федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вологодский государственный университет»**

Институт математики, естественных и компьютерных наук

Кафедра автоматики и вычислительной техники

Курсовая работа

Дисциплина: «Функциональное и логическое программирование»

Название: «Разработка игры “Морской бой”»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \_\_\_\_09.03.01\_\_.  код направления  подготовки/  специальности | \_\_43.10\_\_.  код выпускающей  кафедры | \_\_\_\_\_\_7\_\_\_\_\_\_  регистрационный номер по журналу | \_\_\_\_\_1\_\_\_\_\_  код формы  обучения | \_\_\_2023\_\_\_  год |

|  |  |
| --- | --- |
| Руководитель | доц. Сорокин А.Н. |
|  |  |
| Выполнил (а) студент | Ломков С.А. |
|  | |
| Группа, курс | 4Б09 ПО-41, 4 курс |
| Дата сдачи | |
| Дата защиты |  |
| Оценка по защите |  |
| *(подпись преподавателя)* | |

Вологда

2023 г.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 2](#_Toc153326313)

[1 АНАЛИЗ ЗАДАЧИ И ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К РАЗРАБАТЫВАЕМОЙ ПРОГРАММЫ 3](#_Toc153326314)

[1.1 Назначение программы и основные задачи 3](#_Toc153326315)

[1.2 Выбор инструментальных средств разработки 3](#_Toc153326316)

[1.2.1 Язык программирования JavaScript 3](#_Toc153326317)

[1.2.2 ReactJS 3](#_Toc153326318)

[1.2.3 Среда разработки Visual Studio Code 4](#_Toc153326319)

[2 ПРОЕКТНАЯ ЧАСТЬ РАЗРАБОТКИ 6](#_Toc153326320)

[2.1 Общий алгоритм 6](#_Toc153326321)

[2.2 Разработка пользовательского интерфейса 6](#_Toc153326322)

[2.3 Разработка функциональных возможностей приложения 7](#_Toc153326323)

[3 РЕАЛИЗАЦИЯ ПРОГРАММЫ НА JAVASCRIPT 8](#_Toc153326324)

[3.1 Реализация веб-страницы с игрой 8](#_Toc153326325)

[3.1.1 Компонент игрового поля 8](#_Toc153326326)

[3.1.2 Компонент истории ходов 10](#_Toc153326327)

[3.1.3 Контроллер 10](#_Toc153326328)

[3.4. Использованные функциональные примитивы 14](#_Toc153326329)

[4 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 17](#_Toc153326330)

[ЗАКЛЮЧЕНИЕ 21](#_Toc153326331)

[СПИСОК ИСТОЧНИКОВ 22](#_Toc153326332)

[ПРИЛОЖЕНИЕ 23](#_Toc153326333)

# ВВЕДЕНИЕ

Морской бой - это классическая игра, которая на протяжении многих лет пользуется популярностью у людей всех возрастов. Эта игра представляет собой простую, но захватывающую игру, в которой игроки пытаются потопить корабли противника, расположенные на игровом поле. Разработка игры "Морской бой" является интересным проектом для программистов и любителей игр, которые хотят создать свою собственную версию этой классической игры. В данной курсовой работе будет рассмотрена разработка игры "Морской бой", включая процесс проектирования, программирования и тестирования. В результате работы будет создана полноценная игра, которая будет иметь все необходимые функции и возможности для того, чтобы пользователи могли наслаждаться игрой и проводить время с удовольствием.

# 1 АНАЛИЗ ЗАДАЧИ И ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К РАЗРАБАТЫВАЕМОЙ ПРОГРАММЫ

## 1.1 Назначение программы и основные задачи

Конкретное назначение разрабатываемой игры:

1. Создание интересной и захватывающей игры.

2. Предоставление возможности проводить свободное время с удовольствием.

3. Ориентация на все возрастные категории и возможность игры как индивидуально, так и в компании.

4. Использование игры в качестве обучающей для развития логического мышления и навыков стратегического планирования у детей.

5. Общая цель - предоставить пользователям увлекательный и интересный способ провести свободное время.

## 1.2 Выбор инструментальных средств разработки

Для игры «Морской бой» были использованы следующие средства разработки:

### 1.2.1 Язык программирования JavaScript

JavaScript - это язык программирования, который используется для создания интерактивных веб-страниц и приложений. Он позволяет добавлять динамический контент на страницы, взаимодействовать с пользователем и управлять поведением веб-страницы. JavaScript может быть использован как на стороне клиента, так и на стороне сервера.

### 1.2.2 ReactJS

ReactJS - это библиотека JavaScript, которая используется для создания пользовательских интерфейсов на веб-страницах. Она позволяет разбивать интерфейс на компоненты и управлять ими с помощью JavaScript. ReactJS использует виртуальный DOM, что позволяет оптимизировать производительность при работе с большим количеством данных. ReactJS также может использоваться как на стороне клиента, так и на стороне сервера.

Достоинства технологии:

* Быстрая отрисовка: ReactJS использует виртуальный DOM, что позволяет снизить нагрузку на браузер и ускорить отрисовку страниц.
* Модульность: ReactJS позволяет разбивать приложение на небольшие компоненты, которые могут быть повторно использованы в других частях приложения.
* Удобство работы с данными: ReactJS обеспечивает удобную работу с данными и обновление интерфейса в режиме реального времени.
* Простота интеграции: ReactJS легко интегрируется с другими библиотеками и фреймворками, такими как Redux, Angular и Vue.
* Поддержка сообщества: ReactJS имеет большое сообщество разработчиков, которые активно обсуждают и делятся опытом использования этой технологии.
* Кросс-платформенность: ReactJS может быть использован для разработки как веб-приложений, так и мобильных приложений для iOS и Android.

### 1.2.3 Среда разработки Visual Studio Code

Visual Studio Code является одним из самых популярных редакторов кода в мире. Его преимущества:

1. Бесплатный и открытый исходный код - это означает, что любой может использовать его бесплатно и даже внести свой вклад в развитие проекта.

2. Поддержка множества языков программирования и платформ - это делает его универсальным редактором для разработки любого проекта.

3. Множество функций - интеграция с Git, отладка кода, автодополнение, подсветка синтаксиса и многое другое, что делает работу с кодом более удобной и эффективной.

4. Расширяемость - возможность создавать свои собственные расширения для улучшения функциональности редактора.

5. Кроссплатформенность - Visual Studio Code работает на Windows, macOS и Linux, что делает его доступным для всех пользователей.

В целом, использование Visual Studio Code помогает повысить производительность и удобство работы с кодом.

# 2 ПРОЕКТНАЯ ЧАСТЬ РАЗРАБОТКИ

## 2.1 Общий алгоритм

Алгоритм создания данной игры ReactJS следующий:

1. Инициализация приложения.
2. Подключение всех необходимых зависимостей.
3. Создание разметки для игры и ее стилистика.
4. Создания механизма генерации кораблей.
5. Создание структуры данных и обработчика удара по полям.
6. Отображение промахов и попаданий на поле.
7. Добавление логов игры.
8. Добавление второго игрока.

## 2.2 Разработка пользовательского интерфейса

На следующем рисунке представлен интерфейс игры:



Рисунок 1 – интерфейс игры

Компоненты интерфейса данной игры выполняют различные роли для обеспечения функциональности и взаимодействия с пользователем. Вот подробное описание компонентов и их роли:

1. Игровое поле: необходимо для взаимодействия пользователей, для анализа текущей игровой ситуации и для выполнения ходов.
2. История ходов: этот элемент играет информационную роль, показывает результаты предыдущих ходов и номер игрока текущего хода.

## 2.3 Разработка функциональных возможностей приложения

Эта имеет следующие функциональные возможности, которые реализуются в процессе разработки:

1. Механизм генерации расстановки кораблей.
2. Отображение и смена текущего игрового поля.
3. Возможность взаимодействия с игровым полем.
4. Ведение истории ходов.

# 3 РЕАЛИЗАЦИЯ ПРОГРАММЫ НА JAVASCRIPT

## 3.1 Реализация веб-страницы с игрой

Давайте подробно опишем каждый элемент игры:

### 3.1.1 Компонент игрового поля

Игровое поле реализуется по средствам следующих компонентов:

Игровое поле состоит из строк, основанной на игровой матрице.



Рисунок 2 – Компонент игрового поля

В свою очередь строки игрового поля состоят из элементов (квадратов).



Рисунок 3 – Компонент строки игрового поля

Далее приведена реализация квадрата игрового поля

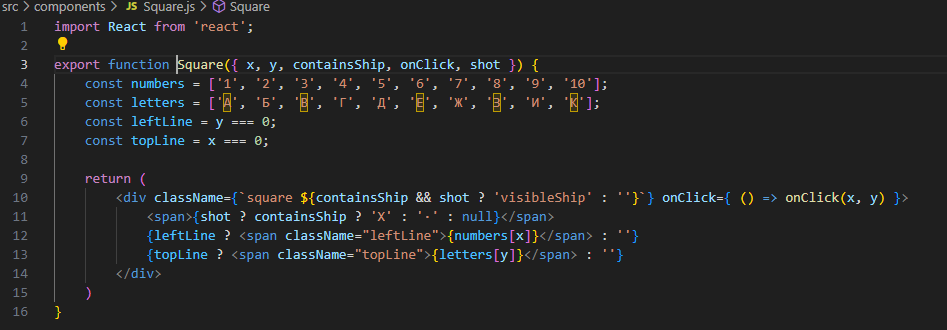


Рисунок 4 – Компонент квадрата игрового поля

Этот компонент отвечает за отрисовку непосредственно самого квадрата в различных состояниях. В изначальном, подбитым, но промазанным (точкой) или подбитым и попавшим в корабль (Х). Также у граничных полей сверху и слева подписываются соответствующие буквы или цифры.

### 3.1.2 Компонент истории ходов

В компоненте истории ходов отображается номер текущего игрока и вся передаваемая в компонент история ходов. Для удобства восприятия история ходов разворачивается.

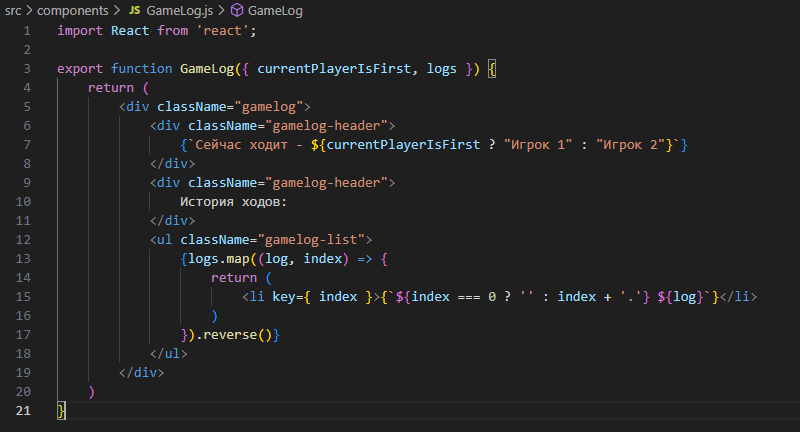


Рисунок 5 – Компонент истории ходов

### 3.1.3 Контроллер

Далее будет рассмотрен элементы с основной логикой игры.

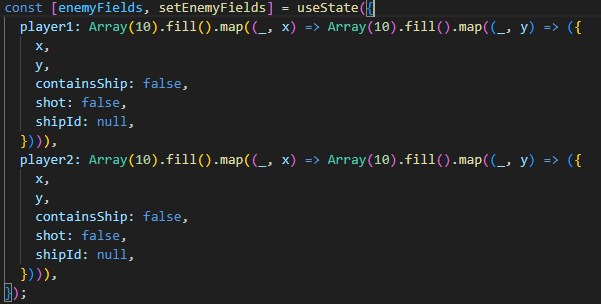


Рисунок 6 – Изначальное (пустое) заполнение полей игроков

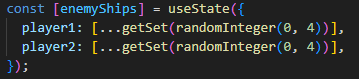


Рисунок 7 – Заполнение расстановки кораблей игроков

Расстановка выбирается случайным образом из некоторых предложенных ниже:



Рисунок 8 – Наборы расстановок

Указываются координаты матрицы с начальной точкой корабля, его длина, направление, размер, уникальный идентификатор и количество очков здоровья у корабля. Далее описаны остальные состояния, включающие в себя флаг окончания игры, массив с историей ходов и состоянием для определения текущего игрока.



Рисунок 9 – Остальные необходимые состояния в контроллере

Также перед началом игры корабли наносятся на игровые поля игроков:

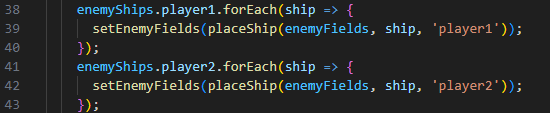


Рисунок 10 – Нанесение кораблей на игровые поля игроков

В зависимости от направления корабля определяем поля, которые он занимает и помечаем их, также указывая идентификатор:

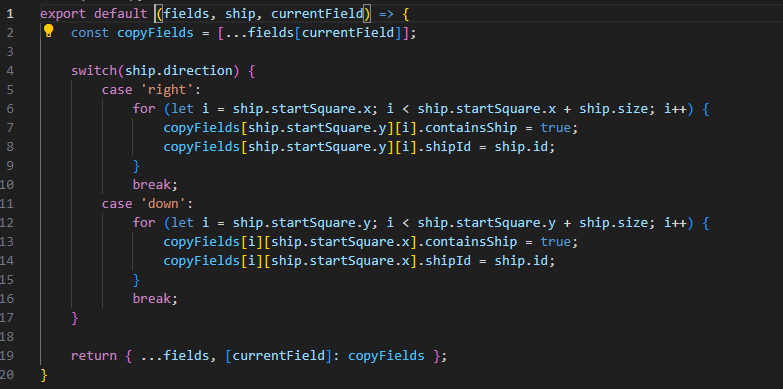


Рисунок 11 – Механизм нанесения кораблей на игровые поля игроков

В завершении описан обработчик нажатия на ячейку игрового поля:



Рисунок 12 – Обработчик нажатия на ячейку игрового поля

Если игра окончена или игрок пытается нажать на уже атакованное поле, тогда прерываем дальнейшие действия. В ином случае помечаем поле как атакованное. Далее необходимо проанализировать игровую ситуацию. Если на атакованном поле нет корабля противника, то записываем в историю ходов «Мимо» и передаем ход следующему игроку. Если произошло попадание в корабль то записываем «Попадание» или «Уничтожил корабль» в соответствии остаются ли очки здоровья корабля или же нет. Если все корабли уничтожены, тогда игра останавливается и объявляется победитель.

## 3.4. Использованные функциональные примитивы

В данной работе использовались следующие функциональные примитивы:

1. Лямбда-выражения: это анонимные блоки, которые обращаются к данным из других фрагментов программы. Они предназначены для реализации функциональных интерфейсов. Работа с ними упрощает программу, делает ее прозрачной и легко читаемой. Пример использования лямбда-выражения:

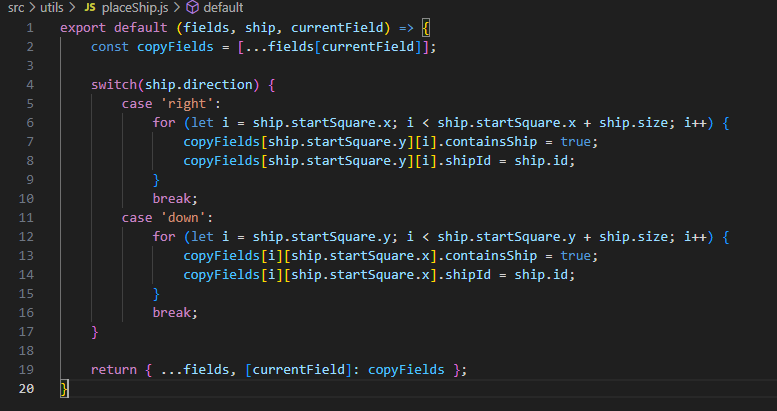


Рисунок 13 – Механизм нанесение кораблей на игровое поле

Данный код демонстрирует механизм нанесение кораблей на игровое поле.

1. Применение .map, .fill, .filter, .reduce… вместо императивных циклов(for, while): следующий код демонстрирует функциональный подход к заполнению игровой матрицы по средствам вызова функций, а не формирования вложенностей императивных циклов

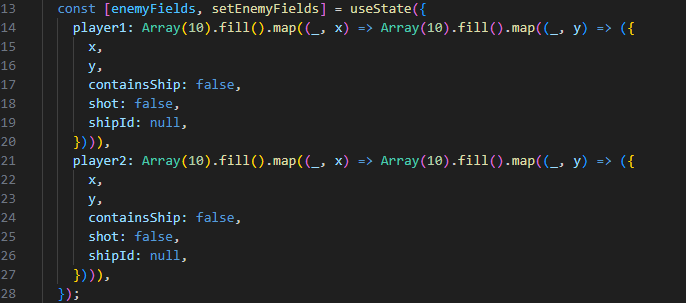


Рисунок 14 – Заполнение матрицы игрового поля

Как плюс можно выделить быстроту и простоту написания, но как минус – неочевидность при незначительном опыте работы с функциональным стилем программирования.

1. Использование чистых функций.

Критериями чистой функции являются детерминированность и отсутствие побочных эффектов. Детерминированностью функции называют требование возвращать одни и те же выходные данные при неизменных входных. Побочным эффектом же называют модификация параметров в процессе вычисления выходного значения. Так, например, в С++ в методы класса неявно передается указатель на экземпляр класса, который метод модифицирует. Но сделав метод константным мы можем избежать побочного эффекта.

Для примера рассмотрим тот же код с механизмом нанесения кораблей на игровое поле:

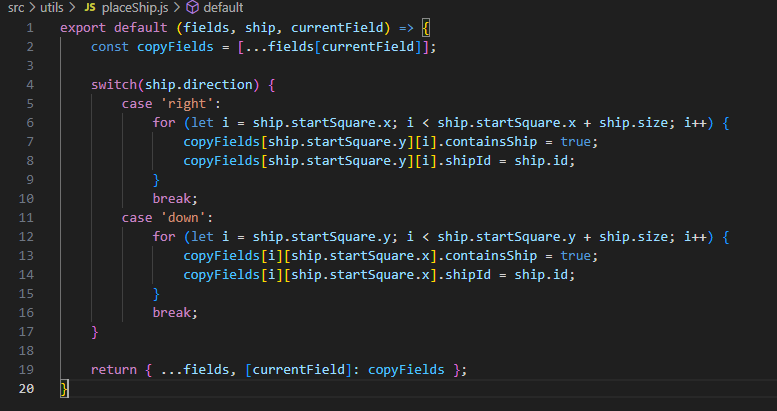


Рисунок 15 – Пример чистой функции в виде механизма нанесение кораблей на игровое поле

Хотя мы и передаем ссылку на объект с данными полей, но при это мы выполняем его копирование, чтобы избежать мутаций параметра. Далее мы работает только с необходимой копией. В качестве выходного значения выдается новый объект поля с нанесенными кораблями. Так как формирование выходного значения зависит только от входных параметров, то значит что эта функция также и детерминированная, что соответствует всем признакам чистой функции. А все необходимые изменения в состояниях будут выполнятся извне.

# 4 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

В данном разделе мы протестируем весь функционал игры.

Изначальный экран:



Рисунок 16 – Первоначальный интерфейс

Отображение промаха на поле:



Рисунок 17 – Промах

Отображение промаха в истории ходов:

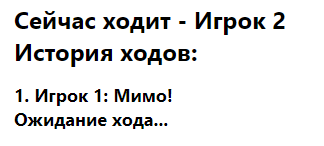


Рисунок 18 – Сообщение о промахе

Отображение попадания:

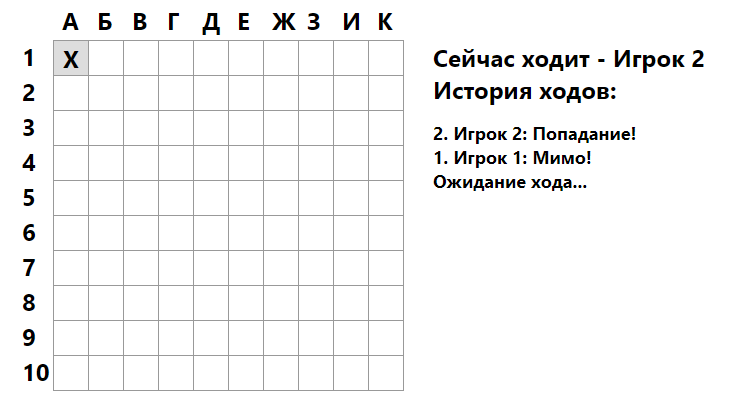


Рисунок 19 – Попадание

Отображение уничтожения корабля:



Рисунок 20 – Уничтожение

Далее игра идет по одному сценарию до победы одного из игроков:



Рисунок 17 – Окончание игры

Таким образом мы видим, что игра обладает необходимым функционалом.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана игра «Морской бой» на языке программирования JavaScript. В процессе работы были изучены основы языка JavaScript, а также принципы работы с ReactJS. Были реализованы функции для создания игрового поля, расстановки кораблей, обработки ходов игроков и определения победителя.

В ходе работы были решены различные задачи, связанные с логикой игры и взаимодействием с пользователем. Были использованы различные технологии и инструменты.

В целом, курсовая работа позволила углубить знания в области программирования на языке JavaScript и получить опыт в разработке игр. Результатом работы является полноценная игра «Морской бой», которая может быть использована как для развлечения, так и для обучения программированию.

.

# СПИСОК ИСТОЧНИКОВ

1. Руководство по JavaScript [Электронный ресурс] // Learn.javascript: сайт. – Режим доступа: <https://learn.javascript.ru/>

2. Руководство по React [Электронный ресурс] // React: сайт. – Режим доступа: <https://react.dev/learn>

3. Руководство по Visual Studio Code [Электронный ресурс] // Code.visualstudio. – Режим доступа: <https://code.visualstudio.com/docs/introvideos/basics>

4. Р. Мартин. Чистый код. Создание, анализ и рефакторинг / пер. с англ. Матвеев Е., Питер, 2020. 464 с.

3. Чистота функции [Электронный ресурс] // wikipedia: сайт. – Режим доступа:<https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D1%82%D0%BE%D1%82%D0%B0_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8>

# ПРИЛОЖЕНИЕ

<https://github.com/stepanLomkov/sea-battle>