Пояснительная записка

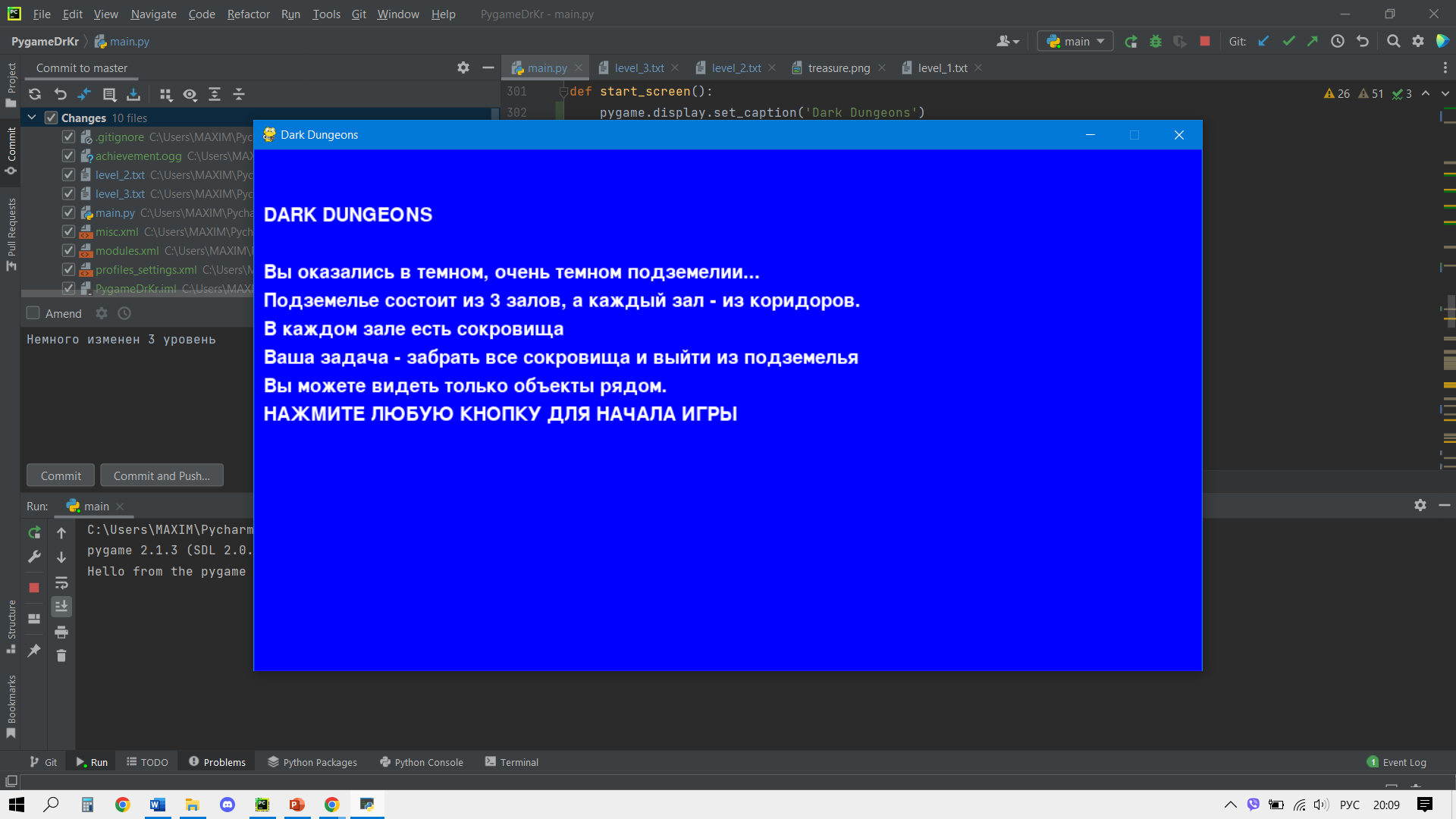
1. Описание

Главная героиня попадает в темное-претемное подземелье. Чтобы выйти из него, она должна пройти все залы, а для этого найти все сокровища и с помощью них открыть дверь в следующий зал или сразу на свободу. Также в каждом зале разбросаны записки, не влияющие на гемплей, но раскрывающий лор(сюжет) игры

2. Функционал

Всего существует 4 основных функции: start\_screen(), run\_level(level\_name), end\_screen() и messages\_screen().

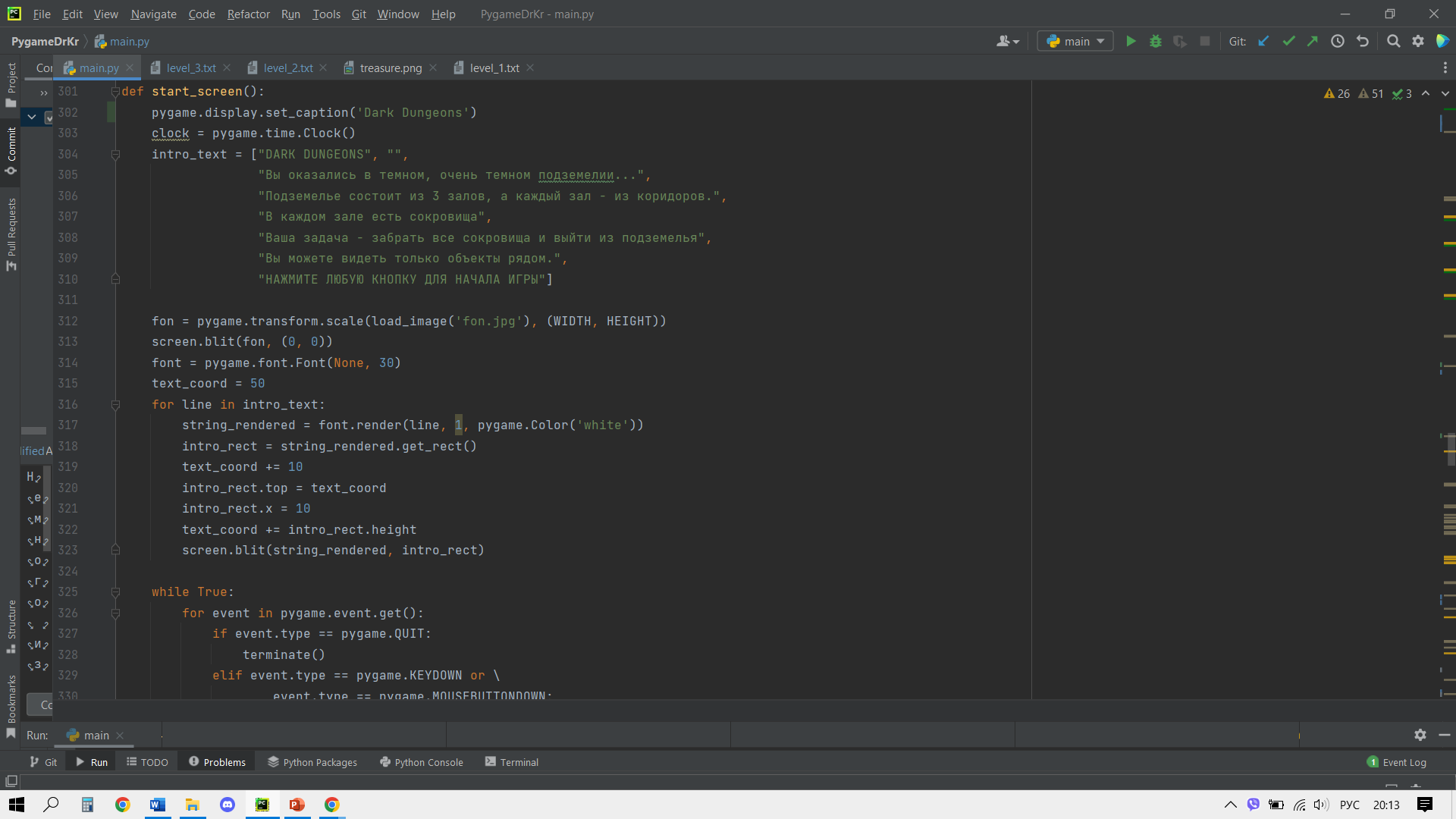
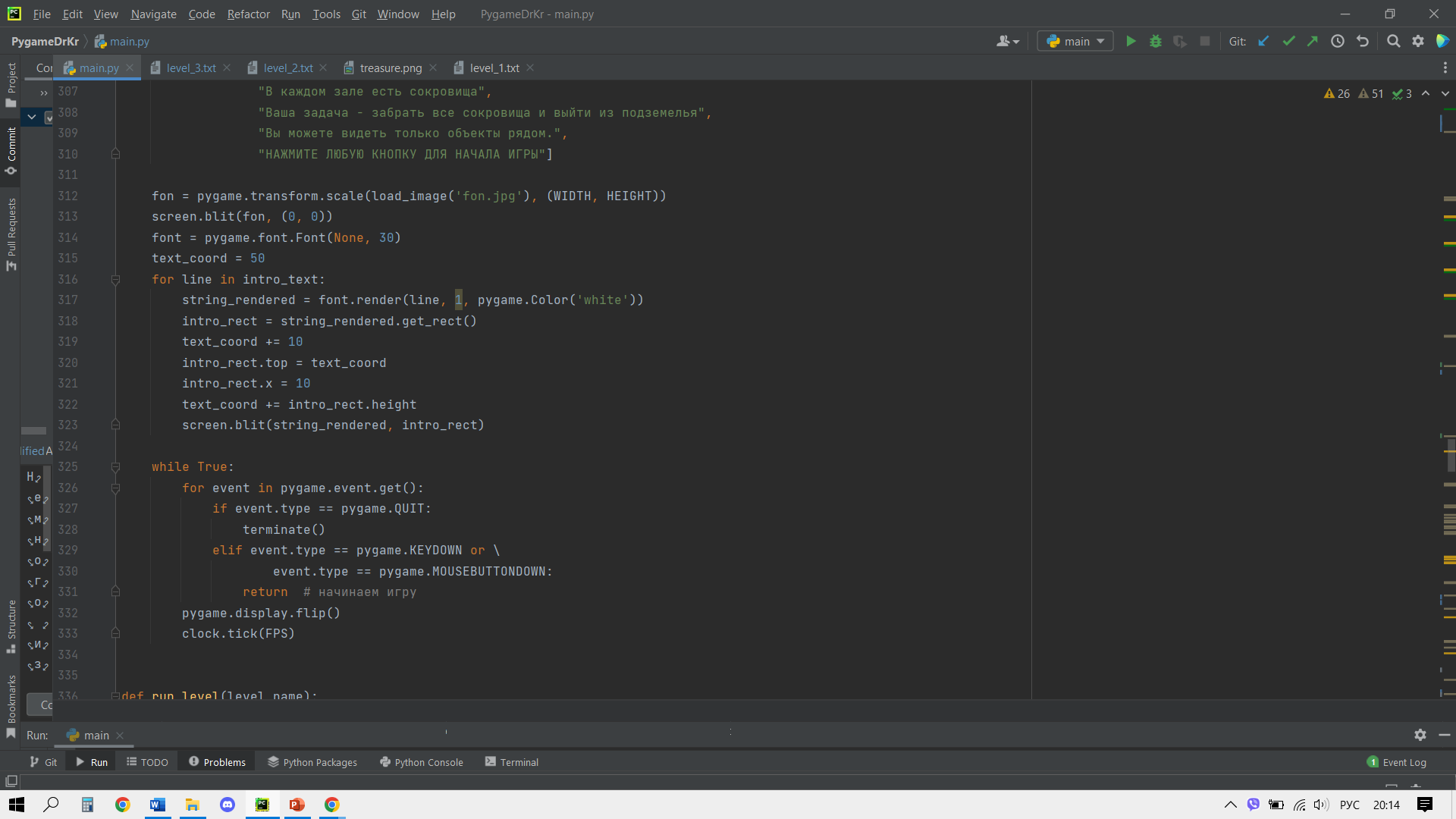
1. start\_screen() – включает заставку игры и рассказыает правила игры.



Заставка игры

Для реализации потребовался фон для игры и текст, отображаемый на экране

Строки кода

Отдельная функция terminate(), которая завершает работу игры.

def terminate():  
 pygame.quit()  
 sys.exit()

2. run\_level(level\_name) – по названию уровня(файл в формате .txt) выстраивает рабочую его версию, где можно управлять игроком

def run\_level(level\_name):  
 global door  
 global screen, size  
 global map\_list  
 global player  
 door = None  
 map\_list = []  
 player = None  
 f = pygame.font.Font(None, 20)  
 level\_complete = False  
 level = load\_level(level\_name)  
 player, level\_x, level\_y, map\_list = generate\_level(level)  
 size = WIDTH, HEIGHT = (level\_x + 1) \* 50, (level\_y + 1) \* 50  
 screen = pygame.display.set\_mode(size)

Этот кусок кода использует глобальные door, screen, size, map\_list и player(в map\_list хранится карта клеток(Tile) и вся информация о них, door и player хранят информацию о выходе из уровня и об играбельном персонаже соответственно, screen и size – для задачи параметров экрана)

Немного об load\_level() и generate\_level()

def load\_level(filename):  
 filename = "data/" + filename  
 # читаем уровень, убирая символы перевода строки  
 with open(filename, 'r') as mapFile:  
 level\_map = [line.strip() for line in mapFile]  
  
 # и подсчитываем максимальную длину  
 max\_width = max(map(len, level\_map))  
  
 # дополняем каждую строку стенами ('#')  
 return list(map(lambda x: x.ljust(max\_width, '#'), level\_map))  
  
  
def generate\_level(level):  
 global door  
 global notes  
 notes = []  
 new\_player, x, y = None, None, None  
 for y in range(len(level)):  
 arr = []  
 for x in range(len(level[y])):  
 if level[y][x] == '.':  
 arr.append(Tile('empty', x, y))  
 arr[x].set\_color()  
 elif level[y][x] == '#':  
 arr.append(Tile('wall', x, y))  
 arr[x].set\_color()  
 elif level[y][x] == '@':  
 arr.append(Tile('empty', x, y))  
 new\_player = Player(x, y)  
 arr[x].set\_image()  
 elif level[y][x] == 'T':  
 arr.append(Tile('empty', x, y))  
 treasures[Treasure(x, y)] = True  
 arr[x].set\_color()  
 elif level[y][x] == 'D':  
 arr.append(Tile('empty', x, y))  
 door = Door(x, y)  
 arr[x].set\_color()  
 elif level[y][x].isdigit():  
 arr.append(Tile('empty', x, y))  
 notes.append(Note(x, y, level[y][x]))  
 arr[x].set\_color()  
 map\_list.append(arr)  
 # вернем игрока, а также размер поля в клетках  
 return new\_player, x, y, map\_list

load\_level() загружает каркас из файла для дальнейшей обработки в generate\_level, который на основе полученного каркаса распределяет типы клеток, а также загружает информацию о других видах объектов. Изначально все клетки становятся черными(метод set\_color())

Следующие строки кода выстраивают начальную картину уровня(изначально все клетки черные)

if player.x - 1 >= 0:  
 map\_list[player.y][player.x - 1].set\_image()  
if player.x + 1 < len(map\_list[0]):  
 map\_list[player.y][player.x + 1].set\_image()  
if player.x + 1 < len(map\_list[0]) and player.y - 1 >= 0:  
 map\_list[player.y - 1][player.x + 1].set\_image()  
if player.x - 1 >= 0 and player.y - 1 >= 0:  
 map\_list[player.y - 1][player.x - 1].set\_image()  
if player.y - 1 >= 0:  
 map\_list[player.y - 1][player.x].set\_image()  
if player.x + 1 < len(map\_list[0]) and player.y + 1 < len(map\_list[0]):  
 map\_list[player.y + 1][player.x + 1].set\_image()  
if player.x - 1 >= 0 and player.y + 1 < len(map\_list[0]):  
 map\_list[player.y + 1][player.x - 1].set\_image()  
if player.y + 1 < len(map\_list[0]):  
 map\_list[player.y + 1][player.x].set\_image()

Далее загружается звуковое сопровождение уровня

pygame.mixer.music.load('data\main\_theme.ogg')  
pygame.mixer.music.play(-1)

Отработка событий(когда игрок передвигается или когда нужно выйти из игры)

while not (level\_complete):  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 terminate()  
 elif event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_UP:  
 player.set\_image(load\_image('mary\_back.png'))  
 level\_complete = player.move\_up()  
 if level\_complete is None:  
 level\_complete = False  
 if event.key == pygame.K\_DOWN:  
 player.set\_image(load\_image('mary\_front.png'))  
 level\_complete = player.move\_down()  
 if level\_complete is None:  
 level\_complete = False  
 if event.key == pygame.K\_RIGHT:  
 player.set\_image(load\_image('Mary\_right.png'))  
 level\_complete = player.move\_right()  
 if level\_complete is None:  
 level\_complete = False  
 if event.key == pygame.K\_LEFT:  
 player.set\_image(load\_image('mary\_left.png'))  
 level\_complete = player.move\_left()  
 if level\_complete is None:  
 level\_complete = False

Вставка текста, информирующего игрока о том, сколько осталось сундуков и какой зал сейчас

text = f.render(f'Осталось сундуков: {list(treasures.values()).count(True)}', True, 'white')  
place = text.get\_rect(  
 center=(100, 100))  
text1 = f.render(f'Зал {level\_name[6]}', True, 'white')  
place1 = text.get\_rect(  
 center=(100, 50))

Далее идет отрисовка объектов. Если объекты вне досягаемости, они убираются методом kill(), иначе добавляются методом add()

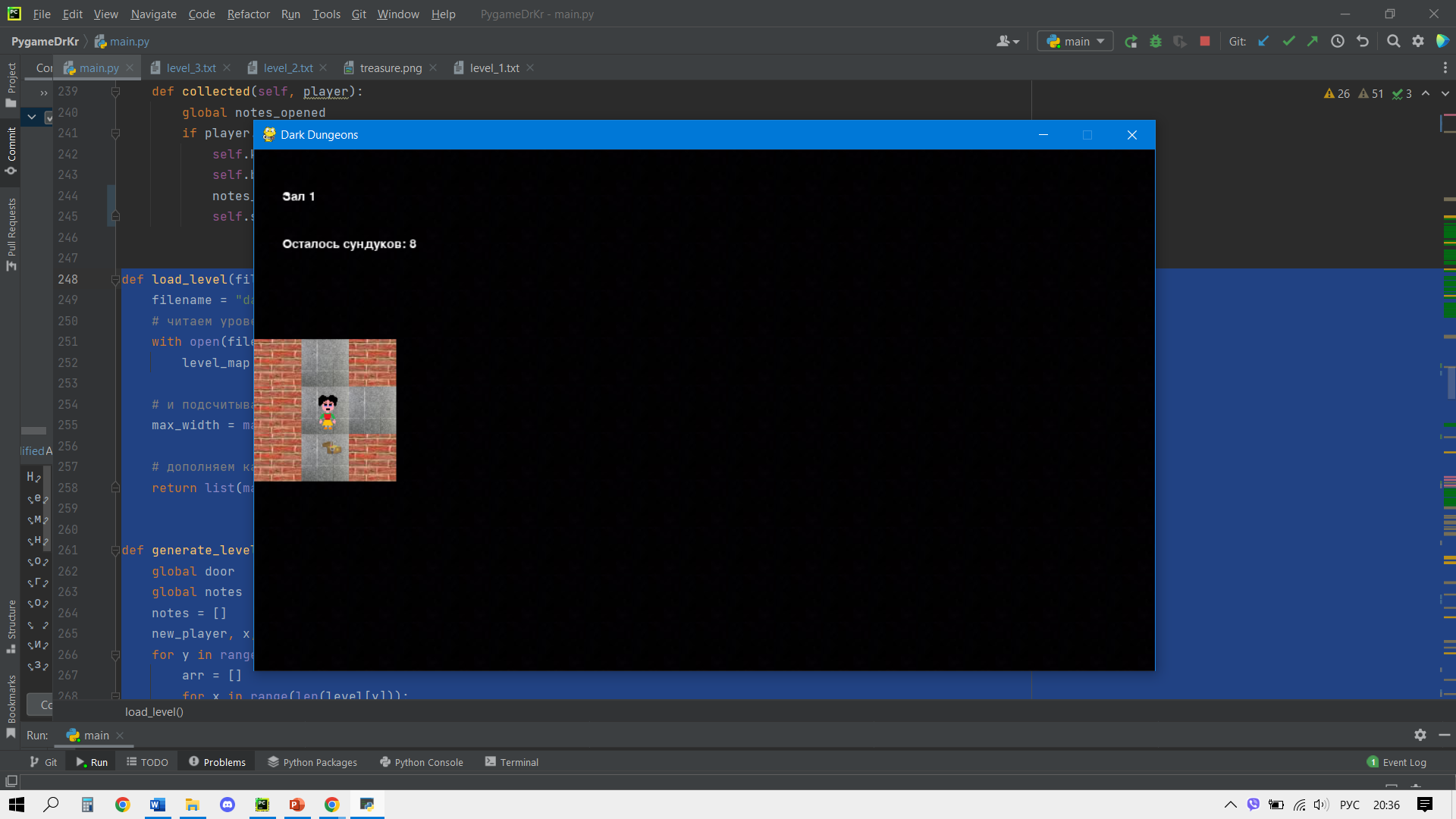
tiles\_group.draw(screen)  
player\_group.draw(screen)  
treasure\_group.draw(screen)  
door\_group.draw(screen)  
note\_group.draw(screen)  
for x in treasures:  
 x.kill()  
 if abs(x.x - player.x) <= 1 and abs(x.y - player.y) <= 1 and treasures[x]:  
 x.add(treasure\_group)  
 if x.x - player.x == 0 and x.y - player.y == 0 and treasures[x]:  
 x.kill()  
 treasures[x] = False  
 pygame.mixer.Sound('data/treasure-achieved.wav').play()  
door.kill()  
if abs(door.x - player.x) <= 1 and abs(door.y - player.y) <= 1 and not (door.bool):  
 door.add(door\_group)  
screen.blit(text, place)  
screen.blit(text1, place1)  
if True not in treasures.values() and not (door.bool):  
 door.open()  
for x in notes:  
 x.kill()  
 if abs(player.x - x.x) <= 1 and abs(player.y - x.y) <= 1 and x.bool:  
 x.add(note\_group)  
 if player.x == x.x and player.y == x.y and x.bool:  
 x.collected(player)  
pygame.display.flip()

clock.tick(fps)

как только уровень закончен, убираются player и treasures. Музыка отключается

pygame.mixer.music.stop()  
player.kill()  
for x in treasures:  
 x.kill()  
return

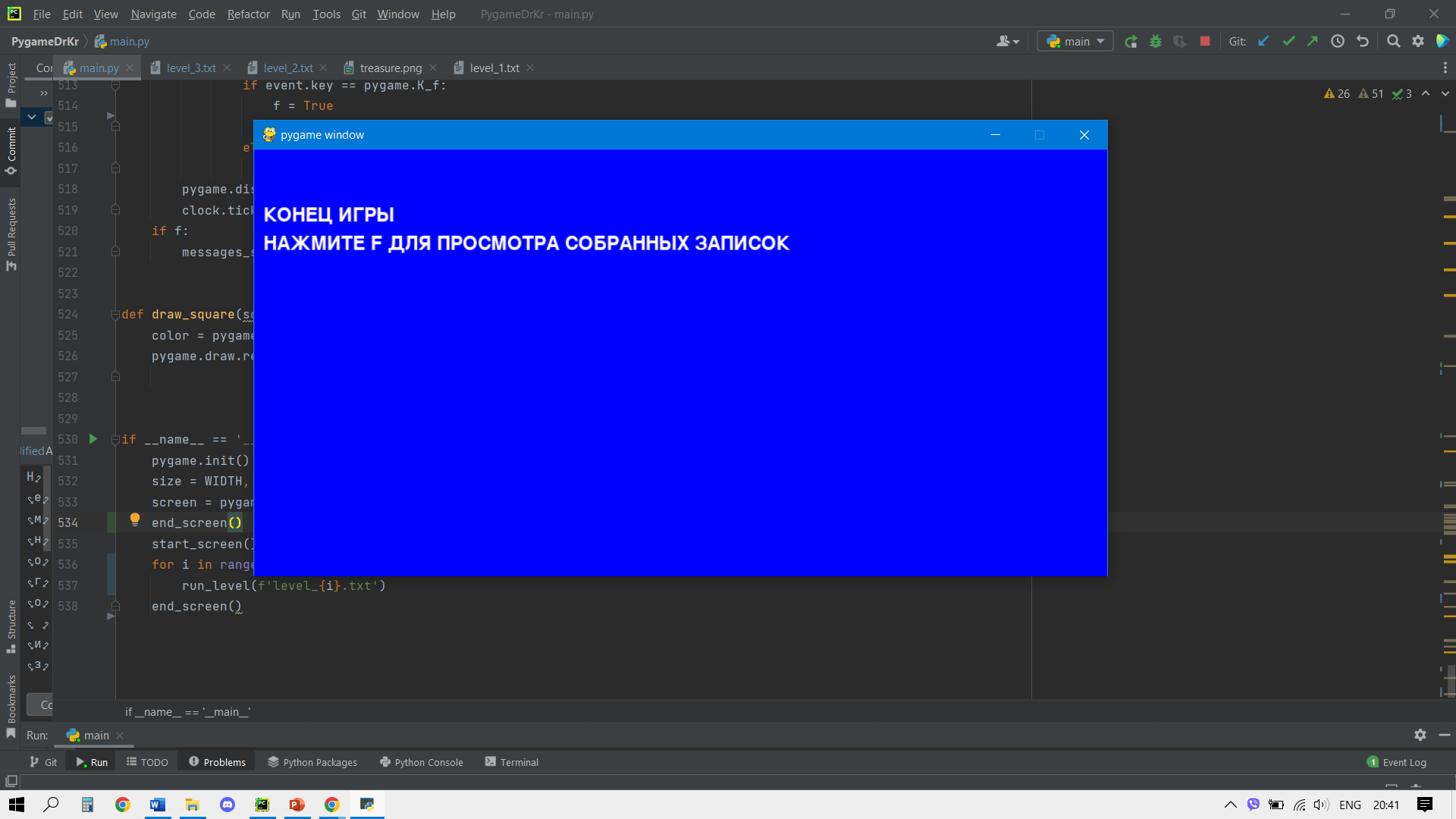
Пример работы кода на примере level\_1.txt:



3. end\_screen() – оповещает о конце игры и о возможности посмотреть оставленные записки

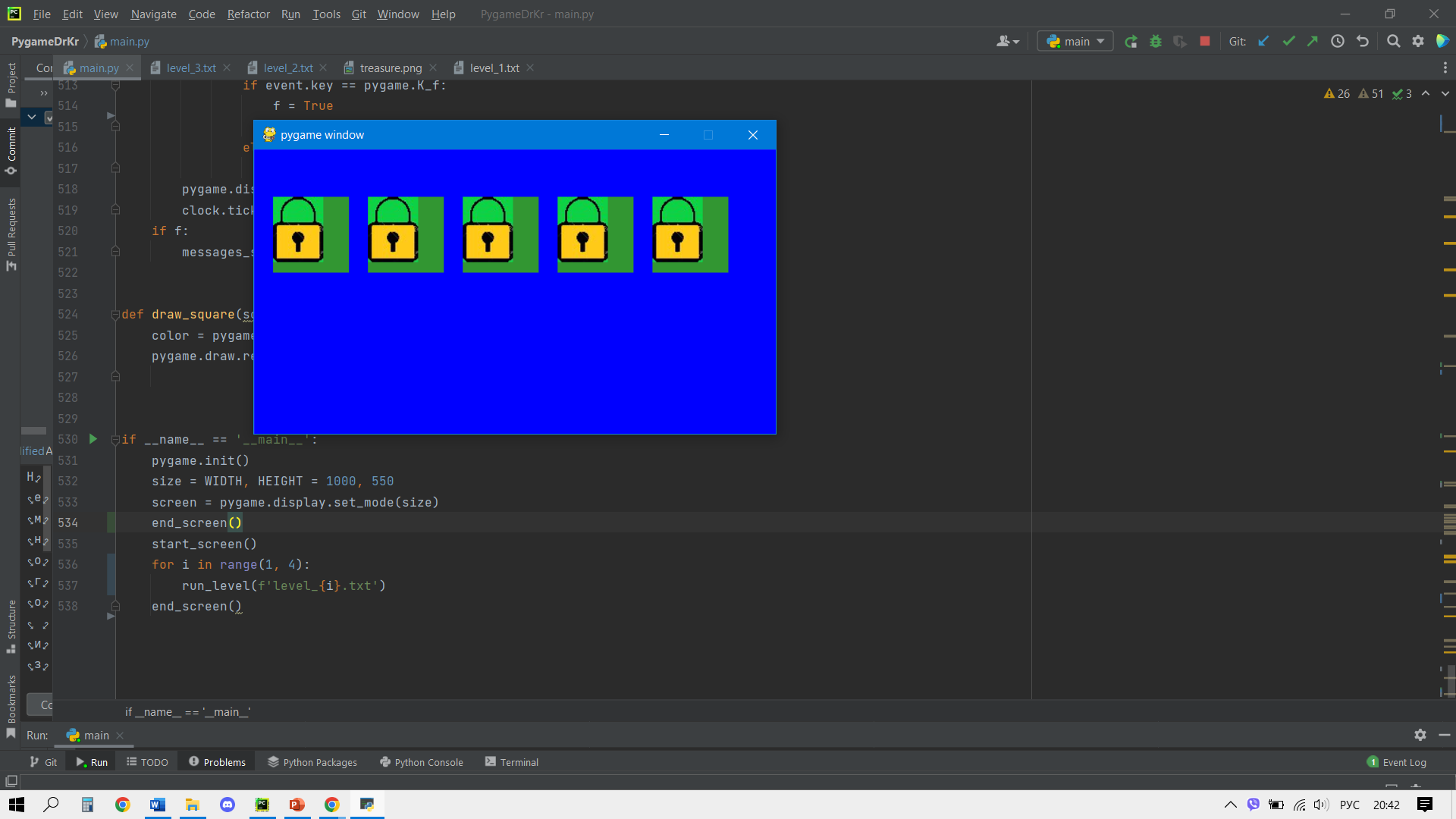
def end\_screen():  
 clock = pygame.time.Clock()  
 size = WIDTH, HEIGHT = 900, 450  
 outro\_text = ["КОНЕЦ ИГРЫ",  
 "НАЖМИТЕ F ДЛЯ ПРОСМОТРА СОБРАННЫХ ЗАПИСОК"]  
 fon = pygame.transform.scale(load\_image('fon.jpg'), (WIDTH, HEIGHT))  
 f = False  
 screen = pygame.display.set\_mode(size)  
 screen.blit(fon, (0, 0))  
 font = pygame.font.Font(None, 30)  
 text\_coord = 50  
 for line in outro\_text:  
 string\_rendered = font.render(line, 1, pygame.Color('white'))  
 outro\_rect = string\_rendered.get\_rect()  
 text\_coord += 10  
 outro\_rect.top = text\_coord  
 outro\_rect.x = 10  
 text\_coord += outro\_rect.height  
 screen.blit(string\_rendered, outro\_rect)  
 running = True  
 while running:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 terminate()  
 elif event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_f:  
 f = True  
 running = False  
 else:  
 return  
 pygame.display.flip()  
 clock.tick(FPS)  
 if f:  
 messages\_screen()

Реализован почти так же, как и start\_screen()



4. messages\_screen.

messages\_screen() хранит информацию обо всех найденных записках. Если записка не найдена – ее будет невозможно прочитать(визуально заблокирована замком)



Код

def messages\_screen():  
 size = WIDTH, HEIGHT = 550, 300  
 screen = pygame.display.set\_mode(size)  
 screen.fill((0, 0, 255))  
 image = load\_image('lock.jpg')  
 running = True  
 coords = [(20, 100), (120, 200), (220, 300), (320, 400), (420, 500)]  
 while running:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 terminate()  
 x, y = pygame.mouse.get\_pos()  
 if event.type == pygame.MOUSEBUTTONDOWN:  
 for i in coords:  
 if i[0] <= x <= i[1] and 50 <= y <= 130 and notes\_opened[str(coords.index(i) + 1)]:  
 message\_screen(str(coords.index(i) + 1), screen)  
  
 for x in range(5):  
 draw\_square(screen, x \* 100, 50)  
 if not (notes\_opened[str(x + 1)]):  
 screen.blit(image, (x \* 100 + 20, 50))  
  
 pygame.display.flip()

Эта функция реализована при помощи 2 – х – message\_screen() и draw\_square()

def message\_screen(n, screen):  
 global note\_dict  
 screen.fill((0, 0, 255))  
 font = pygame.font.Font(None, 30)  
 text\_coord = 50  
 text = note\_dict[n]  
 text.append('НАЖМИТЕ ЛЮБУЮ КНОПКУ ДЛЯ СКРЫТИЯ ЗАПИСКИ')  
 for line in text:  
 string\_rendered = font.render(line, 1, pygame.Color('white'))  
 intro\_rect = string\_rendered.get\_rect()  
 text\_coord += 10  
 intro\_rect.top = text\_coord  
 intro\_rect.x = 10  
 text\_coord += intro\_rect.height  
 screen.blit(string\_rendered, intro\_rect)  
 running = True  
 while running:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 terminate()  
 elif event.type == pygame.KEYDOWN or \  
 event.type == pygame.MOUSEBUTTONDOWN:  
 screen.fill((0, 0, 255))  
 text.pop()  
 return  
 pygame.display.flip()  
 clock.tick(FPS)

def draw\_square(screen, x, y):  
 color = pygame.Color(50, 150, 50)  
 pygame.draw.rect(screen, color,  
 (x + 20, 50, 80, 80), 0)

в функцию main() передаются только start\_screen(), run\_level() и end\_screen()

if \_\_name\_\_ == '\_\_main\_\_':  
 pygame.init()  
 size = WIDTH, HEIGHT = 1000, 550  
 screen = pygame.display.set\_mode(size)  
 start\_screen()  
 for i in range(1, 4):  
 run\_level(f'level\_{i}.txt')  
 end\_screen()

3. Используемые модули и классы

Модули: pygame(используются pygame.mixer, pygame.sprite) – собственно сама игра, os – для загрузки изображений, sys – для загрузки игры в целом и random – для издавания различных звуков при ходьбе игрока. Реализовано 5 классов –

Tile(унаследствован от pygame.sprite.Sprite) – единица уровня

class Tile(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, tile\_type, pos\_x, pos\_y):  
 super().\_\_init\_\_(tiles\_group, all\_sprites)  
 self.tile\_type = tile\_type  
 self.pos\_x = pos\_x  
 self.pos\_y = pos\_y  
 self.image = tile\_images[tile\_type]  
 self.rect = self.image.get\_rect().move(  
 tile\_width \* pos\_x, tile\_height \* pos\_y)  
  
 def set\_image(self):  
 self.image = tile\_images[self.tile\_type]  
 self.rect = self.image.get\_rect().move(  
 tile\_width \* self.pos\_x, tile\_height \* self.pos\_y)  
  
 def set\_color(self):  
 self.image = load\_image('black.jpg')  
 self.rect = self.image.get\_rect().move(  
 tile\_width \* self.pos\_x, tile\_height \* self.pos\_y)

set\_image() задает картинку клетке, если та находится в поле видимости (квадрат 3\*3 с героем в центре), в остальных случая становится черного цвета(метод set\_color())

немного об load\_image()

def load\_image(name, colorkey=None):  
 fullname = os.path.join('data', name)  
 # если файл не существует, то выходим  
 if not os.path.isfile(fullname):  
 print(f"Файл с изображением '{fullname}' не найден")  
 sys.exit()  
 image = pygame.image.load(fullname)  
 return image

load\_image() упрощает загрузку изображения.

Player(унаследствован от pygame.sprite.Sprite) – главный герой(за которого мы играем)

class Player(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, pos\_x, pos\_y):  
 super().\_\_init\_\_(player\_group, all\_sprites)  
 self.x = pos\_x  
 self.y = pos\_y  
 self.image = player\_image  
 self.rect = self.image.get\_rect().move(  
 tile\_width \* pos\_x + 15, tile\_height \* pos\_y + 5)  
 self.step = pygame.mixer.Sound('data\step.ogg')  
 self.step\_2 = pygame.mixer.Sound('data\step\_2.ogg')  
 self.step\_3 = pygame.mixer.Sound('data\step\_3.ogg')  
 for x in treasures.keys():  
 x.kill()  
 if abs(x.x - self.x) <= 1 and abs(x.y - self.y) <= 1 and treasures[x]:  
 x.add(treasure\_group)  
 if x.x - self.x == 0 and x.y - self.y == 0:  
 x.kill()  
 treasures[x] = False  
  
 def set\_image(self, image):  
 self.image = image  
  
 def move\_up(self):  
 if self.y == 0 and True not in treasures.values():  
 return True  
 elif self.y - door.y == 1 and self.x == door.x and not (door.bool):  
 door.open()  
 elif self.y - 1 > -1 and map\_list[self.y - 1][self.x].tile\_type == 'empty':  
 choice([self.step, self.step\_2, self.step\_3]).play()  
 self.y -= 1  
 self.rect = self.image.get\_rect().move(  
 tile\_width \* self.x + 15, tile\_height \* self.y + 5)  
 map\_list[self.y][self.x].set\_image()  
 if self.y - 1 > -1:  
 map\_list[self.y - 1][self.x].set\_image()  
 if self.x - 1 > -1:  
 map\_list[self.y - 1][self.x - 1].set\_image()  
 if self.x + 1 < len(map\_list[0]):  
 map\_list[self.y - 1][self.x + 1].set\_image()  
 if self.y + 2 < len(map\_list):  
 map\_list[self.y + 2][self.x].set\_color()  
 if self.x - 1 > -1:  
 map\_list[self.y + 2][self.x - 1].set\_color()  
 if self.x + 1 < len(map\_list[0]):  
 map\_list[self.y + 2][self.x + 1].set\_color()

Примечание: остальные функции не показаны здесь, т.к. созданы по образу и подобию move\_up()

move\_{направление}() отвечает за движение игрока. Для этого каждый раз за ход 6 квадратов меняют свое состояние: одни становятся невидимыми(черными), другие – видимыми(в соответствие с типом клетки)

Treasure(унаследствован от pygame.sprite.Sprite) – объекты, которые нужно собрать

class Treasure(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, pos\_x, pos\_y):  
 super().\_\_init\_\_(treasure\_group, all\_sprites)  
 self.x = pos\_x  
 self.y = pos\_y  
 self.image = treasure\_image  
 self.rect = self.image.get\_rect().move(  
 tile\_width \* self.x + 15, tile\_height \* self.y + 5)

Door(унаследствован от pygame.sprite.Sprite) – выход из уровня

class Door(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, pos\_x, pos\_y):  
 super().\_\_init\_\_(all\_sprites, door\_group)  
 self.x = pos\_x  
 self.y = pos\_y  
 self.sound\_1 = pygame.mixer.Sound('data\door-opening.mp3')  
 self.sound\_2 = pygame.mixer.Sound('data\door-closed.mp3')  
 self.bool = False  
 self.image = door\_image  
 self.rect = self.image.get\_rect().move(  
 tile\_width \* pos\_x, tile\_height \* pos\_y)  
  
 def open(self):  
 if True in treasures.values():  
 self.sound\_2.play()  
 else:  
 self.sound\_1.play()  
 self.bool = True  
 self.kill()

Когда все сокровища собраны, метод open() убирает дверь.

Note(унаследствован от pygame.sprite.Sprite) – записка, хранящая некоторую информацию.

class Note(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, pos\_x, pos\_y, key):  
 super().\_\_init\_\_(all\_sprites, note\_group)  
 self.x = pos\_x  
 self.y = pos\_y  
 self.key = key  
 self.bool = True  
 self.image = note\_image  
 self.rect = self.image.get\_rect().move(  
 tile\_width \* pos\_x, tile\_height \* pos\_y)  
 self.sound = pygame.mixer.Sound('data/achievement.ogg')  
  
 def collected(self, player):  
 global notes\_opened  
 if player.x == self.x and player.y == self.y:  
 self.kill()  
 self.bool = False  
 notes\_opened[str(self.key)] = True  
 self.sound.play()

метод collected() убирает записку, если ее координата совпадает с координатой игрока.

Спрайты для загрузки классов на экран

all\_sprites = pygame.sprite.Group()  
tiles\_group = pygame.sprite.Group()  
player\_group = pygame.sprite.Group()  
treasure\_group = pygame.sprite.Group()  
door\_group = pygame.sprite.Group()  
note\_group = pygame.sprite.Group()

4. Системные требования

|  |  |
| --- | --- |
| 64-битная операционная система Microsoft Windows | * Windows 10 |
| Компьютер и процессор | * 1,6 ГигаГерц (ГГц) или быстрее * 2 или больше ядер |
| Память | * 6 Мбайт оперативной памяти |
| Дисплей | * Разрешение 1280 x 720 или выше |

5. Список используемой литературы

<https://python-course.readthedocs.io>

<https://devdocs.io/pygame>