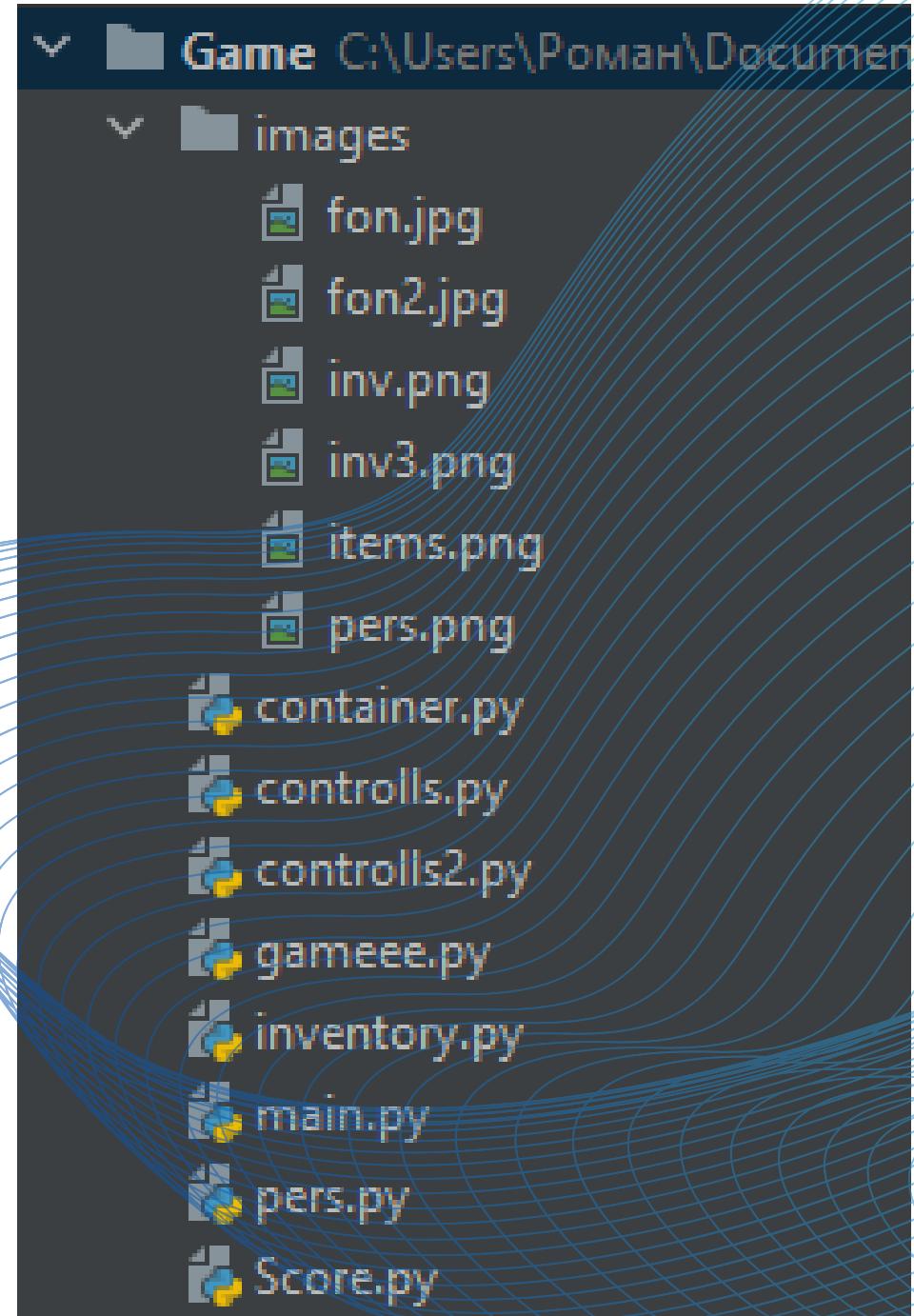




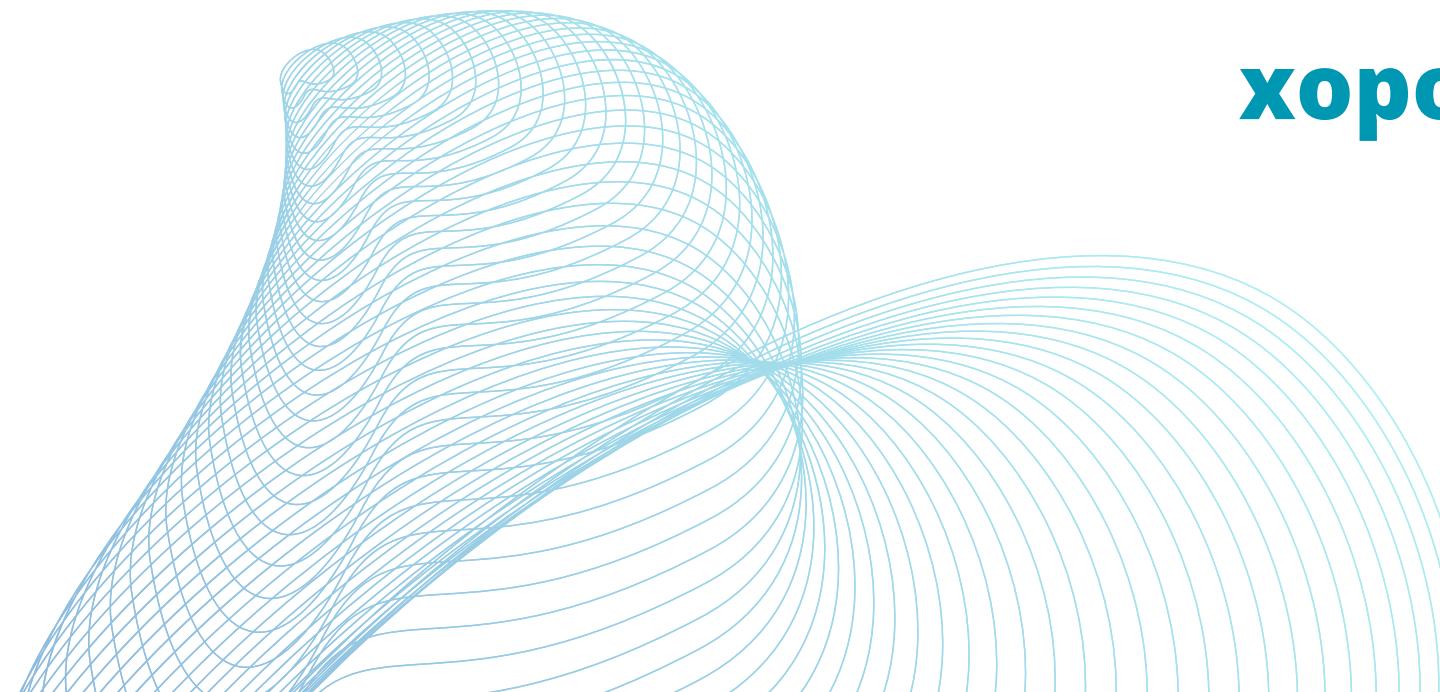
"ГРУЗЧИК"

**Выполняли: Горожий София(ИС-21) и
Сидоренко Роман(ИС-24)**



PROJECT TASK

- 1. Самой главной задачей проекта является закрепление знаний и навыков по ООП на Python;**
- 2. Не мало важным является создание самой игры, которая будет выполнять свои задачи: коротание времени, хорошая позитивная реакция;**



GAME CYCLE

Run() - это функция, приводящая игру в действие, в нее переданы другие функции, а к ним аргументы других взаимосвязанных файлов игры. Имеется две функции: run, run2. Они и определяют уровень игры. В игре максимально простые текстуры, также простой смысл, а именно перетаскивание ящиков с места выгрузки на склад, путем многократного нажатия клавиши SPACE, а также движения персонажа в разные стороны.

```
def run():

    pygame.init()
    screen = pygame.display.set_mode((1400, 1000))
    pygame.display.set_caption("Грузчик")
    background_color = pygame.image.load('images/fon2.jpg').convert()
    pers = Pers(screen)
    inventory = Inventory(screen)
    containers = Group()
    score = Score()
    controls.create_items(screen, containers)

    while True:
        controls.events(pers, containers)
        controls.colvo(containers, inventory, score)
        pers.update_pers()
        controls.update(background_color, screen, pers, containers, inventory)

"""Второй уровень"""
def run2():

    pygame.init()
    screen = pygame.display.set_mode((1400, 1000))
    pygame.display.set_caption("ГРУЗЧИК")
    background_color = pygame.image.load('images/fon.jpg').convert()
    pers = Pers(screen)
    inventory = Inventory(screen)
    containers = Group()
    controls2.create_items(screen, containers)

    while True:
        controls2.events(pers, containers)
        controls2.colvo(containers, inventory)
        pers.update_pers()
        controls2.update(background_color, screen, pers, containers, inventory)
```

CLASS SCORE

Score - класс,
позволяющий
производить подсчет
ящиков, которые были
доставлены на склад, но
не имеет инициализации
на экране.

```
"""подсчет очков"""
class Score:
    def __init__(self):
        self.score = 0
        self.score_up = 1

    def score_upg(self):
        self.score += self.score_up
        print(self.score)
```

DEF EVENTS

Данная функция является основной в управлении персонажем и его взаимодействием с ящиками.

Также в ней реализован выход из игры.

```
def events(pers, containers):
    """обработка событий"""

    """функция трекинга клавиш"""
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        elif event.type == pygame.KEYDOWN:
```

UPDATE AND CREATE ITEMS

Update() - функция,
интегрирующая объекты на
экран.

Create_items() - функция,
создающая с одного
представления целую группу
контейнеров.

```
def update(background_color, screen, pers, containers, inventory):
    """screen.fill(background_color)"""
    screen.blit(background_color, (0, 0))
    inventory.rect.x = 1200
    inventory.rect.y = 300
    inventory.draw(screen)
    pers.output()
    containers.draw(screen)
    pygame.display.flip()

def create_items(screen, containers):
    """create items for screen"""
    container = Items(screen)
    container_height = container.rect.height
    number_container = int((800 - 2 * container_height) / container_height)

    for container_number in range(number_container):
        container = Items(screen)
        container.y = container_height + 1.4 * container_height * container_number
        container.rect.y = container.y
        containers.add(container)
        print(containers)
```

CLASS PERS

Pers() - не просто объект, создаваемый в нашей игре, это первый объект, созданный при написании игры. В нем описаны все элементы, относящиеся к персонажу игру, здесь его инициализация, функция отображения на экране, функция описания передвижения персонажа и характеристики движения.

```
class Pers():

    def __init__(self, screen):
        """инициализация персонажа"""

        self.screen = screen
        self.image = pygame.image.load('images/pers.png')
        self.rect = self.image.get_rect()
        self.screen_rect = screen.get_rect()
        self.rect.x = self.rect.width
        self.rect.y = self.rect.height
        self.rect.centerx = self.screen_rect.centerx
        self.rect.centery = self.screen_rect.centery
        self.centerx = float(self.rect.centerx)
        self.centery = float(self.rect.centery)
        self.rect.bottom = self.screen_rect.bottom
        """для клавиш"""
        self.mtop = False
        self.mdown = False
        self.mright = False
        self.mleft = False

    def output(self):
        """рисование персонажа"""
        self.screen.blit(self.image, self.rect)

    def update_pers(self):
        """обновление позиции персонажа"""

        """обновление для движения вверх"""
        if self.mtop and self.rect.top > self.screen_rect.top:
            self.centery -= 0.7

        """обновление для движения вниз"""
        if self.mdown and self.rect.bottom < self.screen_rect.bottom:
            self.centery += 0.7

        """обновление для движения влево"""
        if self.mleft and self.rect.left > 0:
```

CLASS ITEMS

Items() - это объект, выполняющий в игре роль контейнера, который переносит наш персонаж. Он также как и персонаж имеет свою инициализацию, отображение на экране. Конечно, этот объект более простой и пустой относительно предыдущего, но несет не мало важную роль в самой игре, так как с помощью него выполняется главная идея игры.

```
class Items(pygame.sprite.Sprite):  
  
    def __init__(self, screen):  
        """initial"""  
        super(Items, self).__init__()  
        self.screen = screen  
        self.image = pygame.image.load('images/items.png')  
        self.rect = self.image.get_rect()  
        self.rect.width = 80  
        self.rect.height = 80  
        self.rect.x = self.rect.width  
        self.rect.y = self.rect.height  
        self.x = float(self.rect.x)  
        self.y = float(self.rect.y)  
  
    def draw(self):  
        """рисование ящика"""  
        self.screen.blit(self.image, self.rect)
```

CLASS INVENTORY

Inventory() - объект, инициализирующий инвентарь, также как и остальные объекты он имеет несколько функций в своем "арсенале", такие как инициализация и отрисовка на экране.

```
class Inventory():

    def __init__(self, screen):
        self.screen = screen
        self.image = pygame.image.load('images/inv.')
        self.rect = self.image.get_rect()
        self.rect.x = self.rect.width
        self.rect.y = self.rect.height
        self.rect.width = 350
        self.rect.height = 400

    def draw(self, screen):
        """рисование склада"""
        self.screen.blit(self.image, self.rect)
```

CLASS MENU

Class Menu - это объект, позволяющий создать меню при входе в игру, в котором есть только два действия на выбор: 'ИГРАТЬ', 'ВЫЙТИ'.

**Также данное меню
появляется после
прохождения игрыю**

```
class Menu:

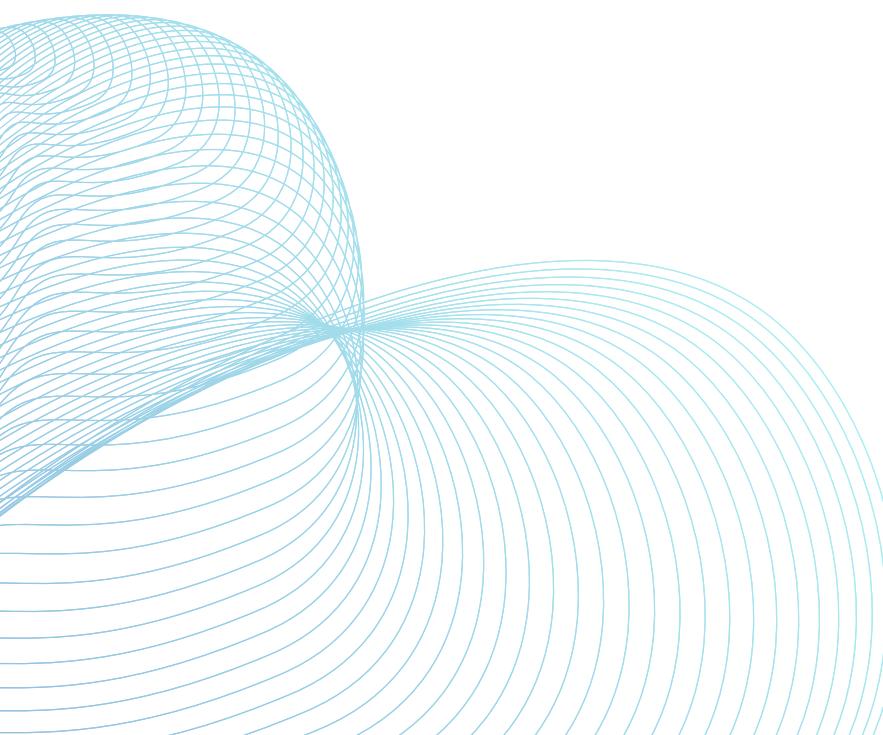
    def __init__(self, punkts = [120, 140, u'Punkt', (250, 250, 30), (250, 30, 250)]):
        self.punkts = punkts

    def render(self, poverhnost, font, num_punkt):
        for i in self.punkts:
            if num_punkt == i[5]:
                poverhnost.blit(font.render(i[2], 1, i[4]), (i[0], i[1]))
            else:
                poverhnost.blit(font.render(i[2], 1, i[3]), (i[0], i[1]))

    def menu(self):
        done = True
        b_c = pygame.image.load('images/menu_fon.jpg').convert()
        screen.blit(b_c, (0, 0))
        font_menu = pygame.font.Font('fonts/ArialRegular.ttf', 50)
        punkt = 0
        while done:

            mp = pygame.mouse.get_pos()
            for i in self.punkts:
                if mp[0] > i[0] and mp[0] < i[0]+155 and mp[1] > i[1] and mp[1] < i[1]+30:
                    punkt = i[5]
            self.render(screen, font_menu, punkt)
```

MENU



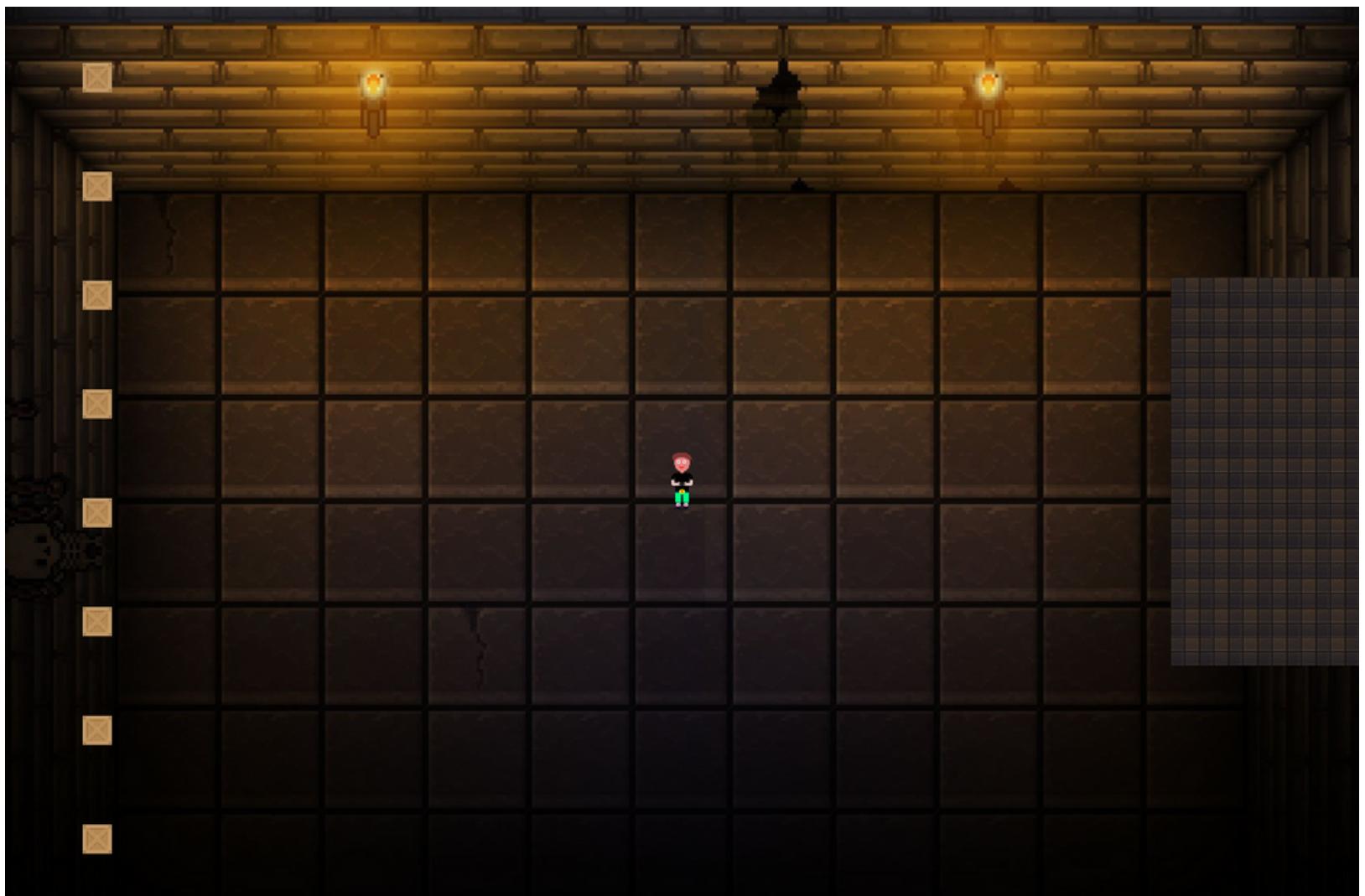
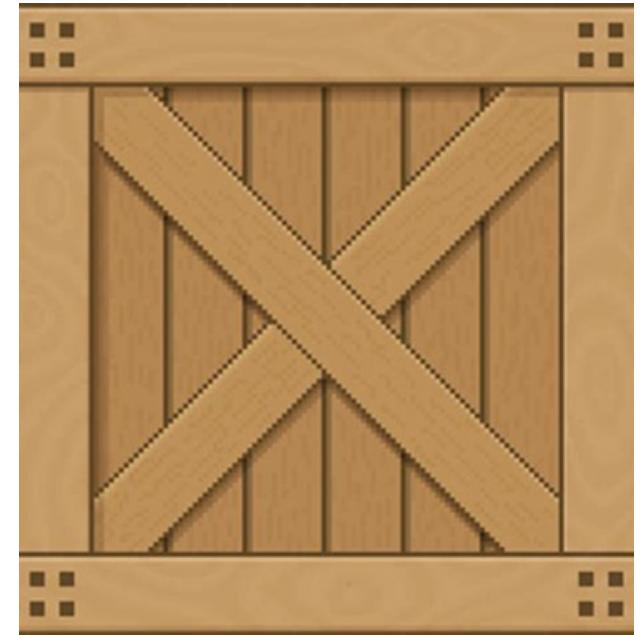
TESTING THE GAME

Здесь представлены два уровня игры, а также текстура объекта Ящик.

Первый уровень имеет 8 ящиков, темный фон и более светлый склад.

Второй уровень имеет 16 ящиков, яркий фон и темный склад.

Текстура ящика выполнена, соответствующая реальному виду ящика.



THE END

Подведем итоги, в ходе разработки проекта мы научились пользоваться библиотекой *pygame*, *sys*.

***Руgате* - библиотека с уже имеющимися в себе функциями, которые облегчают процесс кодинга игры.**

Sys - системная библиотека, имеющая в своем арсенале нужные функции.

Мы закрепили знания по ООП на языке программирования Python.

В итоге проект окончен, задачи выполнены, мы довольны своей работой!!!