Министерство науки и высшего образования РФ
федеральное государственное автономное образовательное учреждение высшего образования
«Омский государственный технический университет»

Лицей Академии Яндекса

R.P.S

Выполнил:
Гончаров Пётр Валентинович

Омск, 2023

РЕФЕРАТ

Отчет 20 с., 4 рис., 2 табл., 3 источн., 1 прил.

ПРОЕКТ, GUI ИНТЕРФЕЙС, ДАННЫЕ, PYTHON, PYGAME,

Предмет исследования – приложение.

Цель исследования – разработка игры R.P.S для отдыха и развития

В процессе выполнения проекта – я получил навыки: работа с .csv форматом, работа с python и библиотекой PyGame и PyGame_gui

В результате выполнения проекта – я получил игру, в которой игрок может сам создавать свои уровни, соревноваться на них и пытаться побить свой старый рекорд

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Текст введения. Во введении обосновывается актуальность и новизна реализуемого проекта.

Цель исследования – разработка приложения R.P.S для для отдыха и развития

Для достижения поставленной цели требуется решить следующие задачи:

1) исследовать проблему предметной области ;
2) спроектировать интерфейс игры и систему хранения данных ;
3) реализовать спроектированное игру.

1 Описание проблемы и формулирование гипотезы для ее решения

Иногда, люди, который приходят с работы, учёбы и т.д. хотят отдохнуть. Бывают игры или фильмы которые с этим справляются, но на все потребности игроков, они могут не ответить, поэтому мною было принято решение, о разработке игры под названием R.P.S.

2 Проектирование приложения

Приложение будет состоять из:

1) *.exe файла*
2) *Папки со всеми зависимостями*

через .exe файл будет осуществляться запуск программы. В папке будут находиться все нужные для работы игры файлы (Картинки, информация об игроках и уровнях), также в этой папке игрок может изменять структуру 2-х уровней

3 Реализация приложения

1) *Сначала мне нужно было создать класс объекта в python под названием Board(), благодаря этому классу и библиотеки PyGame, я реализовал систему отрисовки уровня и игровой доски.*
2) *Потом мне нужно было создать класс Unit() в котором был весь функционал для юнитов (постановка на уровень, тип юнита и т.д.)*

3) *Также для создания условия победы и системы здоровья - был создан класс Base(), в котором был весь функционал для базы (отнимания здоровья, постановка на уровень и т.д.)*
4) *Потом сделать две функций game() и main(). В game() была реализация игры - через все вышеприведенные классы, и перенос данных в .csv файл. В main() была сделана система выбора уровней, подгрузка результатов и отображения лучшего результата игрока на одном из уровней*

ЗАКЛЮЧЕНИЕ

Текст заключения: на каждую задачу, прописанную во введении оформляете один абзац текста с подведением итогов по решению этой задачи. После абзацев по задачам идет абзац текста о дальнейших перспективах реализованного продукта.

1) *Я исследовал проблему предметной области. Смог создать игру для решения этой проблемы*
2) *Я спроектировал интерфейс игры, он отображает всю информацию уровне. Для работы основного .exe файла нужны минимум два файла. 1 из них это Players_data.csv - в котором хранится информация об игроке, его рекордах и т.д.. 2 из них это 1_level.csv, в котором храниться информация об уровне (база игрока, противника, путь)*
3) *Я реализовал спроектированную игру. В неё можно играть, изменять уровни, побеждать свои прошлые рекорды.*
4) *Игра написана так, чтобы можно было легко добавить новый уровень или расширить поле, Можно также добавить новые типы юнитов. Из-за того что все данные об уровнях хранятся в .csv файле, и игрок может изменять их, игра может иметь неплохую перспективу стать популярной.*

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Обзор курса — Основы промышленного программирования | Д22 — Лицей Академии Яндекса (yandex.ru)

2 Pygame tutorial Documentation

3 Documentation - Home Page — Pygame GUI 0.6.9 documentation (pygame-gui.readthedocs.io)

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг

```
import pygame
import csv

import os
import sys

import random

import pygame_gui

side_winner = None
time_win = 0
kill_player = 0
kill_enemy = 0
switch = None

game_over = False


def change_kill_player(var):
    global kill_player
    kill_player = var


def get_kill_player():
    return kill_player


def change_kill_enemy(var):
    global kill_enemy
    kill_enemy = var


def get_kill_enemy():
    return kill_enemy


def change_time_win(var):
    global time_win
    time_win = var
```

```
    def get_time_win():
        return time_win


    def change_game_over(var):
        global game_over
        game_over = var


    def reset():
        global game_over, time_win, kill_player, kill_enemy, switch,
side_winner
        side_winner = None
        time_win = 0
        kill_player = 0
        kill_enemy = 0
        switch = None

        game_over = False


    def game(level, player, speed=50, hp_base=50):
        global time_win
        global kill_player
        global kill_enemy
        global game_over
        global switch


        with open(level, encoding="utf8") as csvfile:
                    reader = csv.DictReader(csvfile, delimiter=',',
quotechar='"')
            level_data = list(reader)
            way_start_player_pos = [elem[0].split(",") for elem in
                                    [elem.split(";") for elem in
level_data[0]["way"].split("|")]]
            way_start_enemy_pos = [elem[-1].split(",") for elem in
                                    [elem.split(";") for elem in
level_data[0]["way"].split("|")]]
            h_w_cell = level_data[0]["h_w_cell"].split(",")

            player_tick = speed // 2
            enemy_tick = 0

            fps = 60

            print(level_data)
            print(way_start_player_pos)
            print(h_w_cell)
```

```
                print(str([elem.split(";")[1:] for elem in
level_data[0]["way"].split("|")]))

    pygame.init()
    pygame.display.set_caption(level.split("/")[-1])
    pygame.font.init()
    manager = pygame_gui.UIManager((800, 800))

    size = width, height = 800, 800
    screen = pygame.display.set_mode(size)

    all_sprites = pygame.sprite.Group()
    player_unit = pygame.sprite.Group()
    enemy_unit = pygame.sprite.Group()

    def load_image(name, colorkey=None):
        fullname = os.path.join('data', name)
        if not os.path.isfile(fullname):
            print(f"Файл с изображением '{fullname}' не найден")
            sys.exit()
        image = pygame.image.load(fullname)

        if colorkey is not None:
            image = image.convert()
            if colorkey == -1:
                colorkey = image.get_at((0, 0))
            image.set_colorkey(colorkey)
        else:
            image = image.convert_alpha()
        return image

    class Unit(pygame.sprite.Sprite):

        def __init__(self, *group, type_unit=None, side=None):
            super().__init__(*group)

            self.add(player_unit if side == "player" else
enemy_unit)

            if type_unit == "paper":
                self.image =
load_image("sprites/paper/paper-50-0.png")
            elif type_unit == "rock":
                self.image =
load_image("sprites/rock/rock-50-0.png")
            elif type_unit == "scissors":
                self.image =
load_image("sprites/scissors/scissors-50-0.png")
            self.type_unit = type_unit

            self.rect = self.image.get_rect()
            self.side = side
```

```
            self.start_pos = None
            self.enemy_c = False

        def update(self, board, event=None):
            if self.side == "player":
                if player_tick == 0:
                    self.update_movement()
                    if pygame.sprite.spritecollideany(self,
enemy_unit):
                        print(pygame.sprite.spritecollideany(self,
enemy_unit))
                        if pygame.sprite.spritecollideany(self,
enemy_unit).get_type_unit() == "paper" and self.type_unit ==
"scissors" or pygame.sprite.spritecollideany(self,
enemy_unit).get_type_unit() == "rock" and self.type_unit == "paper"
or pygame.sprite.spritecollideany(self, enemy_unit).get_type_unit()
== "scissors" and self.type_unit == "rock":
                            change_kill_player(get_kill_player() + 1)
                            pygame.sprite.spritecollideany(self,
enemy_unit).kill()
                        elif pygame.sprite.spritecollideany(self,
enemy_unit).get_type_unit() == self.type_unit:
                            change_kill_enemy(get_kill_enemy() + 1)
                            change_kill_player(get_kill_player() + 1)
                            pygame.sprite.spritecollideany(self,
enemy_unit).kill()
                            self.kill()
                        else:
                            change_kill_enemy(get_kill_enemy() + 1)
                            self.kill()

            elif self.side == "enemy":
                if enemy_tick == 0:
                    self.update_movement()
                    if pygame.sprite.spritecollideany(self,
player_unit):
                        print(pygame.sprite.spritecollideany(self,
player_unit))


        def spawn(self, board, cell):
            self.rect.x = cell[0] * 50 +
board.get_left_and_top()[1]
            self.rect.y = cell[1] * 50 +
board.get_left_and_top()[0]
            self.cell = cell
            if self.side == "player":
                self.start_pos = (cell if cell in list(
                    map(lambda x: tuple(map(lambda y: int(y),
x)), way_start_player_pos)) else self.start_pos)
            else:
                self.start_pos = (cell if cell in list(
```

```
                                map(lambda x: tuple(map(lambda y: int(y),
x)), way_start_enemy_pos)) else self.start_pos)


        def get_cell(self):
            return self.cell

        def get_side(self):
            return self.side

        def get_type_unit(self):
            return self.type_unit

        def update_movement(self):
            enemy_c = False
            if self.side == "player" and not game_over:
                try:
                                                        section   =
way_start_player_pos.index(list(map(lambda       x:       str(int(x)),
self.start_pos)))
                                                          path    =
level_data[0]["way"].split("|")[section].split(";")
                                                        new_cell  =
path[path.index(",".join(list(map(lambda       x:       str(int(x)),
self.cell)))) + 1]
                except IndexError:
                    enemy_base.hearth()
                    self.kill()
                    return
            elif self.side == "enemy" and not game_over:
                                                        section   =
way_start_enemy_pos.index(list(map(lambda       x:       str(int(x)),
self.start_pos)))
                                                          path    =
level_data[0]["way"].split("|")[section].split(";")
                    if path[path.index(",".join(list(map(lambda x:
str(int(x)),     self.cell))))     -     1].split(",")     not     in
way_start_player_pos and not self.enemy_c:
                                                        new_cell  =
path[path.index(",".join(list(map(lambda       x:       str(int(x)),
self.cell)))) - 1]
                elif path[path.index(",".join(list(map(lambda x:
str(int(x)),  self.cell))))  -  1].split(",")  in  way_start_player_pos
and not self.enemy_c:
                                                        new_cell  =
path[path.index(",".join(list(map(lambda       x:       str(int(x)),
self.cell)))) - 1]
                    self.enemy_c = True
                else:
                    player_base.hearth()
                    self.kill()
                    return
```

```
            else:
                return
                 self.spawn(board, tuple(map(lambda x: float(x),
new_cell.split(","))))


    class Base(pygame.sprite.Sprite):

        def __init__(self, *group, side=None):
            super().__init__(*group)

                 self.add(player_unit if side == "player" else
enemy_unit)

                                                        self.image =
load_image("sprites/base/base-100-0.jpg")

            self.type_unit = "base"

            self.rect = self.image.get_rect()

            self.side = side

            self.hp = hp_base


        def update(self, board, event=None):
            pass

        def spawn(self, board, cell):
                                self.rect.x = cell[0] * 50 +
board.get_left_and_top()[1]
                                self.rect.y = cell[1] * 50 +
board.get_left_and_top()[0]
            self.cell = cell

        def get_cell(self):
            return self.cell

        def get_cell(self):
            return self.cell

        def get_side(self):
            return self.side

        def get_type_unit(self):
            return self.type_unit

        def hearth(self):
            self.hp -= 5
            print(self.hp)
```

```
            if self.hp <= 0:
                global switch
                change_game_over(self.side)
                switch = pygame_gui.elements.UIButton(
                    relative_rect=pygame.Rect((250, 10), (300,
50)),
                    text="Exit",
                    manager=manager
                )
                self.kill()


    class Board:
        def __init__(self, width, height):
            self.width = width
            self.height = height
            self.board = [[0] * width for _ in range(height)]
            self.left = 10
            self.top = 10
            self.cell_size = 50
                self.way = [elem.split(";") for elem in
level_data[0]["way"].split("|")]

        def set_view(self, left, top, cell_size):
            self.top = top
            self.left = left
            self.cell_size = cell_size

        print(way_start_player_pos)
        print(way_start_player_pos + way_start_enemy_pos)
        def render(self, screen):
            for j in range(self.height):
                for i in range(self.width):
                    if f"{i},{j}" in str([elem.split(";")[1:-1]
for elem in level_data[0]["way"].split("|")]):
                        pygame.draw.rect(screen,
pygame.Color("green"), (i * self.cell_size + self.top, j *
self.cell_size + self.left, self.cell_size, self.cell_size), 1)
                    else:
                        pygame.draw.rect(screen,
pygame.Color("blue") if any(map(lambda x: True if int(x[0]) == i and
int(x[1]) == j else False, way_start_player_pos +
way_start_enemy_pos)) else pygame.Color("black"), (i * self.cell_size
+ self.top, j * self.cell_size + self.left, self.cell_size,
self.cell_size), 1)

        def get_left_and_top(self):
            return (self.left, self.top)

        def get_cell(self, mouse_pos):
            coord = ((mouse_pos[0] - self.top) // self.cell_size,
(mouse_pos[1] - self.left) // self.cell_size)
```

```
                return None if coord[0] >= len(self.board[0]) or
coord[0] < 0 or coord[1] >= len(self.board) or coord[1] < 0 else
coord

        def get_cell_size(self):
            return self.cell_size

        def get_board(self):
            return self.board

        def on_click(self, cell):
            if cell:
                pass

        def get_click(self, mouse_pos):
            cell = self.get_cell(mouse_pos)
            print(cell)
            self.on_click(cell)


    board = Board(int(h_w_cell[0]), int(h_w_cell[1]))
     board.set_view(width / 2 - (len(board.get_board()) / 2 *
board.get_cell_size()), height / 2 - (len(board.get_board()[0]) / 2 *
board.get_cell_size()), 50)

    subject_selection = "paper"

    player_base = Base(all_sprites, side="player")
    enemy_base = Base(all_sprites, side="enemy")
     player_base.spawn(board, tuple(map(lambda x: int(x),
level_data[0]["base_player"].split(","))))
      enemy_base.spawn(board, tuple(map(lambda x: int(x),
level_data[0]["base_enemy"].split(","))))

    clock = pygame.time.Clock()

    running = True

    my_font = pygame.font.SysFont('Comic Sans MS', 20)

    while running:
        time_delta = clock.tick(fps) // 1000.0
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            if event.type == pygame.MOUSEBUTTONDOWN:
                print(board.get_cell(event.pos))
                if board.get_cell(event.pos):
                    cell = board.get_cell(event.pos)
                    if any(map(lambda x:
                                True if int(x[0]) == cell[0] and
int(x[1]) == cell[1] else False, way_start_player_pos)) \
```

```
                                    and cell not in [elem.get_cell() for
elem in player_unit.sprites()] and not game_over:
                                                Unit(all_sprites,
type_unit=subject_selection,              side="player").spawn(board,
board.get_cell(event.pos))

            elif event.type == pygame.KEYDOWN:
                if event.key == 49:
                    subject_selection = "paper"
                elif event.key == 50:
                    subject_selection = "rock"
                elif event.key == 51:
                    subject_selection = "scissors"

            elif event.type == pygame.USEREVENT:
                                        if   event.user_type   ==
pygame_gui.UI_BUTTON_PRESSED:
                    if event.ui_element == switch and game_over
== "enemy":
                        reader_data = None
                                                with
open("data/players/Players_data.csv", "r") as player_data:
                                                    reader_data  =
list(csv.DictReader(player_data, delimiter=';', quotechar='"'))
                        for i in range(len(reader_data)):
                            if reader_data[i]["player_name"]
== player and int(reader_data[i][level.split("/")[-1]].split(",")[2])
<= get_kill_player():
                                                    buffer  =
reader_data[i][level.split("/")[-1]].split(",")
                                        buffer[0] = get_time_win() //
60
                                        buffer[1] = get_kill_player()
                                        buffer[2] = get_kill_enemy()

reader_data[i][level.split("/")[-1]]  =  ",".join(list(map(lambda  x:
str(x), buffer)))
                        print(reader_data)

                                                with
open('data/players/Players_data.csv', 'w') as f:
                        writer = csv.DictWriter(
                                                        f,
fieldnames=list(reader_data[0].keys()),
                                                    delimiter=';',
quoting=csv.QUOTE_NONNUMERIC)
                        writer.writeheader()
                        for d in reader_data:
                            writer.writerow(d)
                        reset()
                        return
                return
```

```
                manager.process_events(event)
            manager.update(time_delta)
            if enemy_tick == player_tick // 2 and not game_over:
                Unit(all_sprites, type_unit=random.choice(('paper',
'rock',             'scissors')),             side="enemy").spawn(board,
random.choice(tuple(map(lambda  x:  tuple(map(lambda  y:  int(y),  x)),
way_start_enemy_pos))))


            screen.fill((255, 255, 255))
            board.render(screen)
            all_sprites.update(board)
            all_sprites.draw(screen)
            if game_over:
                win_text_surface = my_font.render(f"{'enemy' if
game_over == 'player' else 'player'} win", False, (0, 0, 0))
                time_text_surface = my_font.render(f"time:
{get_time_win() // 60}", False, (0, 0, 0))
                kill_enemy_text_surface = my_font.render(f"kill
enemy: {get_kill_enemy()}", False, (0, 0, 0))
                kill_player_text_surface = my_font.render(f"kill
player: {get_kill_player()}", False, (0, 0, 0))
                screen.blit(win_text_surface, (width // 25 + 50,
height // 15))
                screen.blit(time_text_surface, (width // 2 + 50,
height // 15))
                screen.blit(kill_enemy_text_surface, (width // 1.5 +
50, height // 15))
                screen.blit(kill_player_text_surface, (width // 4 +
50, height // 15))
            else:
                change_time_win(get_time_win() + 1)
            manager.draw_ui(screen)
            pygame.display.flip()

            player_tick += 1
            enemy_tick += 1

            if player_tick >= speed:
                player_tick = 0
            if enemy_tick >= speed:
                enemy_tick = 0


    def main(player):
        pygame.init()
        pygame.font.init()
        pygame.display.set_caption('Menu')
        size = width, height = 800, 800
        screen = pygame.display.set_mode(size)
        manager = pygame_gui.UIManager((800, 800))
```

```
    clock = pygame.time.Clock()
    fps = 60

    my_font = pygame.font.SysFont('Comic Sans MS', 20)

    switch_1_level = pygame_gui.elements.UIButton(
        relative_rect=pygame.Rect((50, 250), (250, 100)),
        text="level_1",
        manager=manager
    )

    switch_2_level = pygame_gui.elements.UIButton(
        relative_rect=pygame.Rect((500, 250), (250, 100)),
        text="level_2",
        manager=manager
    )

    def update_values():
        info_1_level = None
        info_2_level = None

                with open("data/players/Players_data.csv", "r") as
player_data:
                    reader_data = list(csv.DictReader(player_data,
delimiter=';', quotechar='"'))
                for elem in reader_data:
                    if elem["player_name"] == player:
                        info_1_level = elem["1_level.csv"].split(",")
                        info_1_level[0] = f"time: {info_1_level[0]}"
                                    info_1_level[1] = f"kill pl:
{info_1_level[1]}"
                                    info_1_level[2] = f"kill en:
{info_1_level[2]}"
                        info_1_level = ", ".join(info_1_level)

                        info_2_level = elem["2_level.csv"].split(",")
                        info_2_level[0] = f"time: {info_2_level[0]}"
                                    info_2_level[1] = f"kill pl:
{info_2_level[1]}"
                                    info_2_level[2] = f"kill en:
{info_2_level[2]}"
                        info_2_level = ", ".join(info_2_level)
        return [info_1_level, info_2_level]

    info_1_level, info_2_level = update_values()

    running = True
    while running:
        time_delta = clock.tick(fps)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
```

```
            elif event.type == pygame.USEREVENT:
                if event.user_type == pygame_gui.UI_BUTTON_PRESSED:
                    if event.ui_element == switch_1_level:
                        game("data/levels/1_level.csv", "player1", speed=50, hp_base=25)
                        info_1_level, info_2_level = update_values()
                        reset()
                    elif event.ui_element == switch_2_level:
                        game("data/levels/2_level.csv", "player1", speed=50, hp_base=25)
                        info_1_level, info_2_level = update_values()
                        reset()
                    pass


                manager.process_events(event)
            manager.update(time_delta)
            screen.fill((0, 0, 0))
            manager.draw_ui(screen)
            player_text_surface = my_font.render(f"player: {player}", False, (255, 255, 255))
            screen.blit(player_text_surface, (50, 50))
            screen.blit(my_font.render(f"records:", False, (255, 255, 255)), (50, 100))

            screen.blit(my_font.render(f"{info_1_level}", False, (255, 255, 255)), (50, 125))
            screen.blit(my_font.render(f"{info_2_level}", False, (255, 255, 255)), (500, 125))

            pygame.display.flip()
        pygame.quit()


    if __name__ == "__main__":
        main("player1")
```