BIT ADVENTURE

Максим Боткин и Иван Ефимов

ВВЕДЕНИЕ

Идея проекта:

Разработка игры на PyGame

Суть проекта:

Создать 2D-платформер на прохождение с уникальными уровнями

```python
# стартовый экран, из которого можно перейти в настройки и саму игру
def start_screen(level_numb, user_id):
    level_number = level_numb

    # фон стартового экрана
    fon = pygame.transform.scale(load_image('start_fon.png'), (width, height))
    screen.blit(fon, (0, 0))

    # выводим кнопки настроек и играть на экран
    screen.blit(buttons['settings_btn'], (750, 0))
    screen.blit(buttons['play_btn'], (170, 240))

    pygame.mixer.music.set_volume(music_level)

    # цикл стартового экрана
    while True:
        for event in pygame.event.get():
            # screen.blit(cursor_img, coord)
            if event.type == pygame.QUIT:
                # выходим из игры, если пользователь закрыл программу
                terminate()
            if event.type == pygame.MOUSEMOTION:
                pass
            if event.type == pygame.MOUSEBUTTONDOWN:
                # проверяем, куда нажал пользователь
                if 750 <= event.pos[0] <= 800 and 0 <= event.pos[1] <= 50:
                    # проигрываем звук нажатия кнопки
                    menu_up.set_volume(sound_effects_level)
                    menu_up.play()
                    settings_screen(level_numb, user_id)
                elif 130 <= event.pos[0] <= 630 and 200 <= event.pos[1] <= 575:
                    # проигрываем звук нажатия кнопки
                    menu_up.set_volume(sound_effects_level)
                    menu_up.play()
                    main_screen(level_number, user_id)
        keys = pygame.key.get_pressed()
        if keys[pygame.K_ESCAPE]:
            settings_screen(level_numb, user_id)
        # обновляем экран
        pygame.display.flip()
        clock.tick(FPS)
```

СОЗДАНИЕ РАЗИЧНЫХ ЭКРАНОВ

Стартовый, игровой и экран настроек

Вывод различных экранов, в зависимости от того, что игрок сейчас делает. Стартовый экран даёт возможность начинать игру, игровой экран выводит саму игру, а экран настроек позволяет пользователю изменять параметры звука и эффектов.

```python
# класс игрока
class Player(pygame.sprite.Sprite):
    global x_coord, y_coord

    # инициализируем
    def __init__(self, x_coord, y_coord, level_num, user_id):
        super().__init__(player_group, all_sprites)
        self.image = defoltplace
        self.rect = self.image.get_rect().move(x_coord, y_coord)
        self.level_num = level_num
        self.user_id = user_id

    # функция отвечающая за передвижения персонажа
    def update(self, x_coord, y_coord, jump):
        global animation, left, right, jump_height, is_flying
        # если игрок сталкивается с шипами, то появляется экран поражения
        if pygame.sprite.spritecollideany(self, thorns_group):
            self.kill()
            lose_screen(self.level_num, self.user_id)
        # если игрок сталкивается с лавой, то появляется экран поражения
        if pygame.sprite.spritecollideany(self, water_group):
            self.kill()
            lose_screen(self.level_num, self.user_id)
        # если игрок сталкивается со звездой, то появляется экран победы
        if pygame.sprite.spritecollideany(self, finish_group):
            self.kill()
            win_screen(self.level_num, self.user_id)
        # если игрок провалится, то также появляется экран поражения
        if y_coord > 600:
            self.kill()
            lose_screen(self.level_num, self.user_id)
        # игрок падает, не столкнётся с блоком
        if pygame.sprite.spritecollideany(self, tiles_group) is None and not jump:
            is_flying = True
        if pygame.sprite.spritecollideany(self, tiles_group):
            rect = pygame.sprite.spritecollideany(self, tiles_group).rect
            if rect[0] - x_coord >= 40:
                is_flying = True
        # игрок летит с анимацией падения
        if is_flying:
            self.image = fly[0]
            self.rect = self.image.get_rect().move(x_coord, y_coord)
            return
        if pygame.sprite.spritecollideany(self, tiles_group) or jump:
```

СОЗДАНИЕ ПЕРСОНАЖА РЕАЛИЗАЦИЯ ФИЗИКИ

Персонаж, ландшафт, физика

Создание анимированного персонажа с возможностью ходить и прыгать, добавление в игру ландшафта, препятствий, финишной прямой. Реализована физика по отношению к персонажу, смерть при столкновении с преградой, возможность ходить по безопасной земле. При смерти показывается экран поражения, при удачном столкновении с финишной прямой – экран победы.

```python
# Окно входа в аккаунт
class Login_window(PyQt5.QtWidgets.QDialog):
    global level_number

    def __init__(self):
        super().__init__()
        self.widget = None
        uic.loadUi('login_qt.ui', self)
        self.setWindowTitle('Вход/регистрация в аккаунт')
        self.setFixedSize(self.size())

        # Скрытие строки с паролем
        self.password.setEchoMode(2)

        # Обработка кнопок входа и регистрации
        self.login_button.clicked.connect(self.login_in)
        self.sign_up_button.clicked.connect(self.sign_up)

    # Функция открытия главного окна
    def game_starting(self, level_num, user_id):
        Login_window.close(self)
        level_number = level_num[0][0]
        start_screen(level_number, user_id)

    # Функция входа в аккаунт
    def login_in(self):
        # Подключение к базе данных users.db
        con = sqlite3.connect("users.db")
        cursor = con.cursor()
        # Считывание логина и пароля
        login = self.login.text()
        password = self.password.text()
        # Проверка введены ли логин и пароль
        if not login or not password:
            QMessageBox.critical(self, 'Ошибка!', 'Вы заполнили не все поля!', QMessageBox.C
        else:
            # Хеширование пароля с помощью библиотеки hashlib
            password = bytes(password, 'utf-8')
            hash_password = hashlib.sha1(password).hexdigest()
            # Ищем пользователя в таблице users в базе данных
            result_of_execute = cursor.execute('SELECT login, password FROM user WHERE login
                                               (login, hash_password))
            # Проверка - найден ли пользователь
            if result_of_execute.fetchall():
```

ВХОД / РЕГИСТРАЦИЯ

Окно входа в аккаунт

При запуске игры появляется окно с авторизацией в аккаунт пользователя. Это даёт возможность сохранять данные определенных пользователей.

ЗАКЛЮЧЕНИЕ

Итоги и доработка

В итоге всей работы мы создали 2D-платформер, в котором можно проходить определенные уровни. Нашли новые возможности PyGame и улучшили свои навыки.

Возможности по доработке:

1. Добавить больше уровней
2. Индивидуализировать уровни