Министерство образования Республики Беларусь
Учреждение образования
"БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ"

Факультет информационных технологий и управления
Кафедра интеллектуальных информационных технологий

ЛАБОРАТОРНАЯ РАБОТА No3

по дисциплине "Проектирование программного обеспечения интеллектуальных систем"

Выполнил студент группы 121701: В. А. Пахомов

Проверил: С. В. Бутрин

Минск 2023

Цель: Изучить событий-ориентированное программирование с использованием библиотеки pygame

Задание: Разработать игровое приложение согласно выбранному варианту. При разработке игры необходимо изучить функциональность оригинальной игры и по умолчанию реализовывать правила оригинальной игры, если нет ограничивающих требований в условиях задания.

Вариант 8. Jewel Quest

- Реализовать два режима игры: в ограничение по времени и по количеству набранных очков
- Для второго режима разработать не менее 3 уровней
- Важно реализовать анимации (плавность и тайминги) как в оригинальной игре

Описание программы: Программа состоит из одного модуля, который содержит в себе один основной класс Stone, и три основные функции main_menu(), mode_menu(), game().

Класс Stone описывает объект драгоценный камень, а именно:

- Отрисовывает
- Описывает размещение камней по полю
- Накладывает избражение, соответствующее цвету камня

```
class Stone:

    def __init__(self, row_num, col_num):

        self.row_num = row_num
        self.col_num = col_num

        self.color = random.choice(stone_colors)
        image_name = f'stone_{self.color}.png'
        self.image = pygame.image.load(image_name)
        self.image = pygame.transform.scale(self.image, stone_size)
        self.rect = self.image.get_rect()
        self.rect.left = (col_num * stone_width) + 400
        self.rect.top = (row_num * stone_height) + 160

    def draw(self):
        screen.blit(self.image, self.rect)

    def snap(self):
        self.snap_row()
        self.snap_col()

    def snap_row(self):
        self.rect.top = (self.row_num * stone_height) + 160

    def snap_col(self):
        self.rect.left = (self.col_num * stone_width) + 400
```

Функция main_menu() отрисовывает начальное меню игры, две кнопки: «выход» и «начать игру». При нажатии на кнопку «выход» окно закрывается, при нажатии на «начать игру» будет вызвана функция mode_menu() для выбора режима игры.

```
def main_menu():
    engine = True
    while engine:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                engine = False

        mouse_pos = pygame.mouse.get_pos()
        screen.blit(menubcg, (0,0))

        if start_rect.collidepoint(mouse_pos):
            screen.blit(icon, (275,220))
            screen.blit(icon, (850,220))
        if quit_rect.collidepoint(mouse_pos):
            screen.blit(icon, (420,315))
            screen.blit(icon, (690,315))

        screen.blit(start_text, (350,200))
        screen.blit(quit_text, (500,300))

        pygame.display.update()

        if start_rect.collidepoint(mouse_pos) and pygame.mouse.get_pressed()[0]:
            screen.blit(menubcg, (0, 0))
            mode_menu()


        if quit_rect.collidepoint(mouse_pos) and pygame.mouse.get_pressed()[0]:
            engine = False
            pygame.quit()
```

Функция mode_menu() отрисовывает начальное меню игры, две кнопки: «режим по времени» и «режим по очкам». При нажатии на любую кнопку вызывается функция game(), в нее передается значение, соответствующее режиму игры.

```
def mode_menu():
    score_rect = score_mode.get_rect(topleft = (350, 400))
    time_rect = time_mode.get_rect(topleft = (370, 300))
    back_rect = back_button.get_rect(topleft = (0, 0))

    engine = True
    while engine:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                engine = False

        mouse_pos = pygame.mouse.get_pos()
        screen.blit(menubcg, (0,0))

        if score_rect.collidepoint(mouse_pos):
            screen.blit(icon, (275,420))
            screen.blit(icon, (850,420))

        if time_rect.collidepoint(mouse_pos):
            screen.blit(icon, (320,315))
            screen.blit(icon, (815,315))

        if back_rect.collidepoint(mouse_pos):
            screen.blit(icon, (225, 20))
        screen.blit(back_button, (10, 0))
        screen.blit(score_mode, (350, 400))
        screen.blit(time_mode, (390, 300))


        pygame.display.update()

        if score_rect.collidepoint(mouse_pos) and pygame.mouse.get_pressed()[0]:
            screen.blit(menubcg, (0, 0))
            mode = 1
            game(moves, score, mode)

        if time_rect.collidepoint(mouse_pos) and pygame.mouse.get_pressed()[0]:
            screen.blit(menubcg, (0, 0))
            mode = 2
            game(moves, score, mode)

        if back_rect.collidepoint(mouse_pos) and pygame.mouse.get_pressed()[0]:
            main_menu()
```

Функция game() отрисовывает игровую доску, камни на ней и, в зависимости от режима игры, отрисовывает таймер который ведет отсчет времени от 100 секунд до 0. По истечению 100 секунд игра завершится, появится окно «игра окончена», для выхода в главное меню игры нужно нажать «Enter».

Режим по очкам: игра продолжается до момента, когда игрок наберет 100 очков. По набору 100 очков игроком игра завершится, появится окно «вы выйграли», для выхода в главное меню игры нужно нажать «Enter».

```
def game(moves, score, mode):

    clicked_stone = None
    swapped_stone = None
    click_x = None
    click_y = None

    clock = pygame.time.Clock()
    engine = True

    while engine:
        matches = set()

        if mode == 1:
            if score >= 100:
                ok = False
                while not ok:
                    for event in pygame.event.get():
                        screen.blit(menubcg, (0,0))
                        screen.blit(eframe, (345, 270))
                        you_won = font.render('You Won', True, 'Black')
                        screen.blit(you_won, (448, 320))
                        if event.type == pygame.KEYDOWN:
                            if event.key == pygame.K_RETURN:
                                ok = True
                                main_menu()
                        pygame.display.update()


        if mode == 2:
            time_left = 100000
            time_passed = pygame.time.get_ticks()
            time_left -= time_passed
            screen.blit(tframe, (517, 20))
            text = font.render("{}".format(time_left//1000), True, 'Black')
            screen.blit(text, (545, 25))
            if time_left <= 0:
                ok = False
                while not ok:
                    for event in pygame.event.get():
                        screen.blit(menubcg, (0, 0))
                        screen.blit(eframe, (345, 270))
                        you_won = font.render('Game Over', True, 'Black')
                        screen.blit(you_won, (388, 320))
                        if event.type == pygame.KEYDOWN:
                            if event.key == pygame.K_RETURN:
                                ok = True
                                main_menu()
                        pygame.display.update()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                engine = False
                pygame.quit()


            if clicked_stone is None and event.type == pygame.MOUSEBUTTONDOWN:

                for row in board:
                    for stone in row:
                        if stone.rect.collidepoint(event.pos):

                            clicked_stone = stone

                            click_x = event.pos[0]
                            click_y = event.pos[1]

            if clicked_stone is not None and event.type == pygame.MOUSEMOTION:

                distance_x = abs(click_x - event.pos[0])
                distance_y = abs(click_y - event.pos[1])

                if swapped_stone is not None:
                    swapped_stone.snap()

                if distance_x > distance_y and click_x > event.pos[0]:
                    direction = 'left'
                elif distance_x > distance_y and click_x < event.pos[0]:
                    direction = 'right'
                elif distance_y > distance_x and click_y > event.pos[1]:
                    direction = 'up'
                else:
                    direction = 'down'
```

```
    if direction in ['left', 'right']:
        clicked_stone.snap_row()
    else:
        clicked_stone.snap_col()

    if direction == 'left' and clicked_stone.col_num > 0:

        swapped_stone = board[clicked_stone.row_num][clicked_stone.col_num - 1]

        clicked_stone.rect.left = (clicked_stone.col_num * stone_width - distance_x) + 400
        swapped_stone.rect.left = (swapped_stone.col_num * stone_width + distance_x) + 400

        if clicked_stone.rect.left <= (swapped_stone.col_num * stone_width + stone_width / 4) + 400:
            swap(clicked_stone, swapped_stone)
            matches.update(match_three(clicked_stone))
            matches.update(match_three(swapped_stone))
            moves += 1
            clicked_stone = None
            swapped_stone = None

    if direction == 'right' and clicked_stone.col_num < board_width / stone_width - 1:

        swapped_stone = board[clicked_stone.row_num][clicked_stone.col_num + 1]
        clicked_stone.rect.left = (clicked_stone.col_num * stone_width + distance_x) + 400
        swapped_stone.rect.left = (swapped_stone.col_num * stone_width - distance_x) + 400

        if clicked_stone.rect.left >= (swapped_stone.col_num * stone_width - stone_width / 4) + 400:
            swap(clicked_stone, swapped_stone)
            matches.update(match_three(clicked_stone))
            matches.update(match_three(swapped_stone))
            moves += 1
            clicked_stone = None
            swapped_stone = None

    if direction == 'up' and clicked_stone.row_num > 0:

        swapped_stone = board[clicked_stone.row_num - 1][clicked_stone.col_num]

        clicked_stone.rect.top = (clicked_stone.row_num * stone_height - distance_y) + 160
        swapped_stone.rect.top = (swapped_stone.row_num * stone_height + distance_y) + 160

        if clicked_stone.rect.top <= (swapped_stone.row_num * stone_height + stone_height / 4) + 160:
            swap(clicked_stone, swapped_stone)
            matches.update(match_three(clicked_stone))
            matches.update(match_three(swapped_stone))
            moves += 1
            clicked_stone = None
            swapped_stone = None

    if direction == 'down' and clicked_stone.row_num < board_height / stone_height - 1:

        swapped_stone = board[clicked_stone.row_num + 1][clicked_stone.col_num]
        clicked_stone.rect.top = (clicked_stone.row_num * stone_height + distance_y) + 160
        swapped_stone.rect.top = (swapped_stone.row_num * stone_height - distance_y) + 160

        if clicked_stone.rect.top >= (swapped_stone.row_num * stone_height - stone_height / 4) + 160:
            swap(clicked_stone, swapped_stone)
            matches.update(match_three(clicked_stone))
            matches.update(match_three(swapped_stone))
            moves += 1
            clicked_stone = None
            swapped_stone = None

if clicked_stone is not None and event.type == pygame.MOUSEBUTTONUP:

    clicked_stone.snap()
    clicked_stone = None
    if swapped_stone is not None:
        swapped_stone.snap()
```