**Пояснительная записка к игре “Звёздные войны. Эпизод I: Скрытая угроза”**

**Автор: Евгения Аршинова и Руслан Рзаев**

***Идея игры:*** игра по культовому фильму “Звёздные войны Эпизод I: Скрытая угроза” с уровнями, повторяющими сюжет ключевых событий фильма.

***Часть Руслана: стартовое окно, первый уровень***

**Пояснительная записка к проекту "Star Dash"**

**1. Название проекта и идея**

**"Star Dash"** — это уровень, сочетающий в себе элементы популярных проектов, таких как **Flappy Bird** и **Geometry Dash**, с добавлением собственных оригинальных задумок. Главной особенностью игры являются сюжетные кат-сцены, игровые события и ивенты, которые делают процесс прохождения разнообразным.

Игра предлагает пользователю не только преодолевать препятствия, но и следить за ресурсами, такими как топливо, что добавляет стратегический элемент.

**2. Задачи проекта**

Основной целью разработки игры является создание игрового процесса, который будет более менее интересным

Ключевые задачи проекта:

1. **Играбельность** — обеспечить комфортный игровой процесс с интересными механиками.
2. **Залипательность** — сделать игру захватывающей.
3. **Экшен** — добавить динамичные события, такие как побеги от монстров, уклонение от препятствий и управление ресурсами.

**3. Инструменты и технологии**

Проект реализован на языке программирования **Python**, который обеспечивает мощный функционал и гибкость разработки.  
Для реализации ключевых аспектов игры использовались следующие технологии:

* **Pygame** — библиотека для разработки 2D-игр, которая была использована для создания игрового процесса, управления анимациями, обработкой пользовательского ввода и отображением графики.
* **SQLite3** — библиотека для работы с базами данных. Используется для хранения достижений игроков, их прогресса и другой игровой информации.

**4. Описание игрового процесса**

Игрок управляет космическим кораблем, который должен преодолевать опасности и избегать столкновений с препятствиями. На пути игрока появляются монстры, препятствия и сюжетные вставки.  
Основные особенности игрового процесса:

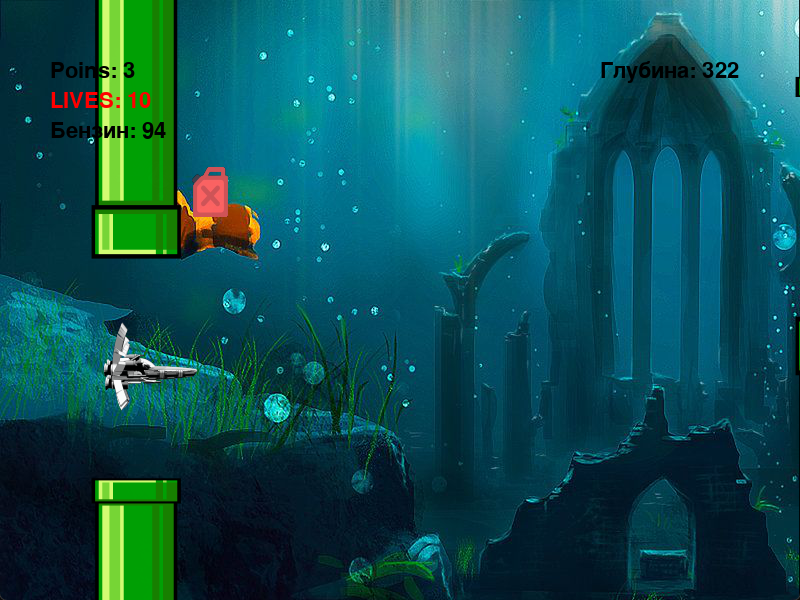
* **Уровень топлива**: Игрок должен следить за индикатором топлива. Если оно закончится, корабль не сможет продолжать движение. Это добавляет элемент стратегии.
* **Кат-сцены**: Сюжетные вставки погружают игрока в историю.
* **Механика достижений**: Игроку предоставляются награды за выполнение определенных условий или преодоление сложных участков уровня.

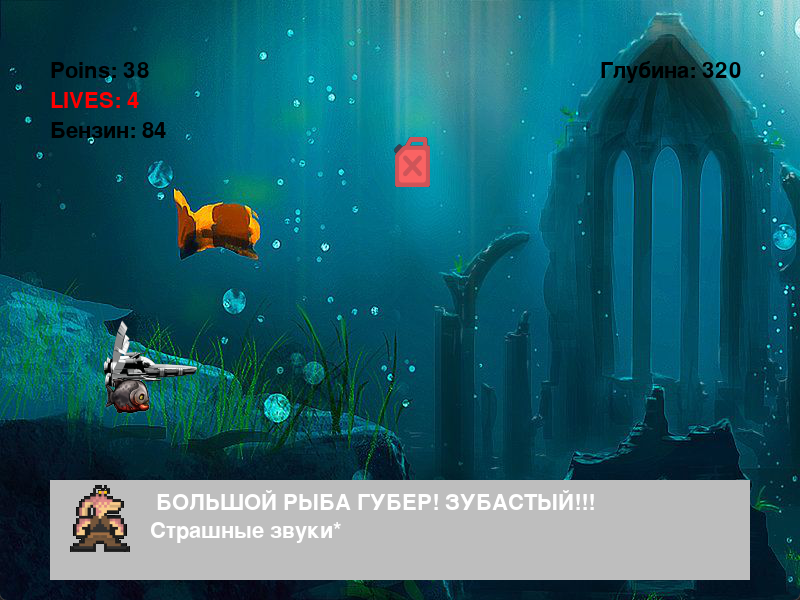
Пример достижения:

* "Рыбы дуреют с этой прикормки" — достижение за успешное прохождение уровня, связанного с морскими обитателями.

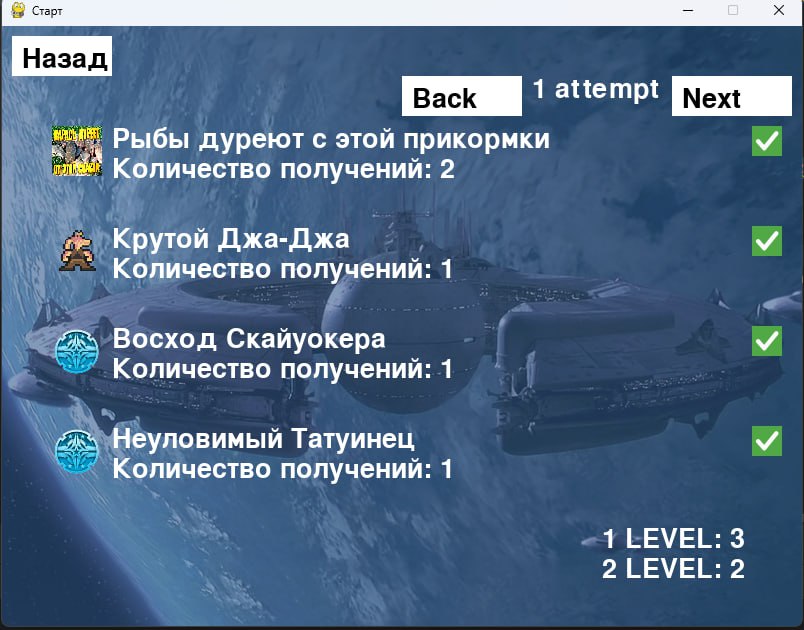
**5. Во время разработки использованы модули**

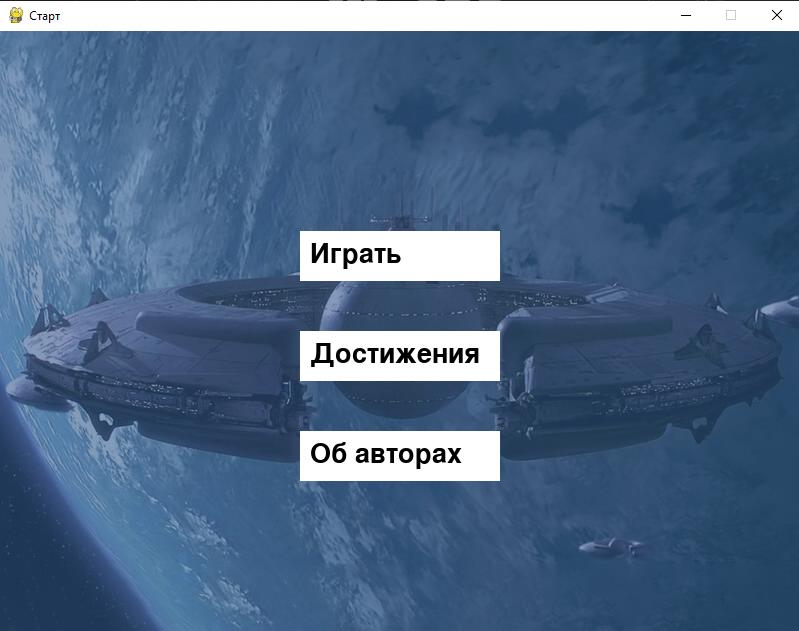
* **Pygame**: для реализации графики, физики и игрового процесса.
* **SQLite3**: для работы с базой данных, хранения достижений и прогресса игрока.

Скриншоты игры: 



Стартовое окно:





***Часть Евгении: второй уровень, межуровневые сцены, финальное окно, сборка проекта***

***Структура проекта:***

* **utils.py** – файл с абстрактными функциями;
* **config.py** – константы, изображения и звуки программы;
* **Scene2** – класс второго уровня игры;
* **final\_screen.py** – финальное окно;
* **episode\_text** – функция показа экрана с межуровневым текстом;
  + **game.db** – база данных с тремя таблицами уровней (по одной таблице на уровень);
  + **Achievement** – класс всплывающего достижения;
  + **level2\_objs** – папка с классами объектов второго уровня;
* **data** - папка с подпапками **img**(изображения) и **sound**(звуки).

***Использованные технологии и библиотеки:***

Pygame, os, random, datetime, sys, sqlite3.

***Основные классы:***

**Scene2** – класс второго уровня, которому при инициализации передаются жизни игрока с предыдущего уровня. Главный метод класса **gameplay** создаёт объекты уровня и запускает главный игровой цикл, в котором игрок должен уворачиваться от астероидов и убивать вражеские корабли, чтобы живым добраться до главного командного центры и уничтожить его. В этом цикле отрисовываются все объекты и спрайты уровня, отслеживаются нажатия на клавиатуру:

* [A] / [Right Arrow] – движение игрока вправо;
* [D] / [Left Arrow] – движение игрока влево;
* [Space] – стрельба лазерами.

При победе или поражении игра останавливается и игроку сообщаются его результаты. При переходе на следующую сцену сохраняется результат (победа или поражение) игрока на втором уровне и метод завершает свою работу.

**episode\_text** – функция, которой передаются на вход тексты для отображения. Текст показывается между всеми уровнями, объясняя события, которые происходят между эпизодами, реализованными в геймплее. Текст и музыку можно остановить, включить заново или просто пропустить, нажав на соответствующие кнопки.

**Achievement** – класс всплывающего достижения, который отображает название, описание и иконку только что заработанного достижения

***Основные решения:***

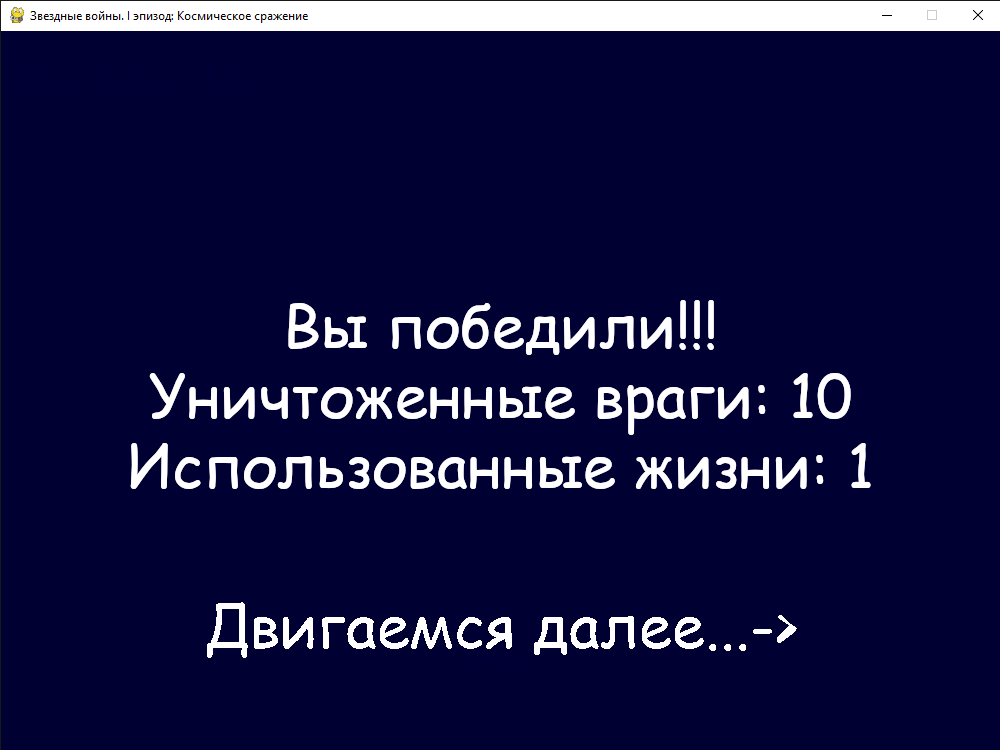
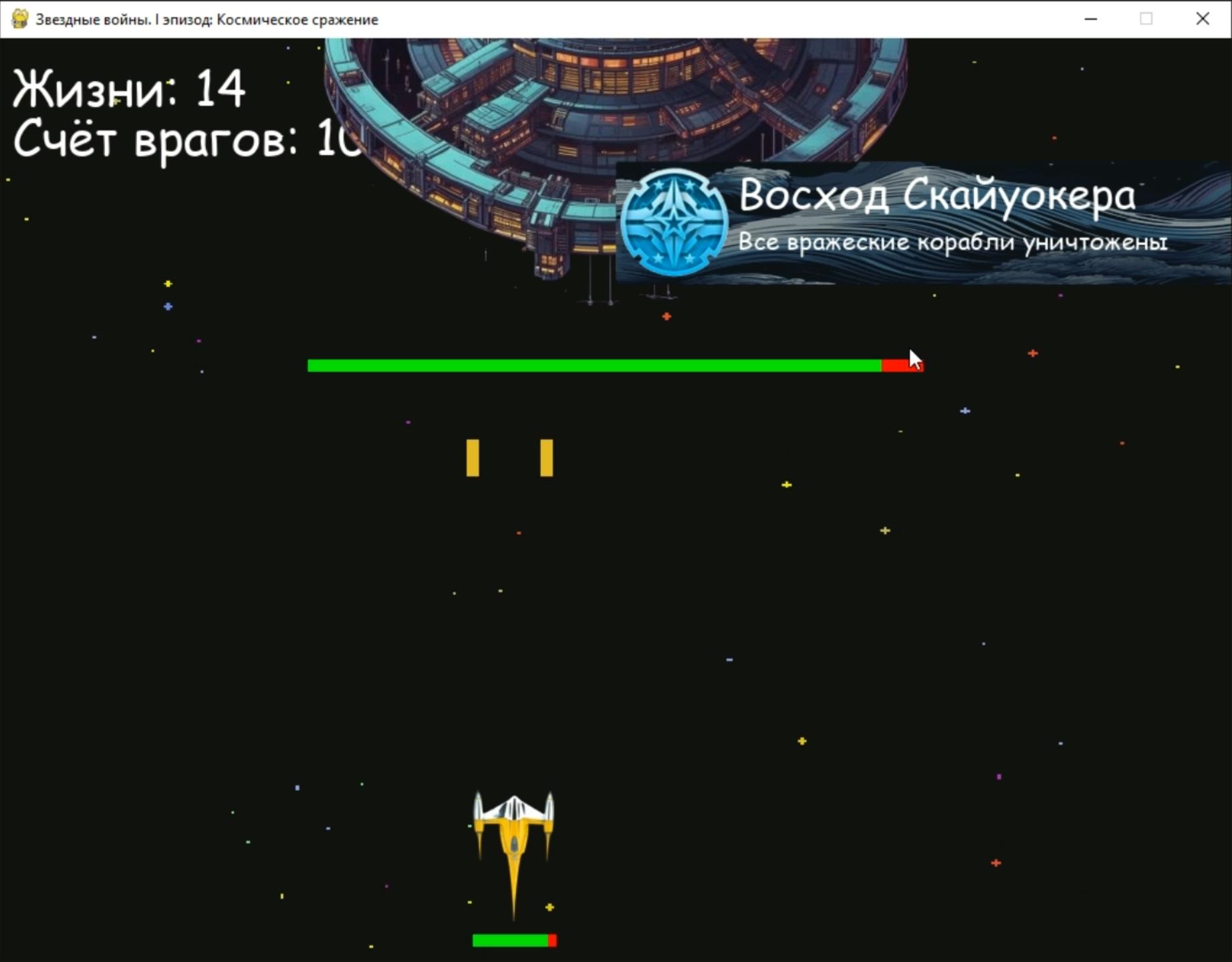
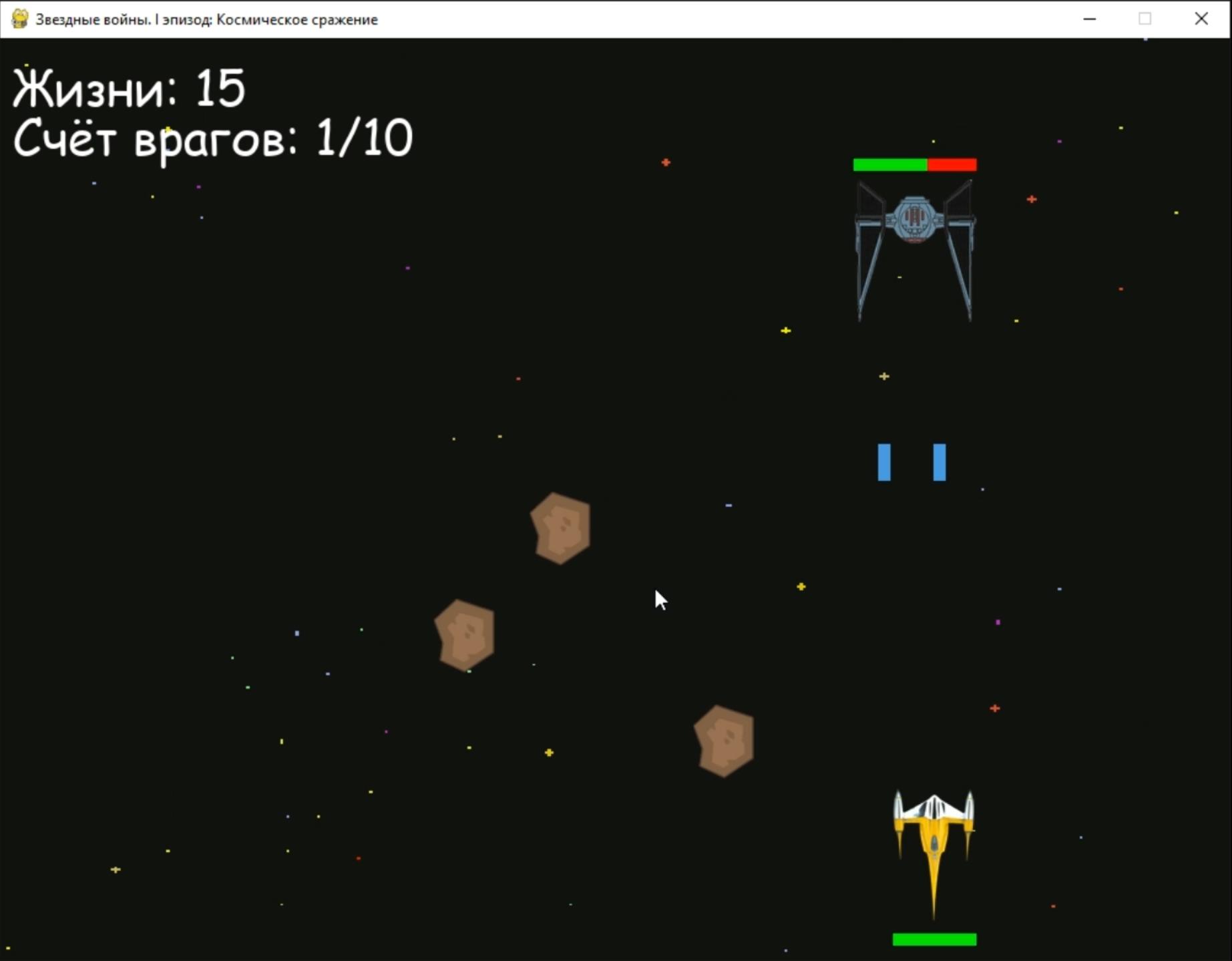
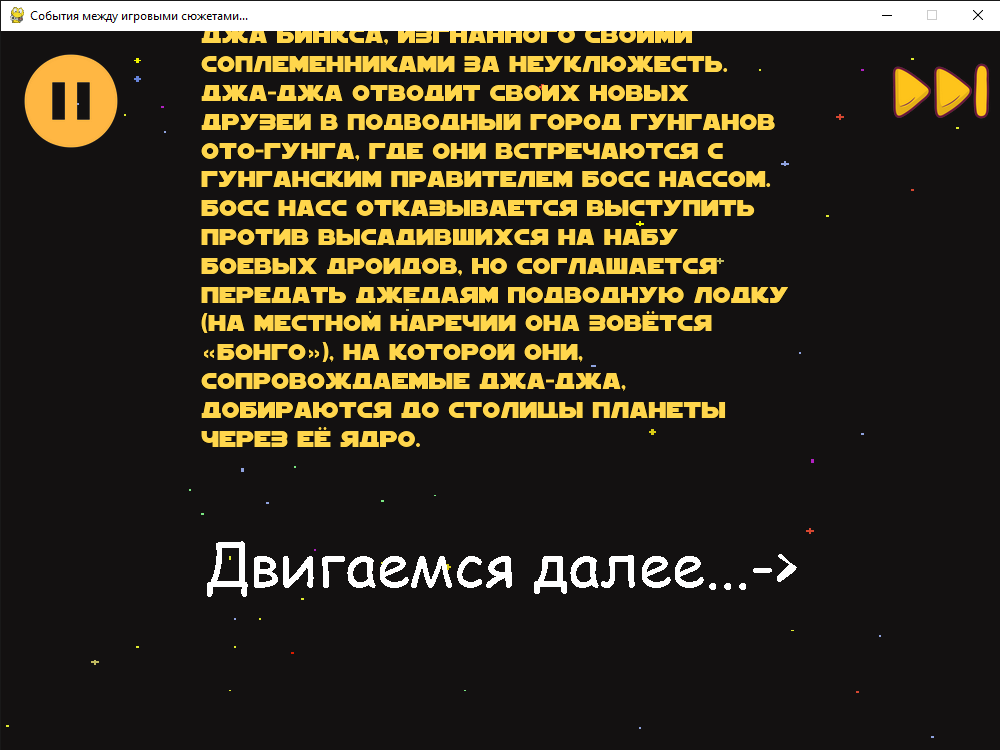
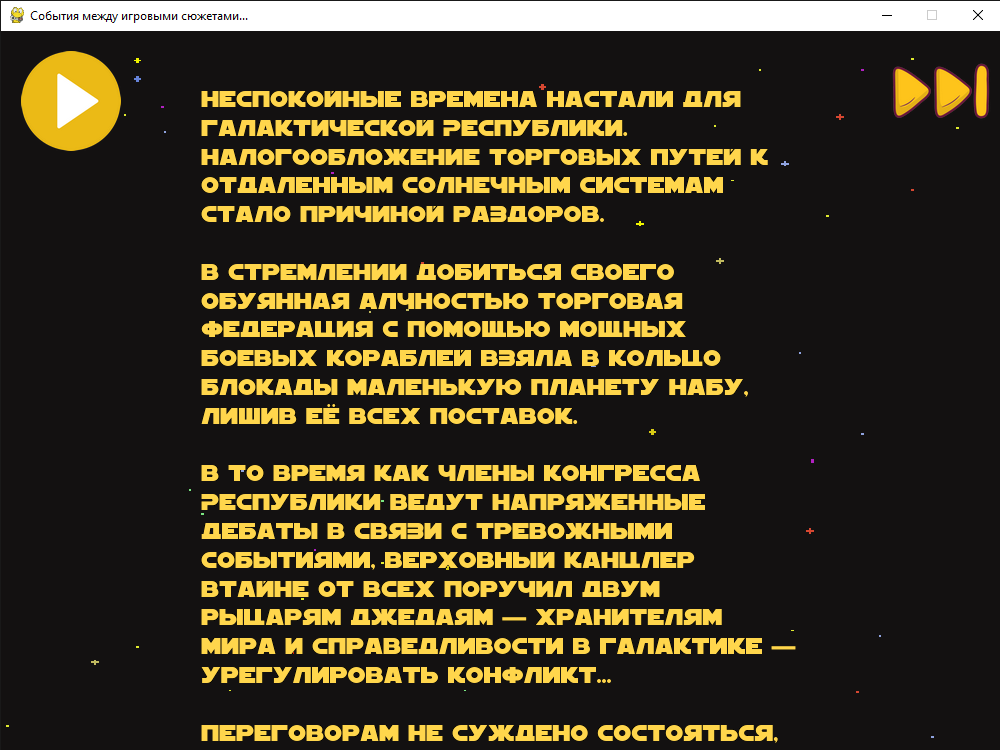
Проект реализован при помощи языка программирования Python. Главная использованная библиотека игры - Pygame. Все уровни представляют собой классы с атрибутами количества жизней, текущей попытки (которая задаётся с помощью вычисления длины базы данных), окна, где будет отображаться уровень, и успеха/неуспеха игрока, а также методом ***gameplay***,в котором происходит игровой процесс. В файле **main.py**работает цикл, в котором идёт поэтапная инициализация уровней и вызов их метода ***gameplay***, после выполнения которого круг цикла продолжается (если ***success == True)*** или начинается следующий. Жизни и успех каждого уровня передаются последующим сценам при инициализации. Если игрок умер, то в атрибут уровня ***success*** сохраняется значение ***False****,* человеку сообщается о его проигрыше, главный цикл всей игры переходит на следующий круг и игра начинается заново, прямо со стартового окна.

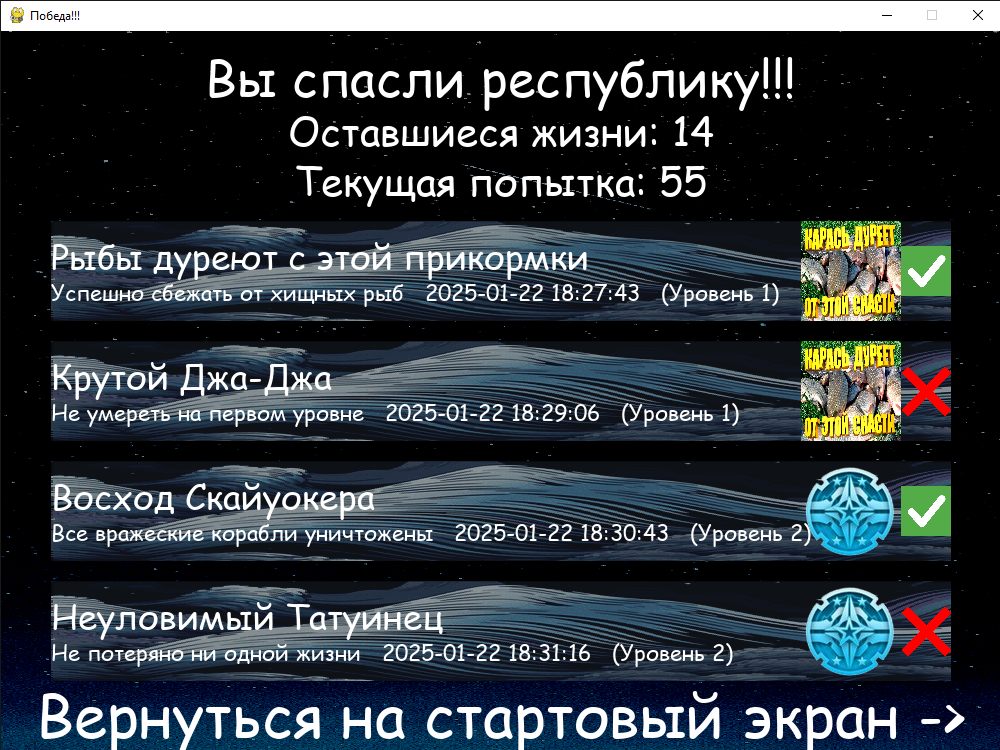
На межуровневых сценах отображаются картинки с текстом сюжета, который происходит между игровыми уровнями. При нажатии на кнопку паузы текст и музыка останавливаются (в коде их скорость становится равной 0), при нажатии на кнопку проигрывания музыка и текст (его скорость возвращается к значению большему чем 0) запускается снова, а при нажатии на кнопку “пропустить” или на текст *“Двигаемся далее->”* главный цикл функции завершается и начинается следующий уровень.

На последнем уровне игры в главном методе класса **Scene2.*gameplay*** в циклесоздаются экземпляры класса игрока **Player**, **SpaceStation,** группа спрайтов взрывов **explosion\_group**, список с объектами вражеских кораблей **enemies**, и список с объектами астероидов **asteroids**. У объектов игрока и врагов общий родитель - **Ship.** Стреляющие лазеры и игрока и врагов отрисовываются в их собственных методах ***move\_lasers***, где происходит отслеживание столкновения с другими объектами. У игрока, вражеских кораблей и космической станции есть шкала здоровья, которая с каждым столкновением о лазеры становится менее зелёной и заполняется красным цветом. Здоровье у игрока и вражеских объектов при соприкосновении с лазерами уменьшается на 10 единиц и, если он становится равным нулю, объект врага удаляется из списка, где он находится, и проигрывается анимация взрыва со звуком. Если же игрок столкнулся с вражеским объектом, то его здоровье уменьшается на 5 (если это был астероид) или 10 (если был вражеский корабль) единиц, а враг уничтожается и опять включается анимация взрыва со звуком. Все объекты отрисовываются в главном цикле метода, который как бы делает “паузу” при победе или проигрыше игрока с помощью внутреннего цикла, который завершается при нажатии игроком на текст *“Двигаемся далее*. Игра, таким образом, переключается на следующую сцену. В главном цикле метода идёт счётчик времени, и когда с начала игры проходит около минуты на экране медленно появляется космическая станция главного командного центра, которая является финальным врагом в игре. При уничтожении всех вражеских кораблей всплывает достижение, название, описание, дата получения и иконка которого записывается в базу данных **game** в таблицу *level2* для последующего отображения. Также, если при окончании уровня, если игрок не потерял ни одной жизни в этой конкретной сцене, ему засчитывается второе достижение. Все заработанные достижения всплывают прямо в момент получения. В конце игры, если какое-либо из доступных достижений не было получено, оно всё равно сохраняется в базу данных и значение в столбце *achieved* (получено достижение или нет) сохраняется как 0. Количество жизней игрока и счёт уничтоженных врагов отображаются на экране и постоянно обновляют свою информацию.

На финальном экране, который реализован функцией **final**\_**screen**, отображается список всех полученных достижений (с их названием, описанием, иконкой, датой получения и зелёной галочкой если достижение было получено, в противном же случае – красным крестиком) за текущую попытку. Все виджеты достижений динамически заполняются информацией из базы данных. При нажатии игроком на текст *“Двигаемся далее->”* подаётся сигнал к завершению главного цикла функции, после чего закончится и выполнение круга самого главного цикла в файле **main.py** и вся игра начнётся сначала.

***Скриншоты программы***

******

******