# Техническое задание по проекту

### 1. Общие сведения

Название проекта: Elemental Escape

Авторы: Артём, Тимофей

### 2. Описание идеи

Проблема и решение:

Наш проект — это 2D-платформер-головоломка, вдохновленный классикой жанра «Огонь и Вода».

Суть игры:

Игроки управляют двумя персонажами (стихии Огня и Воды) на одном экране. Цель — провести обоих героев к финишу, преодолевая препятствия, соответствующие их природе, и решая совместные головоломки (рычаги, кнопки, лифты).

### 3. Описание реализации и архитектура

Проект разрабатывается на языке **Python** с соблюдением принципов **ООП**. Код разбит на независимые модули для обеспечения чистоты и расширяемости.

**Архитектурные решения:**

1. **Управление состояниями:** Используется класс arcade.Window как контейнер, внутри которого переключаются экраны (Меню, Игра, Пауза) через наследование от arcade.View.
2. **Data-Driven Design (Уровни):** Логика игры отделена от контента. Уровни создаются во внешнем редакторе **Tiled**, сохраняются в JSON и загружаются универсальным классом GameView. Это позволяет добавлять новые уровни без изменения кода.
3. **Система сохранений:** Реализована через сериализацию данных в формат **JSON**. Класс-менеджер GameData сохраняет прогресс (открытые уровни, рекорды времени) и настройки в локальный файл, что обеспечивает персистентность данных между сессиями.

**Основные классы:**

* GameView: Основной цикл игры, загрузка карты и управление физикой.
* Player (arcade.Sprite): Класс персонажа с уникальными параметрами (скорость, высота прыжка).
* InteractableObject: Базовый класс для механизмов (рычаги, кнопки), использующий полиморфизм.

### 4. Технологии и библиотеки

Для запуска проекта необходимы:

* **Python 3.13+**
* **arcade:** Основная библиотека для рендеринга, звука и управления окном.
* **json:** Встроенный модуль для работы с файлами сохранений.

### 5. Функционал приложения

Ниже приведен список функций и способ их реализации:

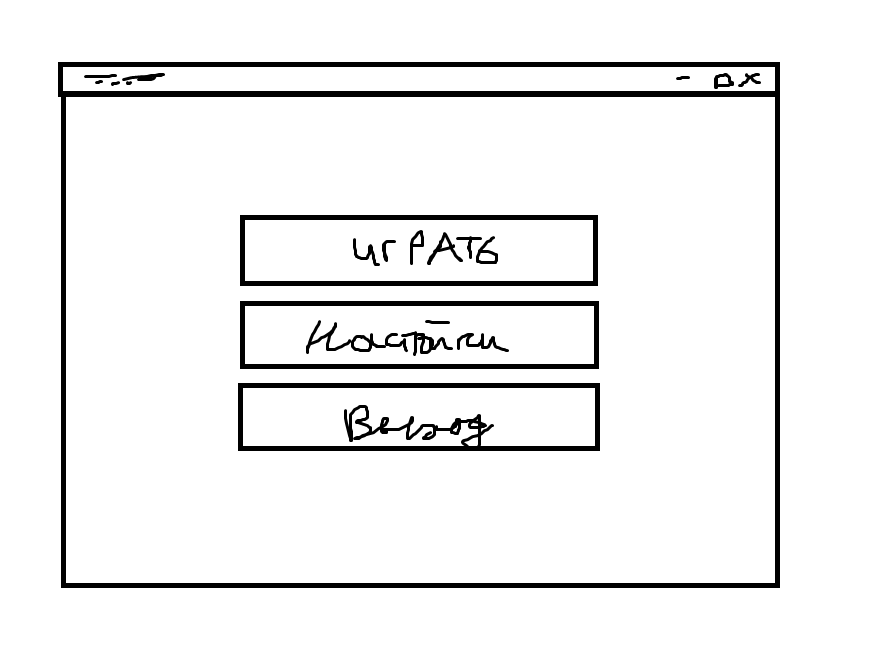
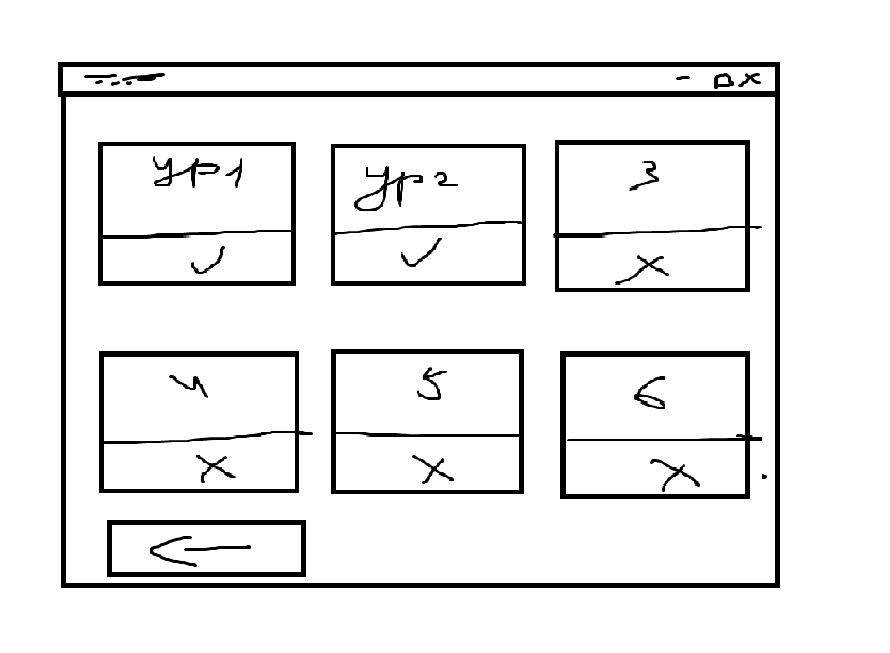
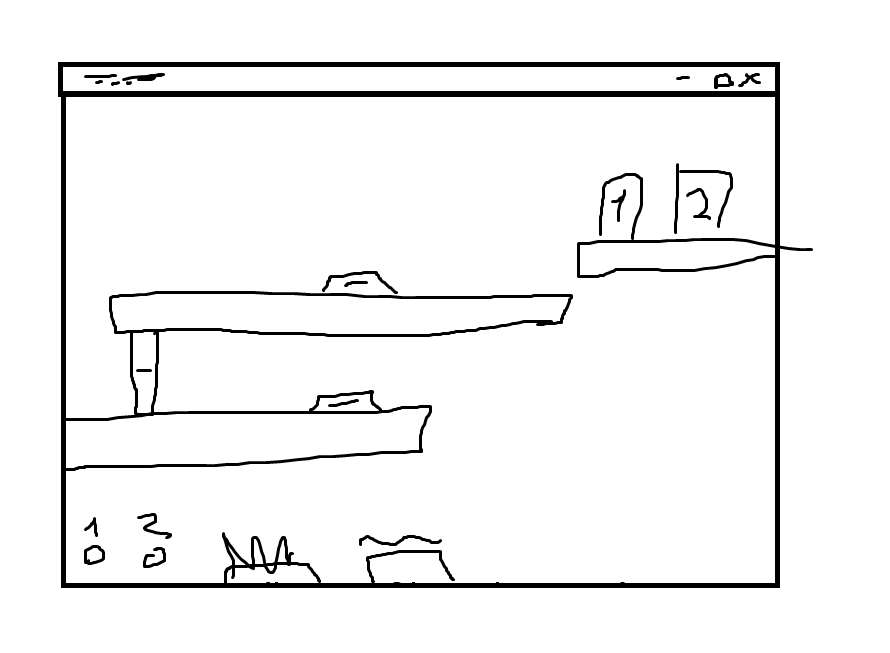
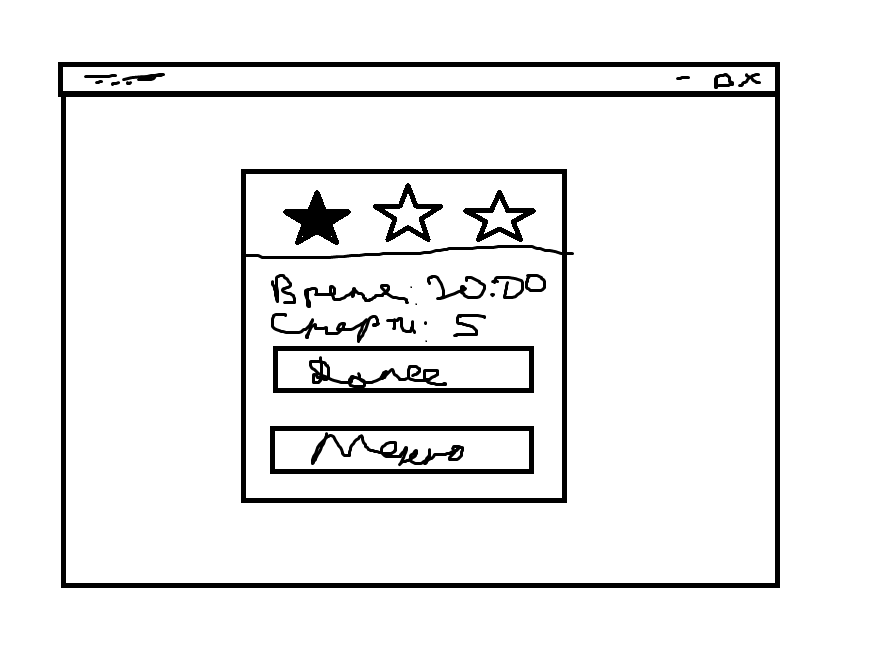
1. **Движение персонажей:**
   * *Реализация:* Использование движка arcade.PhysicsEnginePlatformer. Обработка ввода с клавиатуры (WASD для Воды, Стрелки для Огня) в методе on\_key\_press.
2. **Загрузка уровней:**
   * *Реализация:* Функция загрузки тайловых карт arcade.load\_tilemap с разделением на слои ("Walls", "Hazards", "Decorations"). Карты хранятся в папке assets/maps.
3. **Взаимодействие с механизмами:**
   * *Реализация:* Проверка коллизий arcade.check\_for\_collision\_with\_list. При контакте с рычагом изменяется его спрайт (анимация переключения) и удаляется коллизия у связанной двери.
4. **Сохранение и загрузка прогресса:**
   * *Реализация:* Класс GameData с методами save() (запись словаря в файл savegame.json через json.dump) и load() (чтение и восстановление состояния).
5. **Смерть и рестарт:**
   * *Реализация:* Проверка пересечения хитбокса игрока со слоем "Lava" (для Воды) или "Water" (для Огня). При смерти уровень перезапускается методом setup().
6. **Игровое меню и интерфейс:**
   * *Реализация:* Классы MenuView и PauseView с отрисовкой кнопок и обработкой кликов мыши (on\_mouse\_press).

### 6. План реализации проекта (Timeline)

Процесс разбит на этапы с дедлайнами:

| **Дата (Дедлайн)** | **Функция / Задача** | **Ответственный** |
| --- | --- | --- |
| **День 1-3** | Настройка Git-репозитория. Структура проекта. Базовое меню (MenuView) и класс окна. | Оба |
| **День 4** | Класс Player и интеграция PhysicsEnginePlatformer (гравитация, прыжки). | Участник А |
| **День 5** | Создание тестовой карты в Tiled. Загрузка карты в игру (GameView). | Участник Б |
| **День 7 (Прототип)** | Реализация механики смерти от препятствий и рестарта. **Сдача прототипа.** | Оба |
| **День 9** | Логика интерактивных объектов (рычаги, двери) и переходы между уровнями. | Участник А |
| **День 11** | Дизайн уровней 2-5 (Tiled). Реализация класса GameData (JSON сохранения). | Участник Б |
| **День 13** | Финальная сборка, звуки, чистка кода (PEP8), написание документации. | Оба |
| **День 14** | Подготовка к защите (презентация, проверка репозитория). | Оба |

### 7. Описание интерфейса (Скриншоты/Макеты)

* **Рис 1.** Главное меню (Кнопки "Играть", "Настройки", "Выход").
* **Рис 2.** Меню выбора уровня (Список уровней) 
* **Рис 3.** Схема игрового уровня (Платформы, персонажи, рычаг, дверь, финиш).
* **Рис 4.** Экран победы (Время прохождения, кнопка "Далее").

**Игровые механики**

Игра строится на кооперативном взаимодействии. Игроки зависят друг от друга и не пройдут уровень в одиночку.

**Основные механики:**

* **Управление двумя персонажами:**
  + **Огонь:** управляется стрелками на клавиатуре. Есть иммунитет к красным озерам (лаве) на уровне, но погибает в воде. Может собирать только красные кристаллы.
  + **Вода:** управляется клавишами WASD. Имеет иммунитет к воде и погибает в лаве. Собирает синие кристаллы.
  + На уровне также есть **зелёная кислота**, смертельная для всех.
* **Интерактивные объекты:**
  + **Рычаги:** переключают состояние дверей или платформ (вкл/выкл). Состояние сохраняется после переключения.
  + **Нажимные плиты:** активируют механизм (например, убирают стену) **только** когда один из игроков стоит на кнопке. Это требует координации: пока один держит дверь, второй проходит.
  + **Подвижные ящики:** игроки могут толкать ящики, чтобы зажать плиту или забраться на высокую платформу.
  + **Аэродинамические трубы:** зоны с измененной гравитацией, позволяющие персонажам подниматься на недоступные высоты.
  + **Мобильные платформы:** элементы ландшафта, совершающие циклические движения (горизонтальные/вертикальные). Используются для транспортировки игроков через широкие препятствия.
* **Условия победы:**
  + Оба персонажа должны добраться до своих дверей (Огонь — к красной, Вода — к синей).
* **Система подсчета очков (от этого зависит итоговое количество звезд от 1 до 3):**
  + Таймер уровня (чем быстрее, тем лучше).
  + Количество собранных кристаллов.

При **смерти** одного из персонажей засчитывается поражение, а игру можно начать сначала.