**НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ КОНФЕРЕНЦИЯ**

**«Шаг в будущее»**

**СОЗДАНИЕ МНОГОПОЛЬЗОВАТЕЛЬСКОЙ КОМПЬЮТЕРНОЙ ИГРЫ «МОРСКОЙ БОЙ ОНЛАЙН» С ВОЗМОЖНОСТЬЮ ОБУЧЕНИЯ В РАЗНЫХ СФЕРАХ**

**Болотов Иван Андреевич**

*ГБОУ Инженерная школа №1581, г. Москва*

## **Секция:** *Информационные технологии (ИУ7)*

**Научный руководитель**: Смирнова Светлана Юрьевна, преподаватель информатики и программирования в *ГБОУ Инженерная школа №1581*

*регистрационный номер*

**Москва – 2025**

**Создание многопользовательской компьютерной игры «Морской бой онлайн» с возможностью обучения в разных сферах**

**АННОТАЦИЯ**

Основная цель моей работы - разработка многопользовательской компьютерной игры “Морской бой онлайн” с особыми правилами, направленными на изучение какой-то информации, что поможет совместить приятное занятие с полезным делом. В последнее время огромное количество людей страдают от игромании. Моя игра призвана помочь им. Не менее важной частью моего проекта является тестирование созданного продукта (например, в школе) и оценка пользы (в динамике), которую он смог принести окружающим.

Перед началом работы была проведена оценка программных средств, с помощью которых возможно реализовать техническую часть данного проекта (игровую логику, графический интерфейс игры, а также сетевую логику, то есть проектирование эффективного сервера, способного обрабатывать множество подключений и выдерживать большие перегрузки, а также всю логику клиент-серверного взаимодействия). В качестве универсального средства для покрытия всех целей программной реализации было решено использовать язык программирования Python в связке с фреймворком Pygame (для создания игровой логики и прорисовки всех графических объектов) и библиотекой Websockets (для реализации веб-сервера и всего клиент-серверного взаимодействия). Также были использованы следующие встроенные в Python библиотеки: asyncio, threading и multiprocessing, которые помогли достичь необходимой производительности как самой игры, так и сервера.

К данному моменту была изучена техническая информация для реализации всех поставленных задач проекта, а также реализована вся игровая логика, логика работы сервера и взаимодействия между сервером и клиентом. В ближайшем будущем мной планируется запуск проекта (например, среди одноклассников) и выявление его ценности на практике.

**СОДЕРЖАНИЕ**

[1. ВВЕДЕНИЕ 5](#_Toc192781140)

[2. АНАЛИЗ И ВЫБОР СРЕДСТВ ДЛЯ ПРОГРАММНОЙ РЕАЛИЗАЦИИ ПРОДУКТА 7](#_Toc192781141)

[2.1 Анализ и сравнение доступных технологий. 7](#_Toc192781142)

[2.2 Обобщение полученной информации, выбор наиболее подходящего ПО 8](#_Toc192781143)

[3. ОСОБЕННОСТИ ПРОГРАММНОЙ РЕАЛИЗАЦИИ ДАННОГО ПРОДУКТА 10](#_Toc192781144)

[3.1 Общая структура проекта 10](#_Toc192781145)

[3.2 Игровая логика, обучающая логика, графика игры 12](#_Toc192781146)

[3.3 Серверная логика и клиент-серверное взаимодействие 14](#_Toc192781147)

[4. РЕЗУЛЬТАТЫ 15](#_Toc192781148)

[5. ЗАКЛЮЧЕНИЕ 19](#_Toc192781149)

[6. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ДРУГИХ РЕСУРСОВ 20](#_Toc192781150)

# 1. ВВЕДЕНИЕ

В последнее время всё большие и большие обороты набирает популярность видеоигр. Почти каждый современный человек имеет возможность использовать современные гаджеты с доступом в интернет. С одной стороны, это хорошо, но с другой - в связи с этим возникают новые глобальные проблемы. Одна из самых значимых - проблема зависимости от игромании. Всемирная организация здравоохранения признала зависимость от компьютерных игр болезнью и классифицировала ее как аддитивное расстройство поведения.

Так, например, в 2022 году по итогам исследования Аналитического центра НАФИ [1] было выявлено, что **60% россиян** играют в видеоигры — регулярно или эпизодически. По сравнению с 2018 годом число играющих увеличилось **более чем в 3 раза**.

Согласно статистике, огромное влияние данная проблема оказывает на детей и подростков. Так, например, по данным Всемирной организации здравоохранения [2], каждый 10-й подросток страдает из-за зависимости от социальных сетей и компьютерных игр. Как следствие – можно заметить ухудшение успеваемости ребёнка в школе и его физического состояния, да и в целом отставание в развитии. Например, по данным всё того же НАФИ, **71%** опрошенных россиян считают, что в среднем лучше учатся школьники, которые не играют в видеоигры. По мнению **29%** опрошенных, школьники, которые регулярно играют в видеоигры, учатся в среднем лучше сверстников, не играющих в видеоигры.

Чтобы противодействовать распространению игромании, я поставил перед собой цель создать такую онлайн-игру, которая имела бы в себе акцент на изучение какого-то материала в разных областях в игровой форме. За прототип было решено взять всем известную игру “Морской бой”. Моя реализация должна предусматривать не только все функции стандартной игры, но и возможность проводить тематические турниры, на которых, например, между ходами у игроков будут появляться вопросы по теме турнира, корректность ответов на которые будет учитываться в рейтинге участников соревнования. Помимо этого, достаточно хорошей идеей является реализация возможности любому пользователю самостоятельно загрузить в мою игру обычный текстовый файл с какими-то заданиями и ответами на эти задания для их использования в процессе игры. В таком случае, моя игра будет полезна не только игроманам, которые хотят отучиться от этой вредной привычки и получить новые знания (возможно, приобрести какие-то новые интересы в научной сфере, посредством использования моей игры), но и учителям в школе, чтобы привить любовь к учёбе школьникам младших классов, например. А если добавить в мою игру интеграцию с такими ресурсами, как Решу ОГЭ/ЕГЭ или с открытым банком заданий ФИПИ, то пользу в данном продукте найдут также средние и старшие школьники, которые могут использовать его в процессе своей подготовки к государственным экзаменам.

# 2. АНАЛИЗ И ВЫБОР СРЕДСТВ ДЛЯ ПРОГРАММНОЙ РЕАЛИЗАЦИИ ПРОДУКТА

## **2.1 Анализ и сравнение доступных технологий**

Перед началом работы следовало определиться с выбором тех программных средств, при помощи которых представлялось возможным создать данный продукт.

Когда речь заходит о создании каких-то видеоигр, то первое, что приходит в голову – это использование языка C++ в связке с фреймворком Unreal Engine или C# с фреймворком Unity. Конечно, у данных подходов есть свои плюсы, такие, как возможность создания достаточно сложной графики и достижение значительно большей производительности по сравнению с другими технологиями.

Помимо реализации графики и игровой логики, мне требовалось реализовать производительный веб-сервер, способный обрабатывать множество пользовательских подключений. В этой области (в backend-разработке) используется огромное количество языков программирования. Самыми производительными из них являются всё те же C/C++, также Golang и другие. Но реализовывать сервер на C/C++ - очень затратная по времени задача.

Так как я не располагал большим количеством времени для реализации данного проекта, было желательно выбрать такие технологии, при помощи которых возможно реализовать всю программную логику игры и сервера в сжатые сроки.

## **2.2 Обобщение полученной информации, выбор наиболее подходящего ПО**

В итоге мой выбор пал на язык программирования Python в связке с фреймворком Pygame (для реализации графики и игровой логики) и библиотекой Websockets (для реализации производительного веб-сервера и всей логики клиент-серверного взаимодействия по одноимённому протоколу связи WebSockets).

**Основные преимущества данного подхода**:

* Простота синтаксиса языка программирования Python и, как следствие, значительная скорость разработки проектов на нём.

Конкретно в данном проекте выше указанные преимущества выбранных технологий являются наиболее весомыми, так как он разрабатывался в весьма сжатые сроки.

* Возможность написания как игровой логики (в совокупности с графикой), так и всей необходимой сетевой логики, используя один и тот же язык программирования.

Данное преимущество сыграло решающую роль в выборе между Python и упомянутыми ранее технологиями (С++ с Unreal Engine или C# с Unity), ведь для реализации клиент-серверной логики на этих языках программирования потребуется значительно больше времени. Конечно, был вариант использовать разные языки программирования для написания графической и сетевой логики, но это потребовало бы больше времени на изучение данных технологий, что неприемлемо в реалиях моего проекта.

* Популярность языка программирования Python.

По данным интернет-ресурса TIOBE [3], Python занимает лидирующие позиции по популярности в последние годы. Высокая популярность данных технологий является немаловажным фактором при выборе, ведь она свидетельствует о наличии большого количества информации по данной теме (множество документаций, руководств, обучающих онлайн-курсов) в открытых источниках.

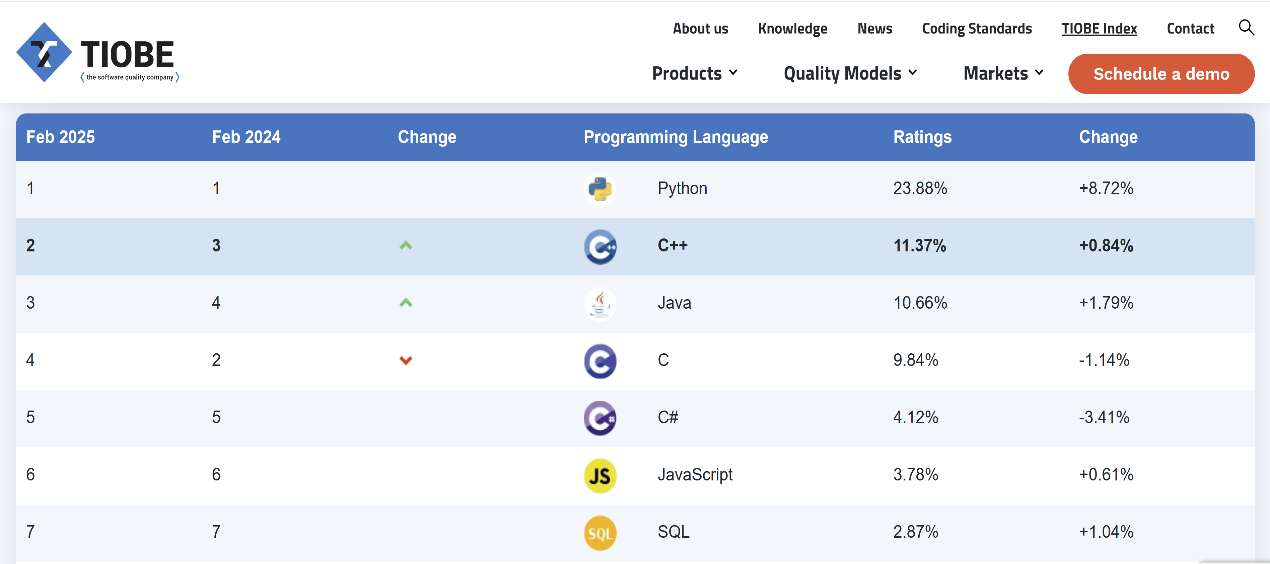


Рисунок 1 – Рейтинг популярности языков программирования

* Кроссплатформенность Python.

Кроссплатформенность используемых технологий означает, что полученный продукт будет возможно запустить практически на любой машине, что также является весомым плюсом использования данного подхода.

* Наличие встроенных в Python библиотек для эффективного использование всей мощности процессора.

В стандартной библиотеке языка Python имеются следующие модули: asyncio, threading, multiprocessing, которые при совместном использовании дают возможность, во-первых, использовать все доступные ядра процессора, а во-вторых, использовать каждое из них максимально эффективно. Это очень мощный инструмент, который может значительно повысить производительность написанной игры и сервера, что оставляет без сомнения верность выбора описываемых технологий.

# 3. ОСОБЕННОСТИ ПРОГРАММНОЙ РЕАЛИЗАЦИИ ДАННОГО ПРОДУКТА

## **3.1 Общая структура проекта**

Всю программную логику своего проекта я реализовал в соответствии с принципами **объектно-ориентированного программирования**, а также структурировал код и другие файлы проекта по соответствующим им директориям:

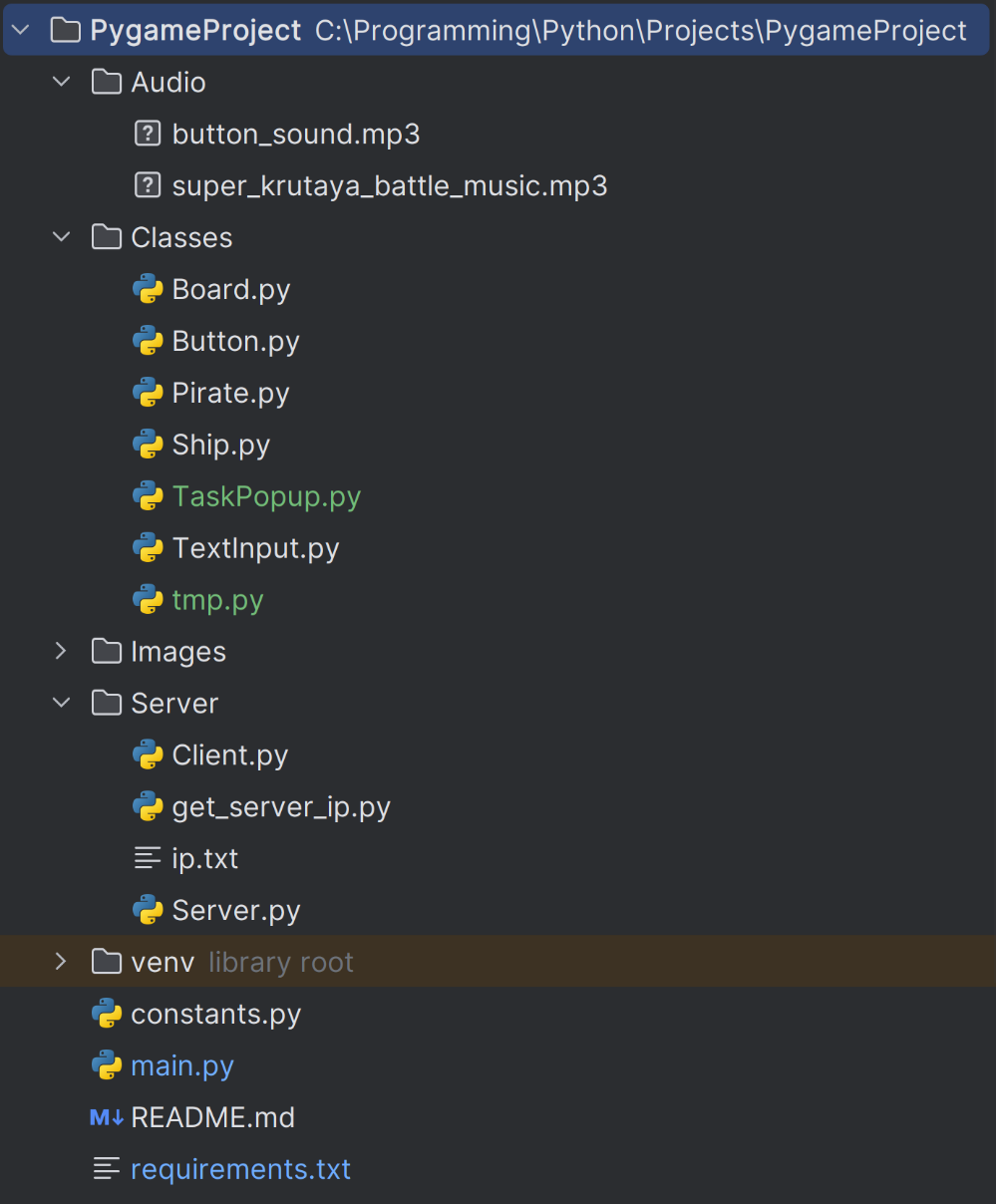


Рисунок 2 – Файлы директории с проектом

**Пояснение к рисунку по структуре проекта**:

* Основная часть игровой логики реализована в главном запускаемом файле корневой директории (main.py), остальная логика игры – в файлах директории /Classes.
* Все графические объекты игры организованы в виде отдельных классов с использованием библиотеки Pygame в директории /Classes.
* Вся серверная логика реализована в файле Server.py директории /Server средствами встроенной в Python библиотеки asyncio (для работы с асинхронным кодом) и сторонней асинхронной библиотеки Websockets.
* Логика сетевого взаимодействия со стороны клиента реализована частично в файле Client.py директории /Server, частично – в основном файле main.py.
* Константы и прочие переиспользуемые компоненты хранятся в файле constants.py корневого каталога проекта.
* Все изображения, используемые в игре, расположены в директории /Images.
* Все элементы звукового сопровождения игры хранятся в директории /Audio.
* Зависимости, необходимые для запуска моего проекта находятся в файле requirements.txt корневого каталога проекта.

## **3.2 Игровая логика, обучающая логика, графика игры**

Логика обработки пользовательских событий реализована в файле main.py. В нём написан класс Game, который содержит в себе основной цикл игры, обрабатывающий нажатие клавиш на клавиатуре, передвижение мыши и другие пользовательские события (средствами библиотеки Pygame).

Помимо этого, файл main.py отвечает за прорисовку всех графических объектов игры. Каждый графический объект представлен в виде отдельного Python-класса. Все они хранятся в каталоге /Classes, о чём было упомянуто ранее. Вот краткая информация по каждому реализованному графическому объекту:

* **Board.**

Данный класс является универсальной реализацией игрового поля для размещения своих кораблей и их прорисовки (как вражеских, так и своих). Он хранит информацию о своём заполнении в виде матрицы (двумерного списка языка программирования Python).

* **Button.**

Представленный класс служит для прорисовки различных кнопок на экране игры. Кнопки могут иметь разные размеры, разный цвет обводки, разный текст внутри себя.

* **Pirate.**

Данный класс представляет из себя реализацию анимации танцующего в такт музыке пирата, привносящего в игру некоторую динамичность.

* **Ship.**

Представленный класс реализует объект корабля для его расстановки на поле (Board). При помощи данного класса можно создавать корабли четырёх типов (однопалубные, двухпалубные, трёхпалубные и четырёхпалубные, аналогично стандартной игре “Морской бой”).

* **TaskPopup.**

В данном файле прописана логика для создания всплывающего окна с текстом вопроса. Если есть необходимость в отображении какой-то математической формулы (представленной в формате LaTeX или MathML), то будет использована сторонняя библиотека mathplotlib, способная справиться с данной задачей. Помимо этого, я рассматриваю возможность автоматической загрузки заданий с таких ресурсов, как Решу ОГЭ/ЕГЭ (при помощи сторонней Python-библиотеки sdamgia) или с Открытого банка заданий ФИПИ (готовой библиотеки для загрузки данных конкретно с этого ресурса я не нашёл, но есть идея воспользоваться средствами библиотеки BeautifulSoap, предназначенной для парсинга html/xml-документов).

* **TextInput.**

Этот класс представляет из себя стандартное поле для пользовательского ввода текстовой информации. Имеет два состояния – активное (готово для ввода после активации курсором мыши) и неактивное (по умолчанию поле имеет серый цвет в неактивном для ввода данных состоянии).

## **3.3 Серверная логика и клиент-серверное взаимодействие**

Как уже было сказано ранее, для написания сервера и всей клиент-серверной логики, использовалась библиотека Websockets. Конкретно на сервере было решено использовать её асинхронную версию в связке со встроенной в Python библиотекой asyncio (для работы с асинхронным кодом). Таким образом, под работающий сервер выделяется один процесс операционной системы, в котором генерируется некоторое количество функций-корутин (зависит от количества подключений), выполняющихся асинхронно (то есть если произвольная из корутин ожидает получения какого-то результата (находится в спящем состоянии), то во время её ожидания выполняются остальные корутины, что позволяет использовать процессорное время крайне эффективно).

При запуске игры (на устройстве пользователя) создаётся 2 процесса операционной системы (средствами библиотеки multiprocessing): в первом из них запускается сама игра, а во втором – подпрограмма для подключения к серверу и взаимодействия с ним. Процессы общаются между собой при помощи специального объекта класса “multiprocessing.Pipe”. Сам же второй процесс разделяется на два потока (средствами встроенной в Python библиотеки threading), которые независимо работают на приём и отправку информации с сервера и на сервер соответственно. Такая модель клиента была выбрана по причине того, что синхронный и асинхронный код не могут эффективно работать вместе. Поэтому на клиенте использовалась синхронная версия библиотеки Websockets.

# 4. РЕЗУЛЬТАТЫ

В ходе работы над данным проектом было изучено и применено на практике колоссальное количество информации. На данный момент готов вполне работоспособный продукт: реализована вся логика стандартной игры “Морской бой” и вся необходимая графика, написан производительный веб-сервер, дающий возможность играть из любой точки мира. Помимо этого, реализована возможность вывода тематических вопросов на экран игры (например, с сайта решу ОГЭ/ЕГЭ), что добавляет игре обучающий характер.

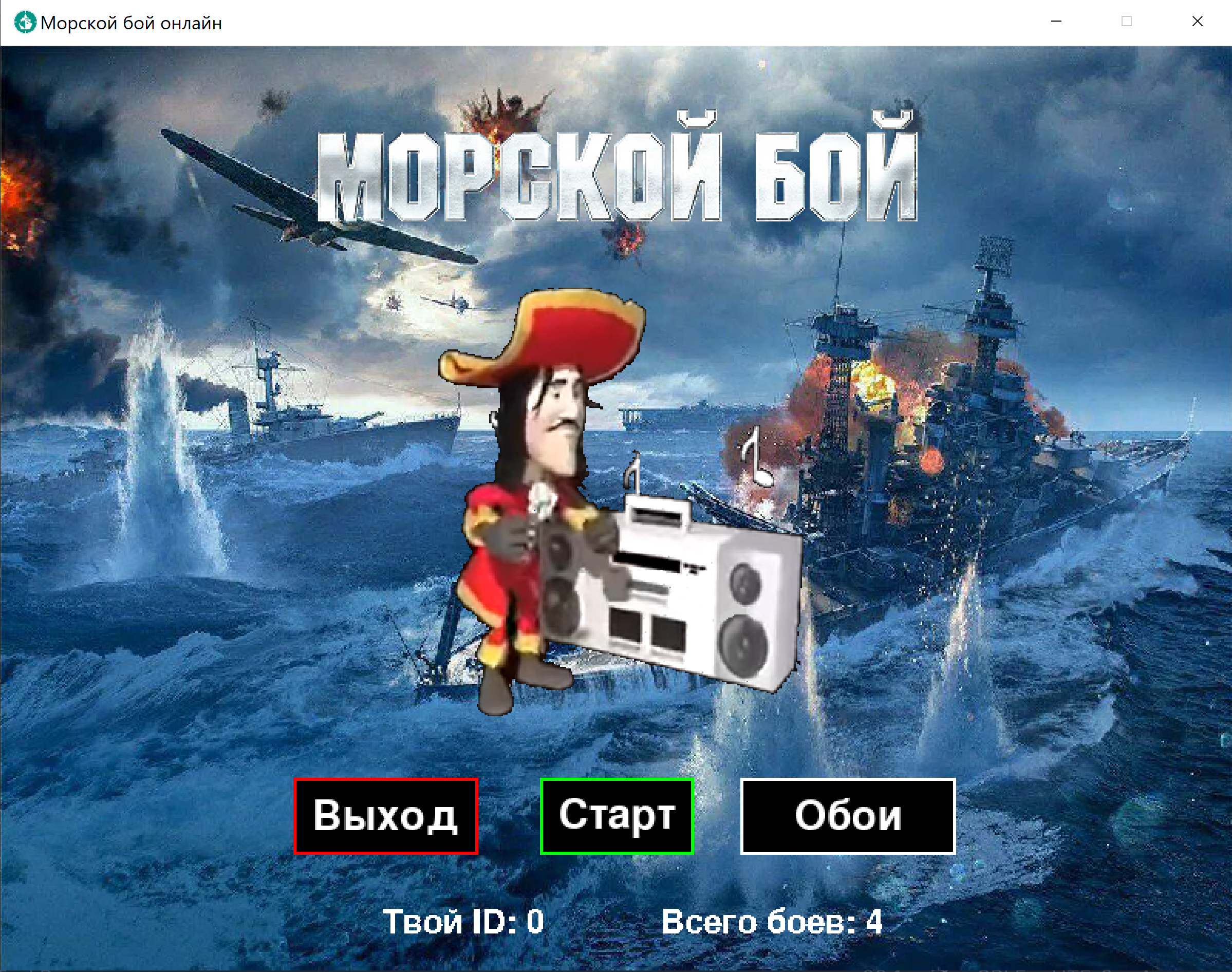


Рисунок 3 – Так выглядит главный экран моей игры

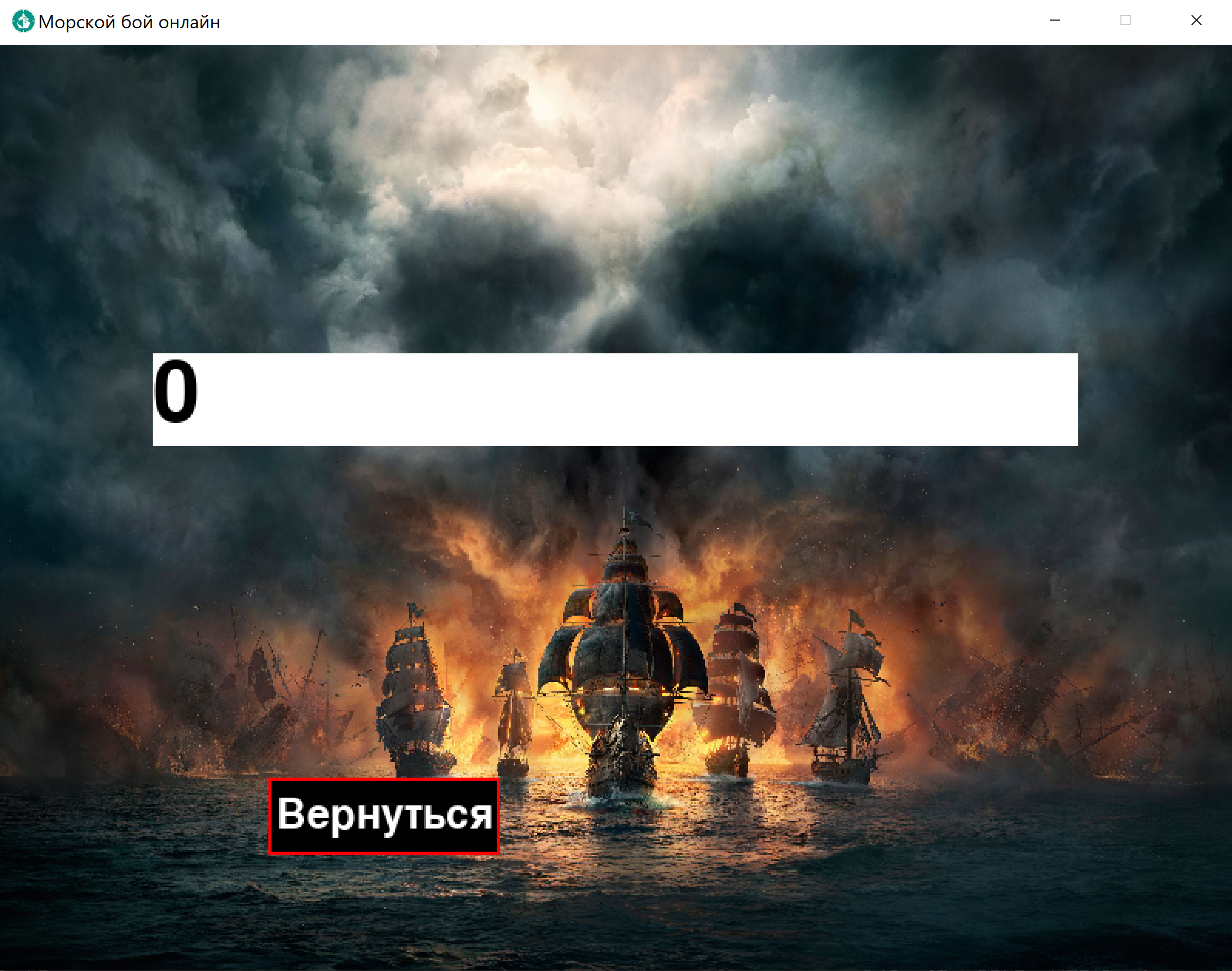


Рисунок 4 – Окно для ввода ID игрока, с кем собираетесь сейчас играть

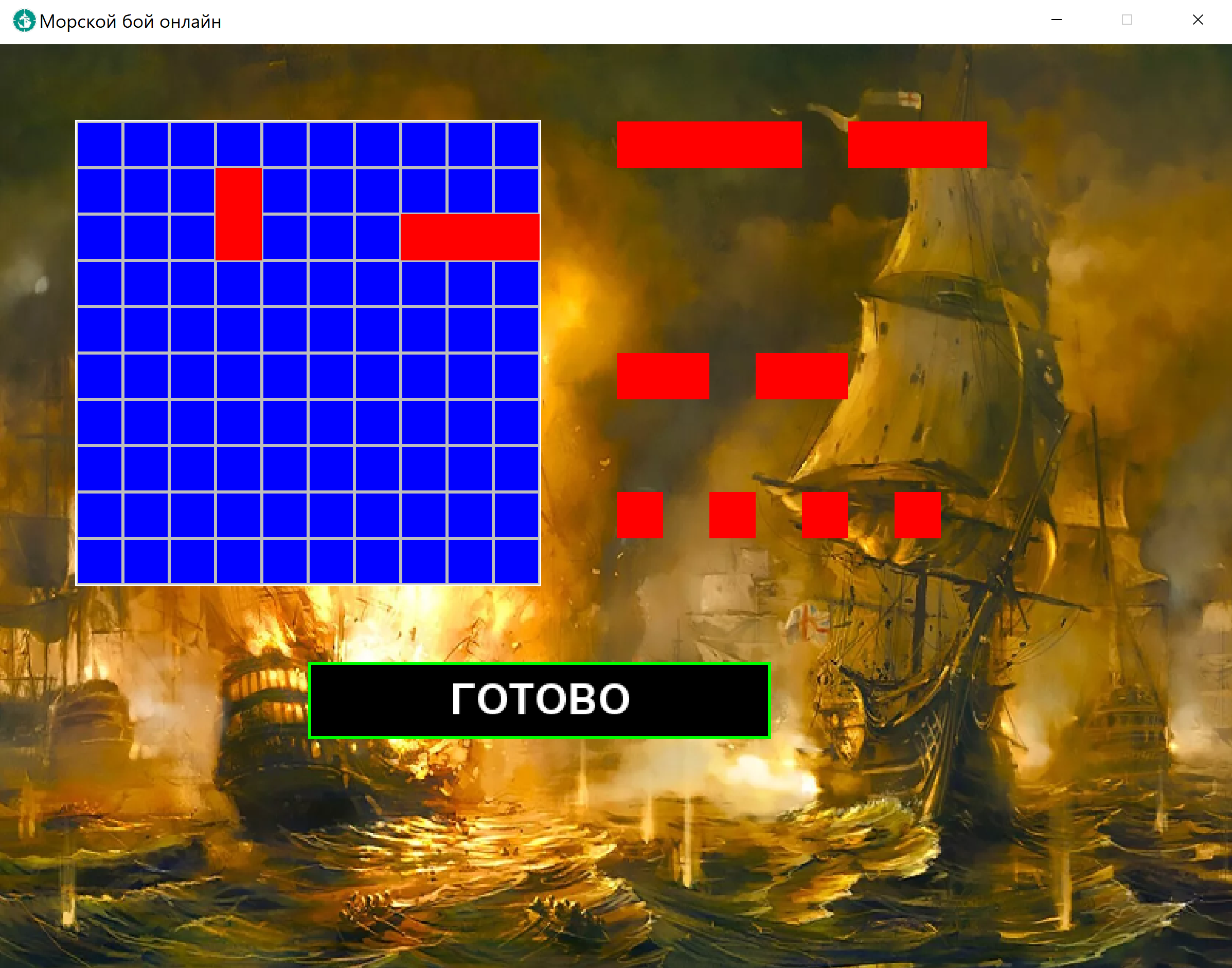


Рисунок 5 – Окно для расстановки кораблей

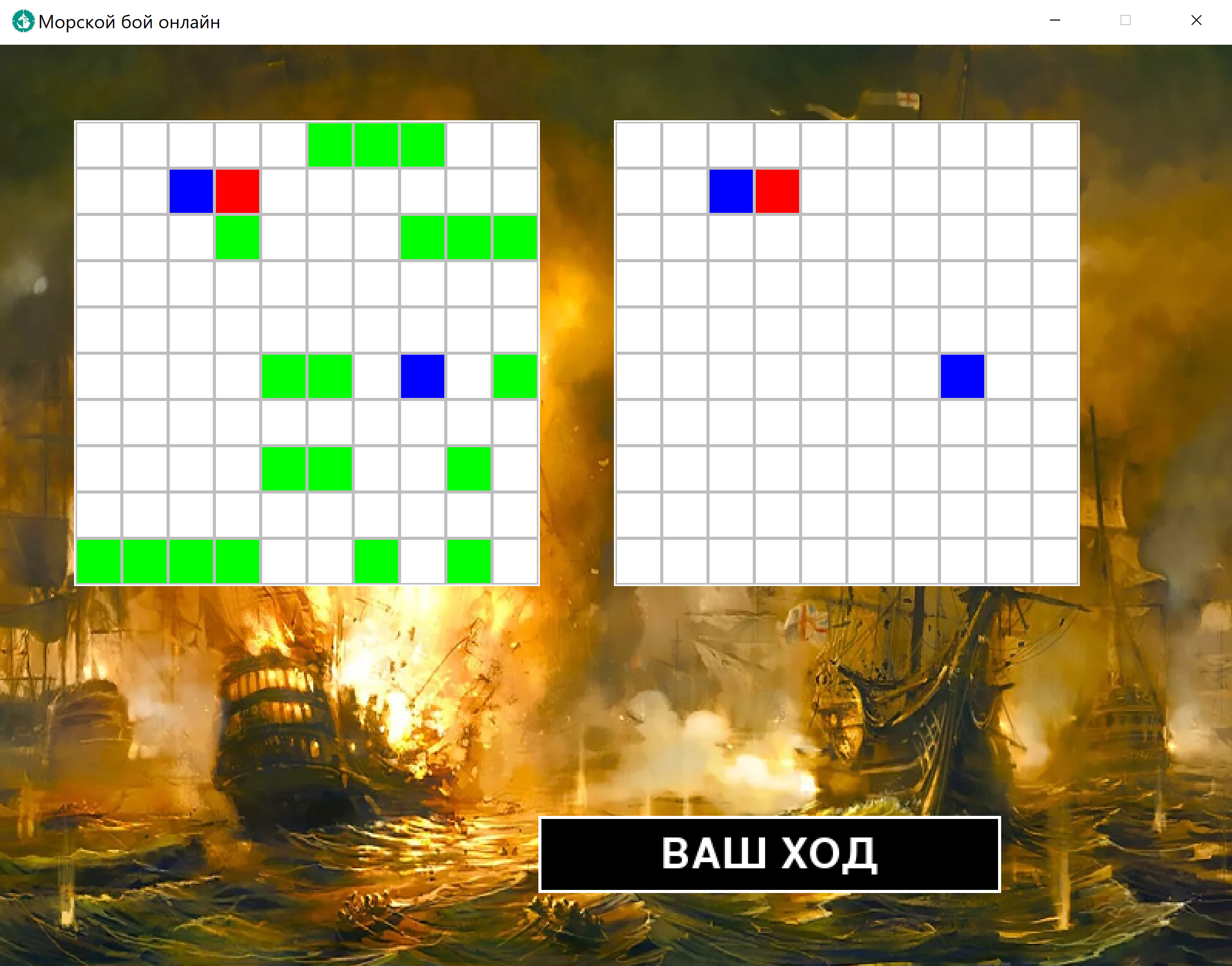


Рисунок 6 – Слева – свои корабли, справа – поле противника

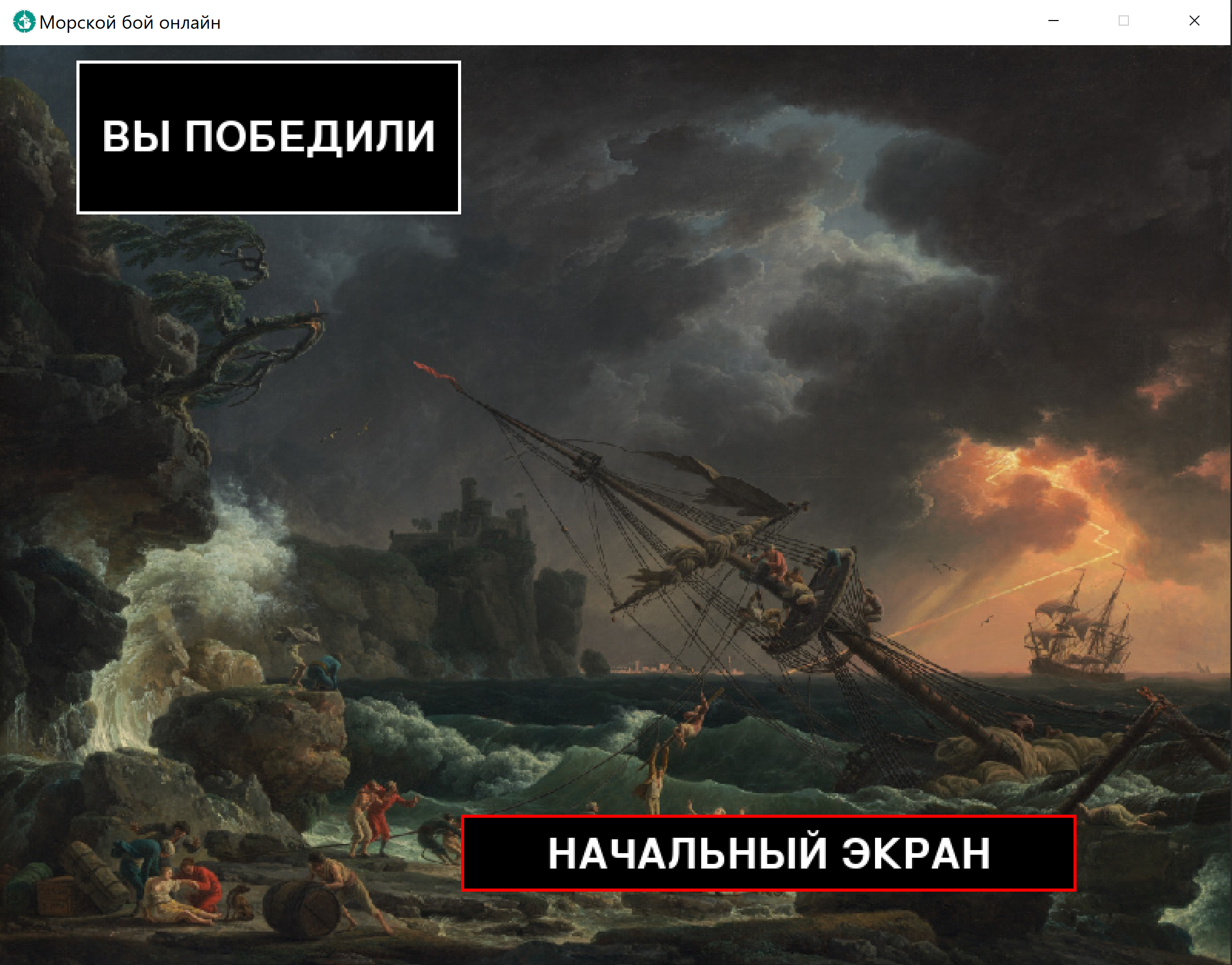
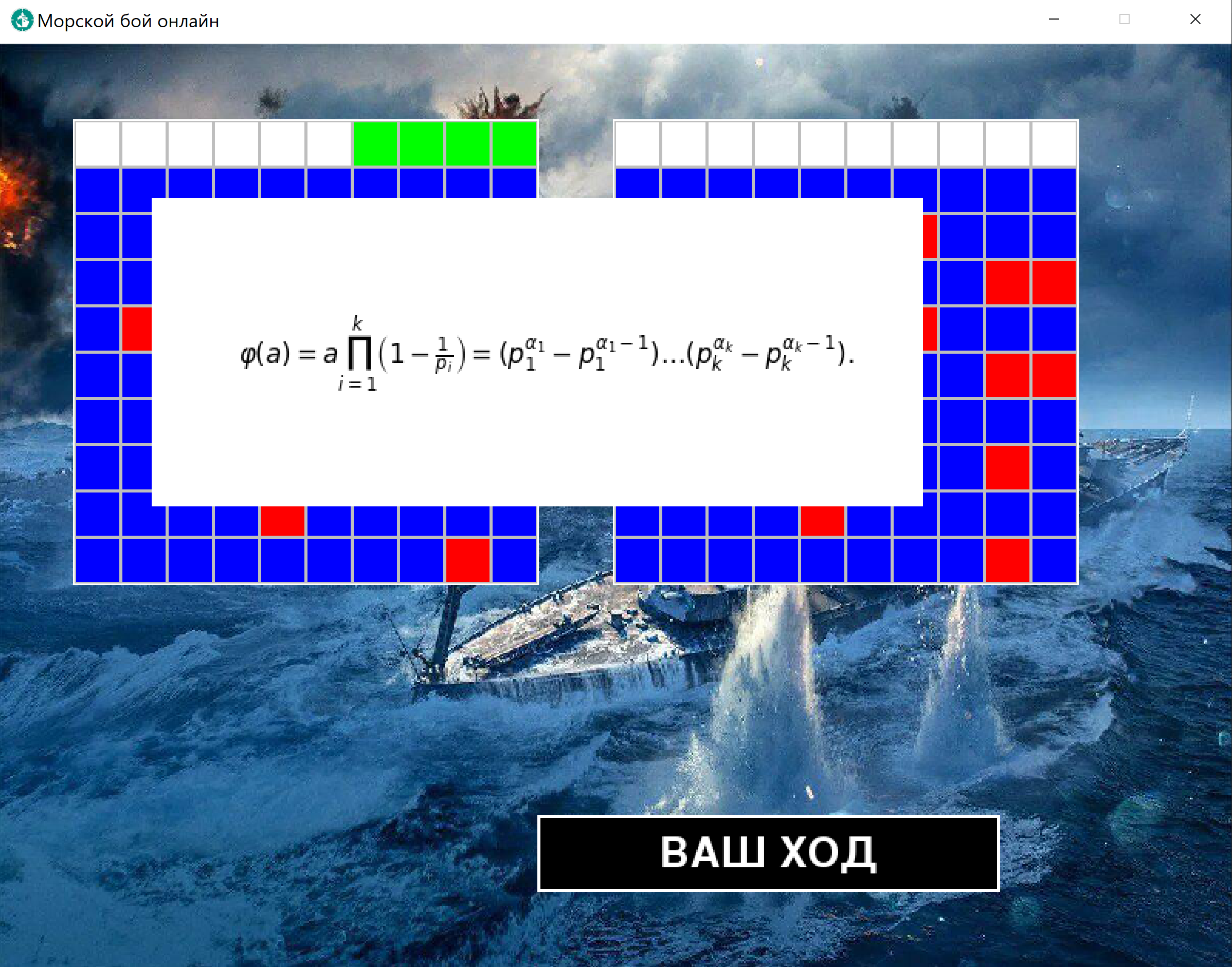


Рисунок 7 – Окно с результатом игры

 Рисунок 8 – В виде такого всплывающего окна у игроков в процессе игры будут появляться какие-то вопросы (здесь в качестве примера просто выводится функция Эйлера (формула была преобразована из MathML-формата))

P.S. Репозиторий с кодом и всеми файлами проекта доступен по ссылке: <https://github.com/Ivan-Bolotov/shag-v-buduschyeye>.

# 5. ЗАКЛЮЧЕНИЕ

Подводя итоги, можно сказать, что на данный момент готово всё, что было запланировано сделать к нынешнему дню. Но это отнюдь не значит, что на этом моя работа над проектом заканчивается. Пока что можно найти достаточное количество аналогичных игр с какой-то обучающей логикой. Но все они не универсальны.

Моя идея дальнейшего развития данного проекта заключается в том, чтобы сделать его доступным и полезным практически для любой аудитории. Конечно, далеко не каждому интересно играть в морской бой. Поэтому я планирую внедрить в свой проект и другие варианты игр (такие, как шахматы, шашки, го и другие). Но в первую очередь, я планирую добавить возможность загрузки своих заданий в эту игру, что может быть полезным учителям младших классов, например. Если преподаватель хочет, чтобы детишки прорешали какие-то задания, и чтобы каждому из них это занятие было интересным, то учителю стоит лишь загрузить файл с заданиями и вариантами ответов в игру и посадить детей играть в неё. Одна из моих учительниц мечтает найти такую игру. Она считает, что это может принести реальную практическую пользу. Поэтому в скором времени я обязательно добавлю эту опцию в свой продукт. Также данную игру можно сделать адаптированной для средних и старших школьников, добавив возможность загрузки заданий с ресурсов для подготовки к государственным экзаменам. Как было сказано ранее, на данный момент у меня уже есть готовый способ загрузки заданий с ресурса Решу ОГЭ/ЕГЭ, и я сейчас в процессе разработки способа получения заданий с ресурса ФИПИ. Стоит отметить, что данное нововведение является не моей спонтанно придуманной идей, а реальным желанием моих одноклассников. Таким образом можно понять, что данной возрастной категории требуется описываемый продукт, что придаёт мне ещё больше уверенности в стремительном росте популярности моего продукта в будущем.

# 6. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ДРУГИХ РЕСУРСОВ

1. Гейминг в России. – URL: <https://www.xn----dtbhaacat8bfloi8h.xn--p1ai/sites/default/files/2022-11/%D0%93%D0%B5%D0%B9%D0%BC%D0%B8%D0%BD%D0%B3%20%D0%B2%20%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8_0.pdf> (дата обращения 10.03.2025).
2. Статистика ВОЗ по игромании в подростковом возрасте (ТАСС). – URL: <https://www.tass.ru/obschestvo/21948057/> (дата обращения 10.03.2025).
3. TIOBE Index. – URL: <https://www.tiobe.com/tiobe-index/> (дата обращения 10.03.2025).