self.gy][self.gx] = 2
                    self.gx = self.gx - 1
                    self.field[self.gy][self.gx] = 3
                elif self.field[self.gy][self.gx - 1] == 4:
                    self.field[self.gy][self.gx] = 0
                    self.pacman.lives -= 1
                    self.pacman.restart()
                    self.restart()
                else:
                    self.field[self.gy][self.gx] = 0
                    self.gx = self.gx - 1
                    self.field[self.gy][self.gx] = 3
            else:
                self.moving_way["left"] = False


class Field:
    def __init__(self, field):
        file = open(field)
        lines = file.readlines()
        self.field = []
        for line in lines:
            lst = list(map(int, line.split(" ")))
            self.field.append(lst)
        print(self.field)
        self.height = len(self.field)
        self.width = len(self.field[0])
        for y in range(self.height):
            for x in range(self.width):
                if self.field[y][x] == 4:
                    self.pacman = Pacman(self.field, x, y)
                if self.field[y][x] == 3:
                    self.ghost = Ghost(self.field, x, y, self.pacman)

    def draw(self, sc):
        aqua = (0, 255, 255)  # морская волна
        black = (0, 0, 0)  # черный
        blue = (0, 0, 255)  # синий
        fuchsia = (255, 0, 255)  # фуксия
        gray = (128, 128, 128)  # серый
        green = (0, 128, 0)  # зеленый
        lime = (0, 255, 0)  # цвет лайма
        maroon = (128, 0, 0)  # темно-бордовый
        navy_blue = (0, 0, 128)  # темно-синий
        olive = (128, 128, 0)  # оливковый
        purple = (128, 0, 128)  # фиолетовый
        red = (255, 0, 0)  # красный
        silver = (192, 192, 192)  # серебряный
        teal = (0, 128, 128)  # зелено-голубой
        white = (255, 255, 255)  # белый
        yellow = (255, 255, 0)  # желтый
        for y in range(self.height):
            for x in range(self.width):
                if self.field[y][x] == 0:
                    color = (100, 100, 100)
                    pygame.draw.rect(sc, color,
                                     (x * 30, y * 30, 30, 30))
                if self.field[y][x] == 1:
                    color = (0, 0, 0)
                    pygame.draw.rect(sc, color,
                                     (x * 30, y * 30, 30, 30))
                if self.field[y][x] == 4:
```

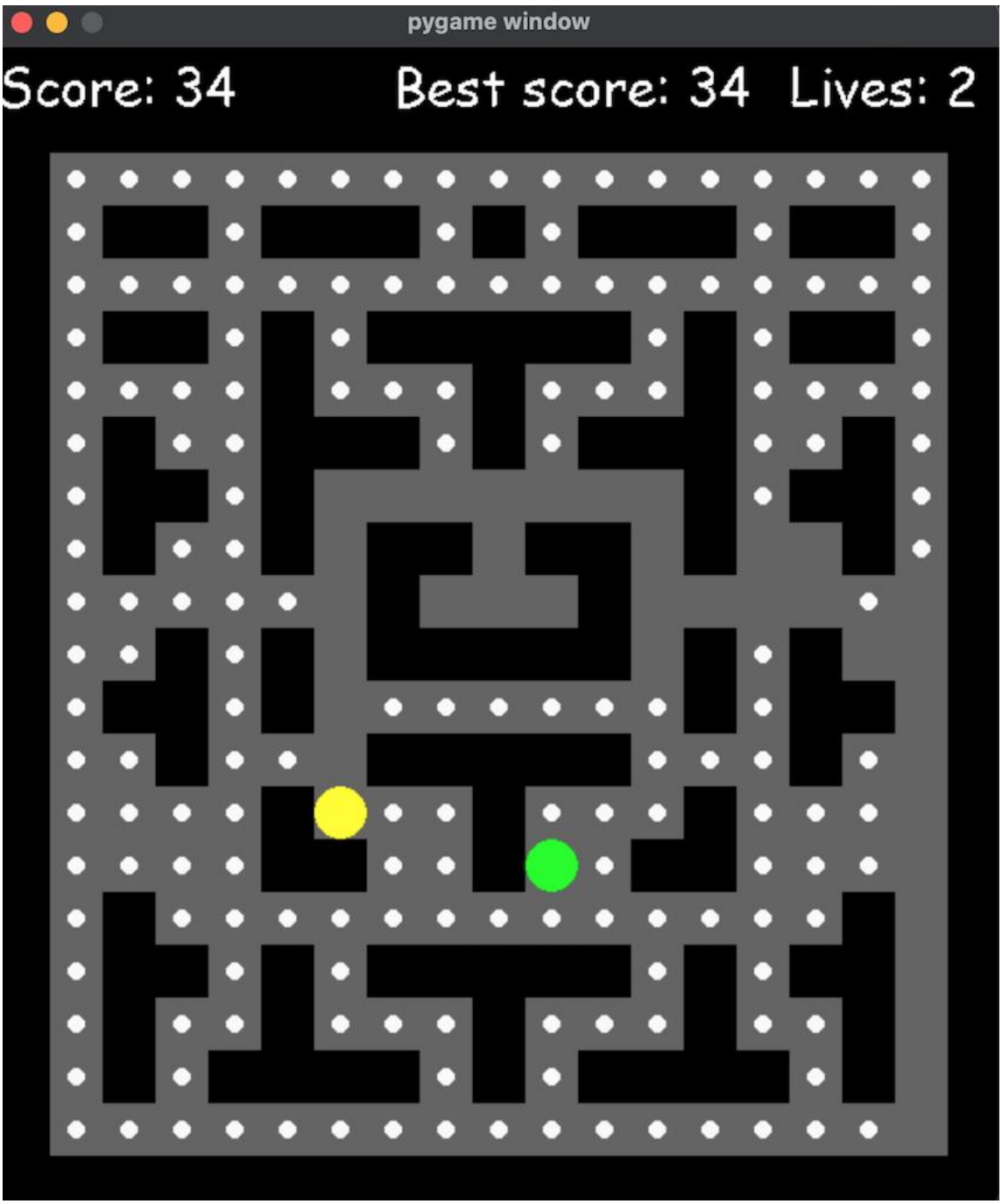
```
                    color = (100, 100, 100)
                    pygame.draw.rect(sc, color,
                                     (x * 30, y * 30, 30, 30))
                    color = (255, 255, 0)
                    pygame.draw.circle(sc, color,
                                       (x * 30 + 15, y * 30 + 15), 15)
                if self.field[y][x] == 2:
                    color = (100, 100, 100)
                    pygame.draw.rect(sc, color,
                                     (x * 30, y * 30, 30, 30))
                    color = (250, 250, 250)
                    pygame.draw.circle(sc, color,
                                       (x * 30 + 15, y * 30 + 15), 5)
                if self.field[y][x] == 3:
                    color = (100, 100, 100)
                    pygame.draw.rect(sc, color,
                                     (x * 30, y * 30, 30, 30))
                    color = (0, 255, 0)
                    pygame.draw.circle(sc, color,
                                       (x * 30 + 15, y * 30 + 15), 15)
                    myfont = pygame.font.SysFont('Comic Sans MS', 30)
                    score_board = myfont.render(f"Score:
{str(self.pacman.score)}", False, (255, 255, 255))
                    sc.blit(score_board, (0, 0))
                    lives_board = myfont.render(f"Lives:
{str(self.pacman.lives)}", False, (255, 255, 255))
                    sc.blit(lives_board, (450, 0))
                    best_score_board = myfont.render(f"Best score:
{str(self.pacman.best_score)}", False, (255, 255, 255))
                    sc.blit(best_score_board, (225, 0))


pygame.init()
field = Field("field.txt")
sc = pygame.display.set_mode((field.width * 30,
                              field.height * 30))
menu = Menu()
menu.menu()
```

Результат программы


pygame window
Pacman
Name :John Doe
Play
Quit


pygame window
Score: 0
Best score: 0
Lives: 3


pygame window
Score: 23
Best score: 22
Lives: 3


pygame window
Score: 34
Best score: 34
Lives: 2