Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Современные платформы программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

СЕТЕВОЕ ПРИЛОЖЕНИЕ «МОРСКОЕ СРАЖЕНИЕ»

БГУИР КП 1-40 01 02 105

Студент: гр.351001 Горбачев М. С.

Руководитель: асс. Колотыгин К.Е.

Минск 2016

Учреждение образования

«Белорусский государственный университет информатики и

радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

––––––––––––––––––––––––

(подпись)

––––––––––––––––– 2016 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Горбачеву Максиму Сергеевичу –\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_––––––––––––––––––––––

1. Тема работы   Сетевое приложение “Морское сражение” –––––\_\_\_\_\_\_

2. Срок сдачи студентом законченной работы– 14.05.2016 г.–––   \_\_\_\_\_\_\_\_

3. Исходные данные к работе Среда программирования Visual Studio 2015\_\_\_

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение.

1. Анализ литературных источников.
2. Постановка задачи.
3. Разработка программного средства.
4. Тестирование.
5. Руководство по использованию программного средства.

Заключение. Список использованных источников. Приложения.

5. Перечень графического материала (с точным обозначением

обязательных чертежей и графиков)

1. Схема алгоритма работы программного средства (А1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_––––––––––––––––––––––

6. Консультант по курсовой работе

Колотыгин К.Е. –––––\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_––––––––––––––

7. Дата выдачи задания 20.02.2016 –––––\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_––––––––––––––––––––––   –

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1, введение к 28.02.2016 – 10% готовности работы;

разделы 2 к 15.03.2016г – 30 % готовности работы;

разделы 3,4 к 15.04.2016г – 45 % готовности работы;

раздел 5 к 05.05.2016г – 90 % готовности работы;

оформление пояснительной записки и графического материала

к 10.05.2016г – 100 % готовности работы.

Защита курсового проекта с 10.05.2016г. по 15.05.2016г.

РУКОВОДИТЕЛЬ\_\_\_\_\_\_\_    Колотыгин К.Е.

(подпись)

Задание принял к исполнению А 20.02.2016г.

**СОДЕРЖАНИЕ**

[1 Анализ литературных источников 6](#_Toc438301327)

[1.1 Анализ существующих аналогов 6](#_Toc438301328)

[1.2 Выбор технологии программирования 7](#_Toc438301329)

[2 Постановка задачи 9](#_Toc438301330)

[3 Разработка программного средства 10](#_Toc438301331)

[3.1Представление 10](#_Toc438301332)

[3.2 Классы для работы с представлением 10](#_Toc438301333)

[3.3Проверка размещения корабля на поле 12](#_Toc438301334)

[4 Тестирование 14](#_Toc438301335)

[5 Руководство по использованию программного средства 17](#_Toc438301336)

[Заключение 22](#_Toc438301337)

[Список использованной литературы 23](#_Toc438301338)

[Приложение А 24](#_Toc438301339)

**ВВЕДЕНИЕ**

Одной из самых популярных игр, в которой могут участвовать два человека является **«Морской бой».** Данное приложение является одной из реализаций классической игры  **«Морской бой».**

**«Морское сражение»** — игра для двух участников, в которой игроки по очереди называют координаты на неизвестной им карте соперника. Если у соперника по этим координатам имеется корабль (координаты заняты), то корабль или его часть «топится», а попавший получает право сделать ещё один ход. Цель игрока — первым поразить все корабли противника.

Существуют варианты игры, отличающиеся правилами (распространённые за пределами России). В основном, это касается количества и размера кораблей, например, вариант компании «Милтон Брэдли» — пятиклеточный, четырёхклеточный, два трёхклеточных и двухклеточный. Существуют варианты, где игрок может стрелять больше одного раза подряд.

Из всего многообразия **«Морской бой»** каждый сможет выбрать вариант для себя. **«Морской бой»** - это отличная игра для веселого времяпрепровождения.

# ****1** АНАЛИЗ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ**

## ****Анализ существующих аналогов****

После анализа приложений были выявлены три лидера «Морской бой»

Online SeaBat 1.4.48 (рис. 1.1).



Рисунок 1.1 – Online *SeaBat* 1.4.48

Играть можно как по сети, так и с компьютером. Корабли противника и игрока имеют разный цвет. Само игровое поле представлено в виде знакомого всем расчерченного в клеточку листа. Язык интерфейса: английский.

Минус Online SeaBat 1.4.48 в сильно нагроможденном интерфейсе, и в наличии лишь одного языка (английского).

Вторым приложением стало BattleShip: The Begining (рис. 1.2)



Рисунок 1.2 - BattleShip: The Begining

Можно играть по сети и с компьютером. Игра сопровождается звуками. Поддерживает один язык интерфейса: английский.

Минусом является сложный, порой непонятный интерфейс.

Третье приложение BattleShip2 (рис. 1.3)

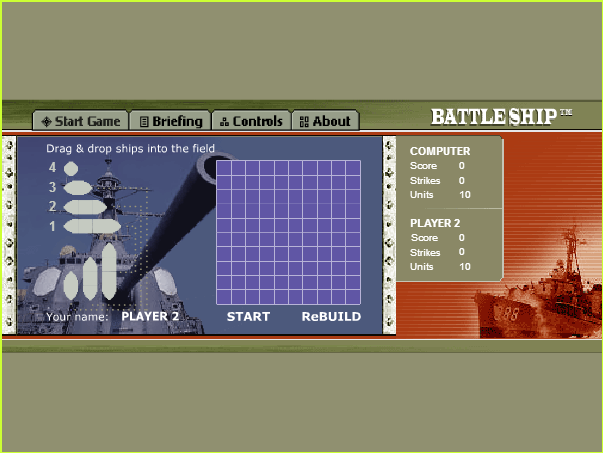


Рисунок 1.3 – BattleShip2

Можно играть через интернет. Достаточно небольшой функционал из-за чего к игре легче привыкнуть.

Минусы: Маленькие надписи с действиями или указаниями и сложность в расстановке кораблей.

## ****Выбор технологии программирования****

**Windows Presentation Foundation (WPF) представляет собой обширный API-интерфейс для создания настольных графических программ имеющих насыщенный дизайн и интерактивность. В отличие от устаревшей технологии Windows Forms, WPF включает новую модель построения пользовательских приложений (в основе WPF лежит мощная инфраструктура, основанная на DirectX).**

Это означает возможность применение развитых графических эффектов, не платя за это производительностью, как это было в Windows Forms. Фактически даже становятся доступными такие графические средства, как поддержка видеофайлов и трехмерное содержимое. Используя эти средства (при наличии хорошего инструмента графического дизайна), можно создавать бросающиеся в глаза пользовательские интерфейсы и визуальный эффекты, которые были просто невозможны в Windows Forms.

Большинство приложений WPF состоит не только из кода, но и из разметки, принятый Microsoft язык XAML (extensible Application Markup Language). В частности, преимуществам разбиения приложения на код и разметку относится возможность распределения процесса разработки между программистами и дизайнерами.

В качестве языка программирования и разметки были выбраны C# и XAML.

XAML — это декларативный язык разметки. С точки зрения модели программирования .NET Framework язык XAML упрощает создание пользовательского интерфейса. Язык XAML напрямую представляет создание экземпляров объектов в конкретном наборе резервных типов, определенных в сборках. В этом заключается его отличие от большинства других языков разметки, которые, как правило, представляют собой интерпретируемые языки без прямой связи с системой резервных типов. Язык XAML обеспечивает рабочий процесс, позволяющий нескольким участникам разрабатывать пользовательский интерфейс и логику приложения, используя потенциально различные средства.

При представлении в виде текста файлы XAML являются XML-файлами, которые обычно имеют расширение .xaml. Файлы можно сохранять в любой кодировке, поддерживаемой XML, но обычно используется кодировка UTF-8.

C# — элегантный, типобезопасный объектно-ориентированный язык, предназначенный для разработки разнообразных безопасных и мощных приложений, выполняемых в среде .NET Framework. С помощью языка C# можно создавать обычные приложения Windows, XML-веб-службы, распределенные компоненты, приложения "клиент-сервер", приложения баз данных и т. д. Visual C# предоставляет развитый редактор кода, конструкторы с удобным пользовательским интерфейсом, встроенный отладчик и множество других средств, упрощающих разработку приложений на базе языка C# и .NET Framework[2].

# ****2**** ПОСТАНОВКА ЗАДАЧИ

Разработать сетевое программное средство «Морское сражение» с использованием технологии построения графического интерфейса WPF и сетевых сокетов.

Приложение должно предоставлять пользователю список оппонентов, находящихся в данный момент в сети. Предоставлять возможность выбора противника, игру с ним по сети и расстановку кораблей пользователя по полю.

# 3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## 3.1Представление

Приложение состоит из одного окна MainWindow, которое отображается при загрузке приложения.

Окно MainWindow описывают два файла:

*-\*.xaml* — разметка страницы на языке XAML;

*- \*.xaml.cs*— это code-behind файл на языке C# страницы.

MainWindow содержит внутри себя элемент TabControl с тремя TabItem: TabMenu, TabEdit, TabGame.

Элемент TabMenu содержит в себе все необходимые формы и кнопки для заполнения и отправки информации на подключение к серверу и выбранному противнику.

ЭлементTabEdit содержит поле для расстановки кораблей, кнопки отвечающие за количество палуб у корабля и кнопку поворота кораблей.

ЭлементTabGame содержит поле игрока с расставленными кораблями и покрытое «туманом войны» поле противника (так как мы не знаем где находятся корабли противника).

## 3.2 Классы для работы с представлением

Модель представлена следующими классами.

Класс *Ship* содержит в себе количество палуб текущего устанавливаемого корабля:

- countOfDecks – хранит количество палуб корабля;

Класс NetworkClient представляет собой класс реализующий паттерн синглтон (так как нам достаточно одного объекта NetworkClient) и предоставляет методы и информацию для подключения к серверу и противнику, а также получения и отправки данных между пользователем и сервером или противником.

- ServerInfo – хранит информацию о сервере

- DNSTable – хранит таблицу данных о подключенных пользователях

- Opponent – хранит информацию о текущем противнике

- User – хранит информацию о пользователе

- UdpConnect() – осуществляет подключение к серверу

-GetUsersTableFromServer() – получает таблицу данных о подключенных пользователях

- SendStepToOpponent() – отправляет ход игрока противнику.

Класс GameFieldCellsHandler содержит в себе методы обработки нажатий на корабли и поле, а также методы установки кораблей с проверкой правильности.

- userBattleField – хранит в себе информацию о расстановке кораблей пользователя

- userBattleShipsRotation – хранит в себе информацию о поворотах кораблей

- opponentBattleField – хранит в себе информацию о расстановке кораблей противника

- opponentBattleShipsRotation – хранит в себе информацию о поворотах кораблей противника

- isWhetherTheUserCanAttackAgain – хранит в себе значение того может ли пользователь походить еще раз

- isShipCanPlace – хранит в себе значение того может ли корабль разместится в этом месте

- CurrentShip – хранит значение о текущем устанавлмиваемом корабле

- ImageMouseDown() – проверяет клетку, на которую нажал пользователь

- ImageMouseEnterAndChecked() – проверяет можно ли установить корабль и отрисовывает результат

- MakeOpponentStep() – отрисовывает ход противника.

Класс UserConnectionData содержит в себе информацию для подключения пользователя к противнику и записи пользователя в таблицу игроков на сервере:

- Name – хранит ник пользователя.

- IP – хранит ip-адресс пользователя.

- Port – хранит порт, на котором пользователь прослушивает входящие подключения в сети.

Класс ServerConnectionData аналогичен классу UserConnectionData, однако не содержит в себе поля Name.

Класс UsersTable содержит в себе таблицу игроков подключенных к серверу:

- Table – хранит коллекцию пользовательских данных.

Класс UsersFieldData содержит в себе расположение кораблей пользователя, а также их ориентацию в пространстве:

- UsersShipsLocation – хранит расположение кораблей на поле

- UsersShipsRotation – хранит повороты кораблей по вертикали или горизонтали.

Класс UserStep содержит в себе номер атакованной клетки и сопровождающую ее информацию:

- TheIndexOfAttackCell – хранит индекс атакуемой клетки

- AdditionalInfo – хранит дополнительную информацию о ходе.

## 3.3Проверка размещения корабля на поле

Проверку возможности размещения корабля и его отрисовкой занимается класс GameFieldCellsHandler. При нажатии на кнопку создания какого-либо корабля, генерируется соответствующее событие, которое создает объект Ship с необходимым количеством кораблей. Затем вызывается метод ImageMouseEnterAndChecked() класса GameFieldCellsHandler и ему передается корабль с необходимым количеством палуб.

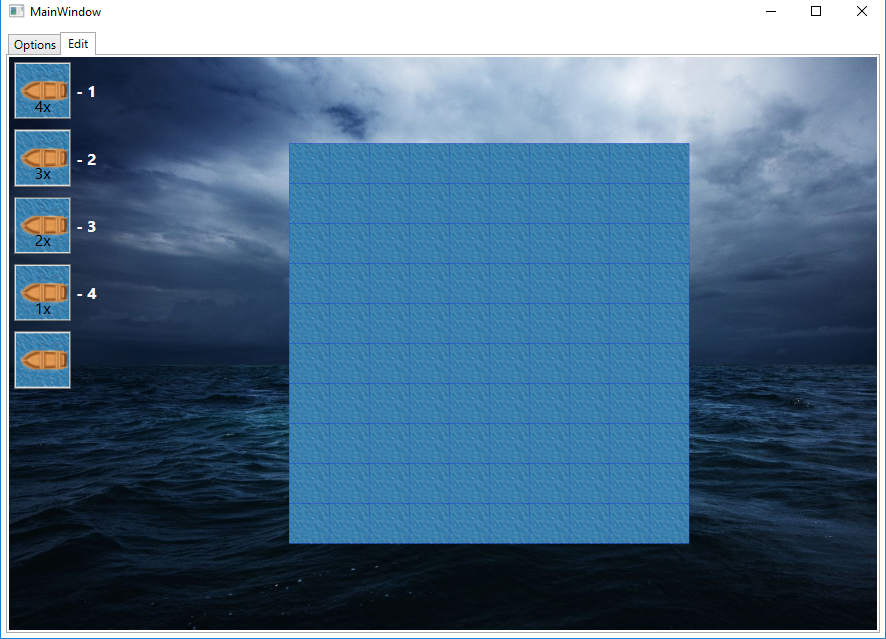


Рисунок 3.1 – Поле редактирования карты кораблей пользователя

Метод ImageMouseEnterAndChecked() проверяет можно ли установить корабль следующим образом:

- IndexOf(UIElement control) – возвращает индекс элемента в коллекции содержимого контейнера.

Затем через полученный индекс находит индексы строки и столбца матрицы расстановки кораблей

В цикле для каждой палубы корабля вызывает следующий метод:

*-* CheckShipPlacement(i, j) – проверяет для одной палубы можно ли ее установить;

Основная задачи класса GameFieldCellsHandler проверить может ли корабль разместиться в том месте, куда указывает курсор пользователя, а также нажатие пользователя по клетке поля противника и отрисовка хода противника.



Рисунок 3.2 – Схема алгоритма CheckShipPlacement

# 4 ТЕСТИРОВАНИЕ

Важным этапом при разработке является этап тестирования. Существует много разновидностей и подходов для тестирования программных средств, однако все они выполняют одну и ту же роль – повышают качество разрабатываемого программного средства. Для тестирования программного средства был разработан набор тест-кейсов. Разработанные тест-кейсы приведены в таблице 4.1.

Таблица 4.1 – Список тест-кейсов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Название тест-кейса | Последовательность действий | Ожидаемый результат | Полученный результат |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Отправка сообщения серверу на подключение | 1. Открыть меню настроек;  2. Ввести ник;  3. Ввести порт;  4. Нажать кнопку подключится к серверу. | Появление списка оппонентов каждые десять секунд | Полное соответствие |
| 2 | Попытка походить первым | 1. Открыть меню настроек;  2. Нажать на поле First;  3. Нажать кноку Start Game;  4. Нажать на поле противника. | Пользователь совершает ход первым, в то время как противник ожидает хода пользователя | Полное соответствие |
| 3 | Проверка возможности установки корабля | 1. Открыть меню редактирования;  2. Выбрать корабль для установки;  3. Навести на поле | Отрисовывание корабля при движении курсора по клеткам поля | Полное соответствие |
| 4 | Установка корабля | 1. Открыть меню редактирования;  2. Выбрать корабль для установки;  3. Навести на поле;  4. Нажать на любую кнопку мыши; | Корабль установится и его нельзя больше двигать по карте | Полное соответствие |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 5 | Поворот кораблей | 1. Открыть меню редактирования;  2. Нажать кнопку повернуть корабль;  3. Выбрать корабль;  4. Навести на поле; | Отрисовывание  Повернутого вертикально корабля при движении курсора по клеткам поля | Полное соответствие |
| 6 | Установка повернутого корабля | 1. Открыть меню редактирования;  2. Нажать кнопку повернуть корабль;  3. Выбрать корабль;  4. Навести на поле;  5. Нажать на любую кнопку мыши; | Перевернутый корабль установится и его нельзя больше двигать по карте | Полное соответствие |
| 7 | Проверка возможности установки корабля рядом с другим кораблем | 1. Открыть меню редактирования;  2. Выбрать корабль;  3. Навести на поле рядом с другим кораблем; | Отрисованный корабль должен быть «подорван» | Полное соответствие |
| 8 | Установки корабля рядом с другим кораблем | 1. Открыть меню редактирования;  2. Выбрать корабль;  3. Навести на поле рядом с другим кораблем;  4. Нажать на любую кнопку мыши; | Отрисованный корабль должен быть «подорван» и он не установится | Полное соответствие |
| 9 | Сделать ход | 1. Открыть меню настроек;  2. Нажать кнопку StartGame  3. Нажать на поле противника | Появление черной точки (промах) или взрывающеегося корабля (попадание) на карте противника | Полное соответствие |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 10 | Принять ход | 1. Открыть меню настроек;  2. Нажать кнопку StartGame  3. Нажать на поле противника  4. Дождатся пока противник нажмет на ваше поле | Появление черной точки (промах) или взрывающеегося корабля (попадание) на вашей карте | Полное соответствие |

# 5 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

Для запуска приложения необходимо, чтобы был запущен сервер (рис. 5.1), и имелось подключение к сети.

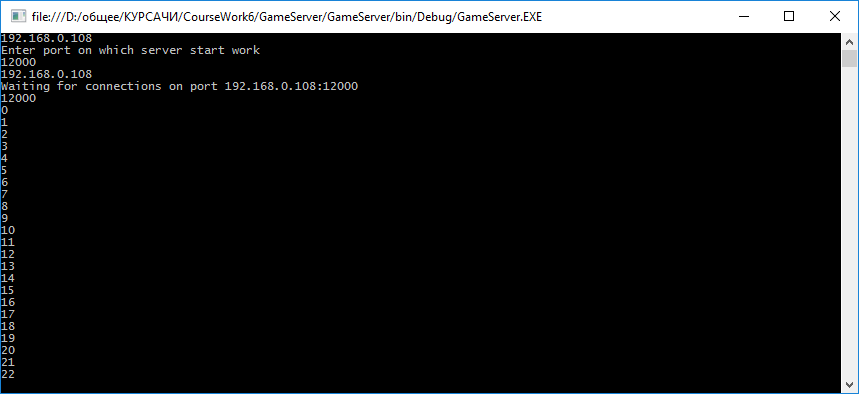


Рисунок 5.1 – Запущенный сервер

Сервер представляет собой отдельное приложение, которое должно быть подключено к сети. Задача сервера – регистрация новых игроков (добавление нового игрока в таблицу игроков) и отправка таблицы пользователей игрокам.

Игровое приложение поддерживает английский язык. После запуска можно увидеть главную страницу приложения, которая содержит меню настроек (рис. 5.2).

Для подключения к серверу необходимо заполнить поля Nick и Port соответственно своим именем в игре и свободным портом.

После этого необходимо нажать кнопку Connect to server (рис. 5.3).

Это приводит к тому, что в поле оппонентов появятся находящиеся в игре пользователи. Список будет автоматически обновляться каждые десять секунд.

Затем откройте панель Edit. На панели вы увидите поле – макет, на котором надо расставить корабли. Чтобы поставить корабль нажмите на кнопку с изображением корабля и подписью количества палуб у этого корабля и затем наведите на поле – макет. Если корабль нужно повернуть вдоль другой оси нажмите на нижнюю левую кнопку. Если корабль будет нарушать правила расстановки кораблей, то он будет нарисован как взорванный и его нельзя будет поставить, пока его не наведут в разрешенную зону.

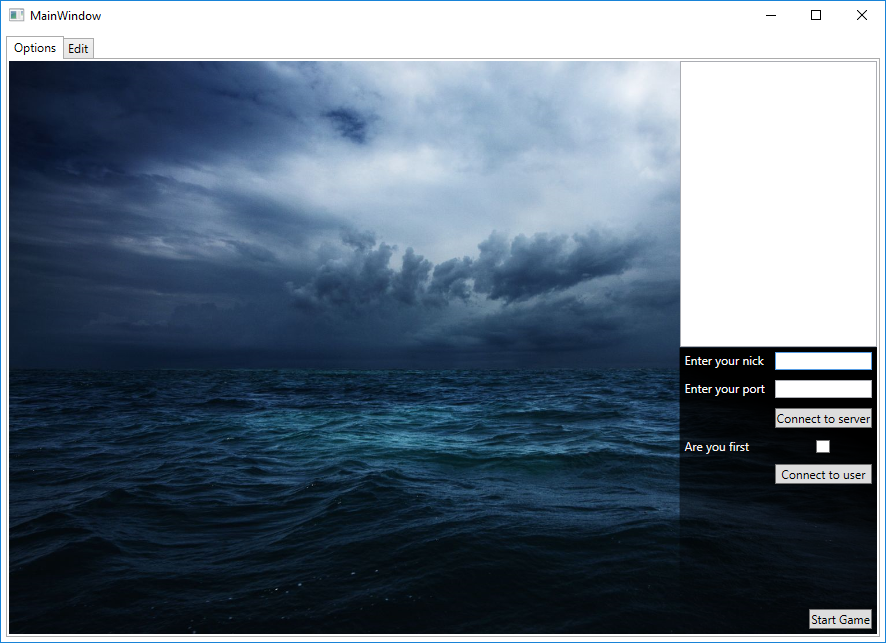


Рисунок 5.2 – Меню настроек соединения



Рисунок 5.3 – Подключение к серверу

Корабли так же ограничены в количестве, которое показано рядом с самим кораблем. Если пользователь захочет поставить кораблей одного типа больше чем их должно быть, то приложение ему этого не позволит.

Затем откройте панель Edit. На панели вы увидите поле – макет, на котором надо расставить корабли. Чтобы поставить корабль нажмите на кнопку с изображением корабля и подписью количества палуб у этого корабля и затем наведите на поле – макет. Если корабль нужно повернуть вдоль другой оси нажмите на нижнюю левую кнопку. Если корабль будет нарушать правила расстановки кораблей, то он будет нарисован как взорванный и его нельзя будет поставить, пока его не наведут в разрешенную зону.

Корабли так же ограничены в количестве, которое показано рядом с самим кораблем. Если пользователь захочет поставить кораблей одного типа больше чем их должно быть, то приложение ему этого не позволит.

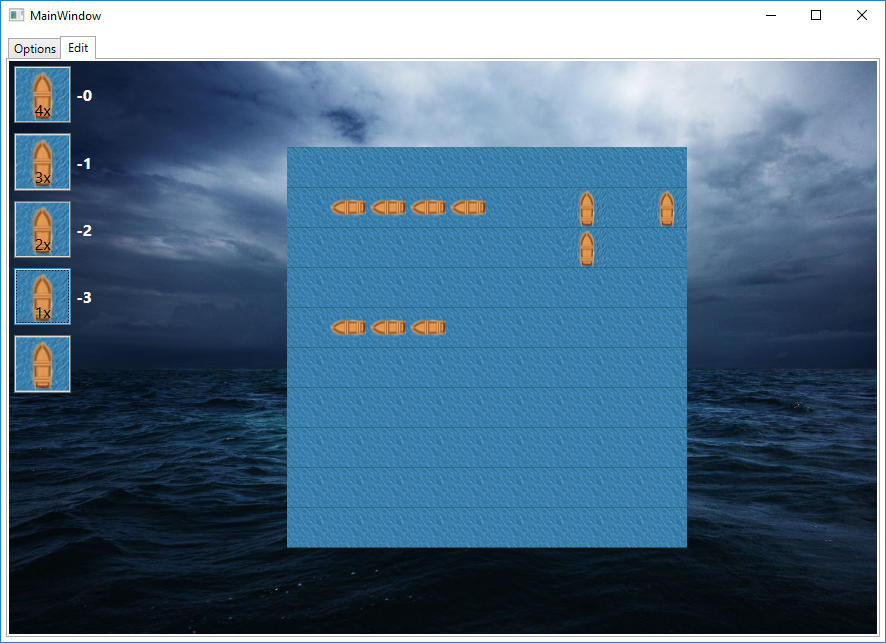


Рисунок 5.4 – Расстановка кораблей

При попытке расставить корабли близко друг к другу будет отображаться картинка подорванного корабля (рис. 5.5).

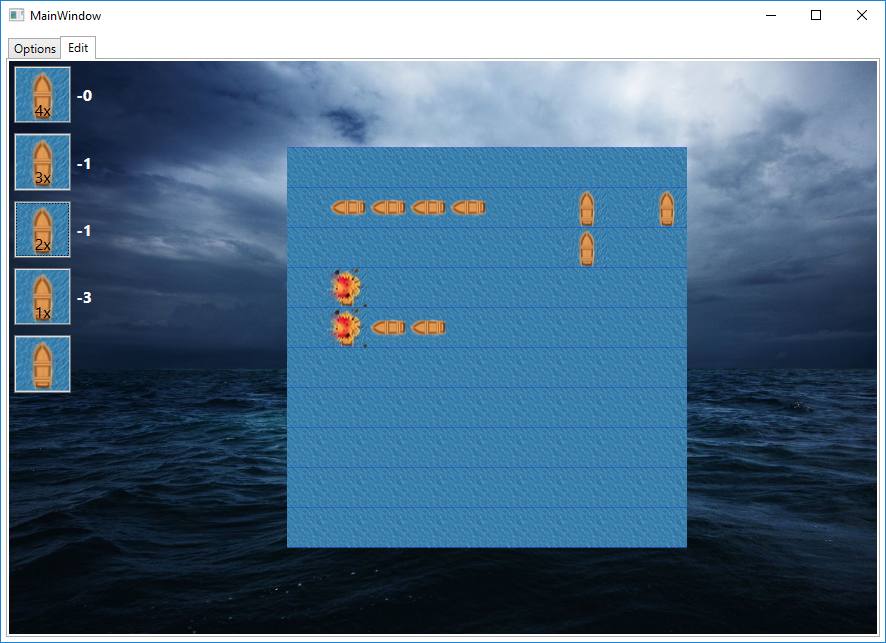


Рисунок 5.5 – Попытка поставить корабли рядом

После того как корабли расставлены необходимо вернутся на вкладку Menu. Дальше выберете противника в списке текущих игроков. Если по договоренности с противником вы должны ходить первыми, то нажмите на поле First. Затем нажмите кнопку Connect to user. После нажмите кнопку Start Game (рисунок 5.6).

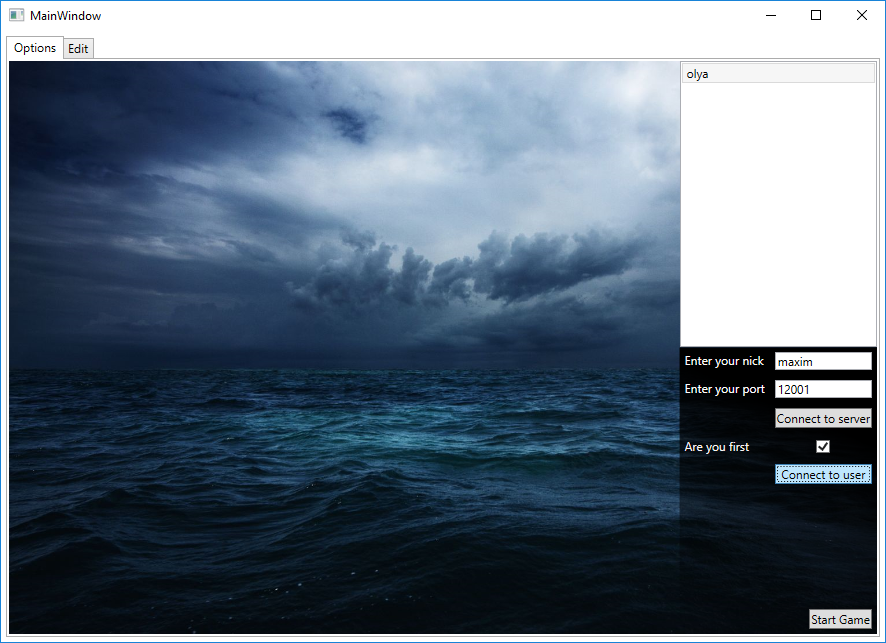


Рисунок 5.6 – Подключение к противнику

Вы попадете на вкладку Game. Для игры нужно просто нажимать на поле противника, пытаясь уничтожить все его корабли раньше, чем он уничтожит ваши (рисунок 5.7).

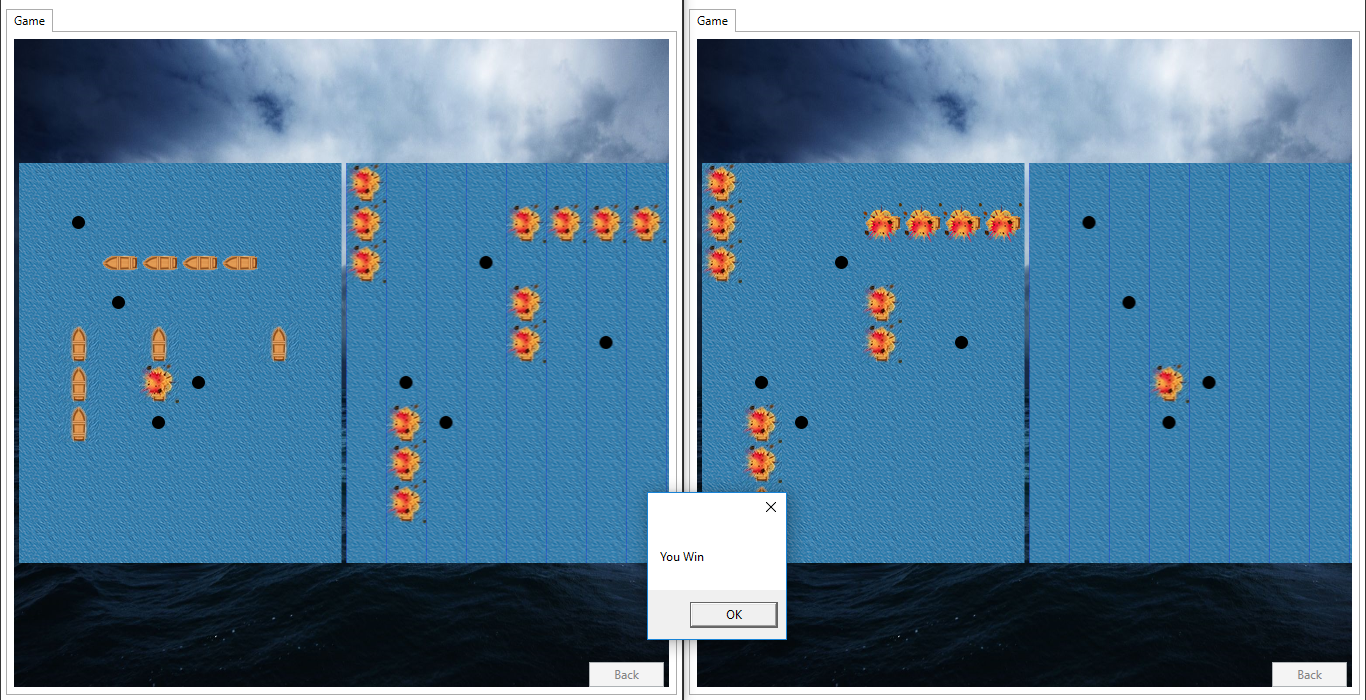


Рисунок 5.7 – Победа в игре

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано программное средство «Морское сражение» для игры по сети с использованием технологии построения графических интерфейсов WPF на языке программирования C#, отвечающее следующим требованиям:

- предоставлять пользователю список оппонентов, находящихся в данный момент в игре;

- предоставлять возможность выбора противника из доступных пользователей;

- расстановку кораблей пользователя по полю;

- предоставляет возможность играть с противником по сети.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Игра «Морской бой» [Электронный ресурс]. – Электронные данные. – Режим доступа: https://en.wikipedia.org/wiki/Battleship\_(game)

[2] WPF (Windows Presentation Foundation) - разработка приложений под Windows [Электронный ресурс]. – Электронные данные. – Режим доступа: http://professorweb.ru/my/WPF/base\_WPF/level1/info\_WPF.php.

[3] Общие сведения о языке XAML [Электронный ресурс]. – Электронные данные. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/ms752059(v=vs.110).aspx.

[4] Введение в язык C# и .NET Framework [Электронный ресурс]. – Электронные данные. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/z1zx9t92.aspx.

[5] Ликнесс Д.. Приложения для Windows на C# и XAML. -2013 - 368с.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Листинг программы**

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Newtonsoft.Json;

using Newtonsoft.Json.Bson;

using System.ServiceModel;

using BSONSerialization;

using UsersDataClasses;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.Collections.ObjectModel;

namespace CourseWork5\_4

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private const int serverUdpPort = 11100;

private bool isHandlersWereSetted = false;

public delegate void UpdateUsersTableContent();

public delegate void CreateMessageboxDelegate(string message);

public delegate void DrawOpponentStepDelegate(UserStep opponentStep);

private bool isGameContinue = false;

private bool isUserCanGo = false;

int countOfFourDecksShips = 1, countOfThreeDecksShips = 2, countOfTwoDecksShips = 3, countOfOneDecksShips = 4;

public MainWindow()

{

InitializeComponent();

InitializeGrid(GameFieldPrototype, true);

GameFieldCellsHandler.CellsHandler.SetGameFone(MainMenuGrid);

GameFieldCellsHandler.CellsHandler.SetGameFone(GameField);

GameFieldCellsHandler.CellsHandler.SetGameFone(MainEditGrid);

}

private void ImageMouseEnter(object sender, MouseEventArgs e)

{

GameFieldCellsHandler.CellsHandler.ImageMouseEnter(sender, e);

}

private void ImageMouseLeave(object sender, MouseEventArgs e)

{

GameFieldCellsHandler.CellsHandler.ImageMouseLeave(sender, e);

}

private void ImageMouseDown(object sender, MouseEventArgs e)

{

if (isUserCanGo)

{

UserStep step;

step = GameFieldCellsHandler.CellsHandler.ImageMouseDown(sender, e);

if (GameFieldCellsHandler.CellsHandler.IsWhetherTheUserCanAttackAgain == 2) // если нажали на корабль

{

SendDataToOpponent(step);

}

else

{

if (GameFieldCellsHandler.CellsHandler.IsWhetherTheUserCanAttackAgain == 0) // если нажали на корабль

{

SendDataToOpponent(step);

isUserCanGo = false;

}

}

}

}

private void ImageMouseEnterAndChecked(object sender, MouseEventArgs e)

{

GameFieldCellsHandler.CellsHandler.ImageMouseEnterAndChecked(sender, e);

}

private void ImageMouseLeaveAndChecked(object sender, MouseEventArgs e)

{

GameFieldCellsHandler.CellsHandler.ImageMouseLeaveAndChecked(sender, e);

}

private void ImageMouseDownAndChecked(object sender, MouseEventArgs e)

{

GameFieldCellsHandler.CellsHandler.ImageMouseDownAndChecked(sender, e);

if (GameFieldCellsHandler.CellsHandler.IsShipCanPlace)

{

ResetEvents();

}

}

private void InitializeGrid(Grid grid, bool isGridBelongsToUser)

{

GameFieldCellsHandler.CellsHandler.InitializeGrid(grid, isGridBelongsToUser);

}

public void GetUsersTable()

{

UpdateUsersTableContent updateDelegate = UpdateOpponentsContent;

while (true)

{

NetworkClient.NetWorkClient.GetUsersTableFromServer();

Dispatcher.BeginInvoke(updateDelegate);

Thread.Sleep(10000);

}

}

private void SendDataToOpponent(UserStep step)

{

NetworkClient.NetWorkClient.SendStepToOpponent(step);

}

private void ButtonConnectionToServer\_Click(object sender, RoutedEventArgs e)

{

UsersConnectionData connectionData = new UsersConnectionData();

connectionData.Port = Convert.ToInt32(UserPort.Text);

connectionData.Name = UserLogin.Text;

NetworkClient client = NetworkClient.NetWorkClient;

client.UdpConnect(connectionData);

//OpponentIP.Text = client.ServerInfo.IP;

isGameContinue = true;

Thread tableGetThread = new Thread(new ThreadStart(GetUsersTable));

tableGetThread.Start();

StartListenerThread();

}

private void ButtonStartGame\_Click(object sender, RoutedEventArgs e)

{

TabGame.Visibility = System.Windows.Visibility.Visible;

TabGame.Focus();

TabEdit.Visibility = System.Windows.Visibility.Collapsed;

TabMenu.Visibility = System.Windows.Visibility.Collapsed;

InitializeGrid(UserField, true);

InitializeGrid(OpponentField, false);

isUserCanGo = (bool)IsUserFirst.IsChecked;

foreach (Image image in OpponentField.Children)

{

image.MouseEnter += ImageMouseEnter;

image.MouseLeave += ImageMouseLeave;

image.MouseDown += ImageMouseDown;

}

}

private void ButtonFourDeck\_Click(object sender, RoutedEventArgs e)

{

if (countOfFourDecksShips > 0)

{

GameFieldCellsHandler.CellsHandler.CurrentShip = new Ship(4);

GameFieldCellsHandler.CellsHandler.IsShipCanPlace = true;

SetEvents();

countOfFourDecksShips--;

LabelLeftFourDeckShips.Content = "-" + countOfFourDecksShips.ToString();

}

}

private void ButtonThreeDeck\_Click(object sender, RoutedEventArgs e)

{

if (countOfThreeDecksShips > 0)

{

GameFieldCellsHandler.CellsHandler.CurrentShip = new Ship(3);

GameFieldCellsHandler.CellsHandler.IsShipCanPlace = true;

SetEvents();

countOfThreeDecksShips--;

LabelLeftThreeDeckShips.Content = "-" + countOfThreeDecksShips.ToString();

}

}

private void ButtonTwoDeck\_Click(object sender, RoutedEventArgs e)

{

if (countOfTwoDecksShips > 0)

{

GameFieldCellsHandler.CellsHandler.CurrentShip = new Ship(2);

GameFieldCellsHandler.CellsHandler.IsShipCanPlace = true;

SetEvents();

countOfTwoDecksShips--;

LabelLeftTwoDeckShips.Content = "-" + countOfTwoDecksShips.ToString();

}

}

private void ButtonOneDeck\_Click(object sender, RoutedEventArgs e)

{

if (countOfOneDecksShips > 0)

{

GameFieldCellsHandler.CellsHandler.CurrentShip = new Ship(1);

GameFieldCellsHandler.CellsHandler.IsShipCanPlace = true;

SetEvents();

countOfOneDecksShips--;

LabelLeftOneDeckShips.Content = "-" + countOfOneDecksShips.ToString();

}

}

private void SetEvents()

{

if (!isHandlersWereSetted)

{

foreach (Image image in GameFieldPrototype.Children)

{

image.MouseEnter += ImageMouseEnterAndChecked;

image.MouseLeave += ImageMouseLeaveAndChecked;

image.MouseDown += ImageMouseDownAndChecked;

}

isHandlersWereSetted = true;

}

}

private void ResetEvents()

{

if (isHandlersWereSetted)

{

foreach (Image image in GameFieldPrototype.Children)

{

image.MouseEnter -= ImageMouseEnterAndChecked;

image.MouseLeave -= ImageMouseLeaveAndChecked;

image.MouseDown -= ImageMouseDownAndChecked;

}

isHandlersWereSetted = false;

}

}

private void ButtonRotation\_Click(object sender, RoutedEventArgs e)

{

List<Button> buttons = new List<Button>();

List<Image> images = new List<Image>();

buttons.Add(ButtonFourDeck);

buttons.Add(ButtonThreeDeck);

buttons.Add(ButtonTwoDeck);

buttons.Add(ButtonOneDeck);

for (int i = 0; i < buttons.Count; i++)

{

Grid grid = (Grid)buttons[i].Content;

foreach(var child in grid.Children)

{

if (child is Image)

{

images.Add((Image)child);

}

}

}

images.Add((Image)ButtonRotation.Content);

GameFieldCellsHandler.CellsHandler.RotateShips(images);

}

public void UpdateOpponentsContent()

{

ObservableCollection<string> opponents = new ObservableCollection<string>();

for (int i = 0; i < NetworkClient.NetWorkClient.DNSTable.Table.Count; i++)

{

if (NetworkClient.NetWorkClient.DNSTable.Table[i].Name != NetworkClient.NetWorkClient.User.Name)

{

opponents.Add(NetworkClient.NetWorkClient.DNSTable.Table[i].Name);

}

}

ListboxOpponents.ItemsSource = opponents;

}

public void StartListenerThread()

{

Thread listenerThread = new Thread(new ThreadStart(ListenerThreadWork));

listenerThread.Start();

}

public void ListenerThreadWork()

{

IPAddress ipAddr = IPAddress.Parse(NetworkClient.NetWorkClient.User.IP);

IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, (NetworkClient.NetWorkClient.User.Port));

TcpListener listener = new TcpListener(ipEndPoint);

bool isUserGetTables = false;

listener.Start();

// Начинаем слушать соединения

while (isGameContinue)

{

TcpClient client = listener.AcceptTcpClient();

NetworkStream tcpStream = client.GetStream();

if (isUserGetTables)

{

UserStep opponentStep = NetworkClient.NetWorkClient.ReceiveOpponentStep(tcpStream);

if (opponentStep.AdditionalInfo == null)

{

isGameContinue = MakeOpponentStep(opponentStep);

isUserCanGo = true;

}

else

{

if (opponentStep.AdditionalInfo == "KILLED")

{

isGameContinue = MakeOpponentStep(opponentStep);

isUserCanGo = false;

}

else

{

CreateMessageboxDelegate opponentWin = CreateMessageboxOpponentWin;

if (opponentStep.AdditionalInfo == "YOU\_LOOSE")

{

MakeOpponentStep(opponentStep);

Dispatcher.BeginInvoke(opponentWin, new object[] { "You loose" });

}

else

{

Dispatcher.BeginInvoke(opponentWin, new object[] { "Opponent leave game" });

}

isGameContinue = false;

}

}

client.Close();

}

else

{

UserFieldData opponentFieldData = NetworkClient.NetWorkClient.ReceiveOpponentTables(tcpStream);

GameFieldCellsHandler.CellsHandler.OpponentBattleField = opponentFieldData.UsersShipsLocation;

GameFieldCellsHandler.CellsHandler.OpponentBattleShipsRotation = opponentFieldData.UsersShipsRotation;

isUserGetTables = true;

}

}

listener.Stop();

isGameContinue = true;

isUserGetTables = false;

}

public bool MakeOpponentStep(UserStep opponentStep)

{

try

{

DrawOpponentStepDelegate drawer = DrawOpponentStep;

Dispatcher.BeginInvoke(drawer, new object[] { opponentStep });

return true;

}

catch(Exception exc)

{

MessageBox.Show(exc.Message);

return false;

}

}

public void CreateMessageboxOpponentWin(string message)

{

MessageBox.Show(message);

}

public void DrawOpponentStep(UserStep opponentStep)

{

GameFieldCellsHandler.CellsHandler.MakeOpponentStep(opponentStep, UserField);

}

private void ButtonChooseOpponent\_Click(object sender, RoutedEventArgs e)

{

NetworkClient.NetWorkClient.ConnectToOpponent((string)ListboxOpponents.SelectedItem);

}

}

}

NetworkClient.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using UsersDataClasses;

using System.Net;

using System.Net.Sockets;

using BSONSerialization;

namespace CourseWork5\_4

{

class NetworkClient

{

static private NetworkClient netWorkClient = new NetworkClient();

static public NetworkClient NetWorkClient { get { return netWorkClient; } }

private BSONSerializer<UserStep> userStepSerializer = new BSONSerializer<UserStep>();

private BSONSerializer<UsersTable> usersTableSerializer = new BSONSerializer<UsersTable>();

private BSONSerializer<UserFieldData> userFieldDataSerializer = new BSONSerializer<UserFieldData>();

private TcpClient user;

private const int serverUdpPort = 11100;

public ServerConnectionData ServerInfo { get; private set; }

public UsersTable DNSTable { get; set; }

public UsersConnectionData Opponent { get; set; }

public UsersConnectionData User { get; set; }

private NetworkClient(){ }

public void UdpConnect(UsersConnectionData userData)

{

byte[] userDataFromServer = new byte[10000];

byte[] newUserDataFromServer;

BSONSerializer<ServerConnectionData> serverConnectionDataSerializer = new BSONSerializer<ServerConnectionData>();

BSONSerializer<UsersConnectionData> userConnectionDataSerializer = new BSONSerializer<UsersConnectionData>();

UdpClient udpClient = new UdpClient(serverUdpPort);

IPEndPoint remoteIpEndPoint = null;

byte[] serverInfo = udpClient.Receive(ref remoteIpEndPoint);

udpClient.Close();

ServerInfo = (ServerConnectionData)serverConnectionDataSerializer.DeserializeMessage(serverInfo);

IPAddress ipAddr = IPAddress.Parse(ServerInfo.IP);

IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, ServerInfo.Port);

TcpClient client = new TcpClient();

client.Connect(ipEndPoint);

NetworkStream stream = client.GetStream();

byte[] data = userConnectionDataSerializer.SerializeMessage(userData);

stream.Write(data, 0, data.Length);

int length = stream.Read(userDataFromServer, 0, userDataFromServer.Length);

newUserDataFromServer = new byte[length];

for (int i = 0; i < length; i++)

{

newUserDataFromServer[i] = userDataFromServer[i];

}

User = (UsersConnectionData)userConnectionDataSerializer.DeserializeMessage(newUserDataFromServer);

client.Close();

}

public void SendStepToOpponent(UserStep step)

{

byte[] data = userStepSerializer.SerializeMessage(step);

IPAddress ipAddr = IPAddress.Parse(Opponent.IP);

IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, Opponent.Port);

user = new TcpClient();

user.Connect(ipEndPoint);

NetworkStream stream = user.GetStream();

stream.Write(data, 0, data.Length);

}

public void GetUsersTableFromServer()

{

IPAddress ipAddr = IPAddress.Parse(ServerInfo.IP);

IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, ServerInfo.Port);

TcpClient client = new TcpClient();

byte[] message = Encoding.UTF8.GetBytes("GET\_USERS\_TABLE");

client.Connect(ipEndPoint);

NetworkStream stream = client.GetStream();

stream.Write(message, 0, message.Length);

DNSTable = ReceiveUsersTableMessage(stream);

client.Close();

}

public void ConnectToOpponent(string opponentName)

{

Opponent = DNSTable.Table.Single(opponent => opponent.Name == opponentName);

IPAddress ipAddr = IPAddress.Parse(Opponent.IP);

IPEndPoint ipEndPoint = new IPEndPoint(ipAddr, Opponent.Port);

user = new TcpClient();

user.Connect(ipEndPoint);

BSONSerializer<UserFieldData> fieldDataSerializer = new BSONSerializer<UserFieldData>();

NetworkStream stream = user.GetStream();

UserFieldData fieldData = new UserFieldData();

fieldData.UsersShipsLocation = GameFieldCellsHandler.CellsHandler.UserBattleField;

fieldData.UsersShipsRotation = GameFieldCellsHandler.CellsHandler.UserBattleShipsRotation;

byte[] info = fieldDataSerializer.SerializeMessage(fieldData);

stream.Write(info, 0, info.Length);

//GameStream = user.GetStream();

}

public void LeaveGame()

{

user.Close();

}

private UsersTable ReceiveUsersTableMessage(NetworkStream stream)

{

byte[] usersTable = ReceiveMessage(stream);

return (UsersTable)usersTableSerializer.DeserializeMessage(usersTable);

}

public UserStep ReceiveOpponentStep(NetworkStream stream)

{

byte[] opponentStep = ReceiveMessage(stream);

return (UserStep)userStepSerializer.DeserializeMessage(opponentStep);

}

public UserFieldData ReceiveOpponentTables(NetworkStream stream)

{

byte[] opponentFieldData = ReceiveMessage(stream);

return (UserFieldData)userFieldDataSerializer.DeserializeMessage(opponentFieldData);

}

public void CleanOpponentData()

{

Opponent = null;

}

private byte[] ReceiveMessage(NetworkStream stream)

{

byte[] fullData = new byte[10000];

int size;

size = stream.Read(fullData, 0, fullData.Length);

byte[] data = new byte[size];

for (int i = 0; i < size; i++)

{

data[i] = fullData[i];

}

return data;

}

}

}

MainWindow.xaml

<Window x:Class="CourseWork5\_4.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="MainWindow" Height="650" Width="900">

<TabControl x:Name="TabControl" Margin="5">

<TabItem x:Name="TabMenu" Header="Options">

<Grid x:Name="MainMenuGrid">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<Grid x:Name="GridOption" Grid.Column="2">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<ScrollViewer Grid.Row="0" VerticalScrollBarVisibility="Visible" HorizontalScrollBarVisibility="Visible">

<ListBox x:Name="ListboxOpponents"/>

</ScrollViewer>

<Grid x:Name="GridMenu" Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<Label Content="Enter your nick" Grid.Row="0" Grid.Column="0"/>

<Label Content="Enter your port" Grid.Row="1" Grid.Column="0"/>

<Label Content="Are you first" Grid.Row="3" Grid.Column="0"/>

<TextBox x:Name="UserLogin" Grid.Row="0" Grid.Column="1" Margin="5"/>

<TextBox x:Name="UserPort" Grid.Row="1" Grid.Column="1" Margin="5"/>

<CheckBox x:Name="IsUserFirst" Grid.Row="3" Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Center"/>

<Button x:Name="ButtonConnectionToServer" Grid.Row="2" Grid.Column="1" Margin="5" Content="Connect to server" Click="ButtonConnectionToServer\_Click"/>

<Button x:Name="ButtonChooseOpponent" Content="Connect to user" Margin="5" Grid.Row="4" Grid.Column="1" Click="ButtonChooseOpponent\_Click"/>

</Grid>

<Button x:Name="ButtonStartGame" Grid.Row="2" Margin="5" Content="Start Game" HorizontalAlignment="Right" VerticalAlignment="Bottom" Click="ButtonStartGame\_Click"/>

</Grid>

</Grid>

</TabItem>

<TabItem x:Name="TabEdit" Header="Edit">

<Grid x:Name="MainEditGrid">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<Button x:Name="ButtonFourDeck" Grid.Row="0" Grid.Column="0" MinWidth="40" MinHeight="40" Margin="5,5,0,5" Click="ButtonFourDeck\_Click">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

</Grid.RowDefinitions>

<Image Source="Resources/ship5.jpg" Grid.Row="0"/>

<Label Grid.Row="0" Content="4x" HorizontalAlignment="Center" VerticalAlignment="Bottom" FontSize="16" MaxHeight="28"/>

</Grid>

</Button>

<Button x:Name="ButtonThreeDeck" Grid.Row="1" Grid.Column="0" MinWidth="40" MinHeight="40" Margin="5,5,0,5" Click="ButtonThreeDeck\_Click">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

</Grid.RowDefinitions>

<Image Source="Resources/ship5.jpg" Grid.Row="0"/>

<Label Content="3x" Grid.Row="0" HorizontalAlignment="Center" VerticalAlignment="Bottom" FontSize="16" MaxHeight="28"/>

</Grid>

</Button>

<Button x:Name="ButtonTwoDeck" Grid.Row="2" Grid.Column="0" MinWidth="40" MinHeight="40" Margin="5,5,0,5" Click="ButtonTwoDeck\_Click">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

</Grid.RowDefinitions>

<Image Source="Resources/ship5.jpg" Grid.Row="0"/>

<Label Content="2x" Grid.Row="0" HorizontalAlignment="Center" VerticalAlignment="Bottom" FontSize="16" MaxHeight="28"/>

</Grid>

</Button>

<Button x:Name="ButtonOneDeck" Grid.Row="3" Grid.Column="0" MinWidth="40" MinHeight="40" Margin="5,5,0,5" Click="ButtonOneDeck\_Click">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

</Grid.RowDefinitions>

<Image Source="Resources/ship5.jpg" Grid.Row="0"/>

<Label Content="1x" Grid.Row="0" HorizontalAlignment="Center" VerticalAlignment="Bottom" FontSize="16" MaxHeight="28"/>

</Grid>

</Button>

<Button x:Name="ButtonRotation" Grid.Row="4" Grid.Column="0" MinWidth="40" MinHeight="40" Margin="5,5,0,5" Click="ButtonRotation\_Click">

<Image Source="Resources/ship5.jpg"/>

</Button>

<Label x:Name="LabelLeftFourDeckShips" Content="- 1" Grid.Row="0" Grid.Column="1" VerticalAlignment="Center" HorizontalAlignment="Center" Foreground="Black" FontSize="16" FontWeight="Bold"/>

<Label x:Name="LabelLeftThreeDeckShips" Content="- 2" Grid.Row="1" Grid.Column="1" VerticalAlignment="Center" HorizontalAlignment="Center" Foreground="Black" FontSize="16" FontWeight="Bold"/>

<Label x:Name="LabelLeftTwoDeckShips" Content="- 3" Grid.Row="2" Grid.Column="1" VerticalAlignment="Center" HorizontalAlignment="Center" Foreground="Black" FontSize="16" FontWeight="Bold"/>

<Label x:Name="LabelLeftOneDeckShips" Content="- 4" Grid.Row="3" Grid.Column="1" VerticalAlignment="Center" HorizontalAlignment="Center" Foreground="Black" FontSize="16" FontWeight="Bold"/>

</Grid>

<Grid Grid.Column="1" Margin="5">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid x:Name="GameFieldPrototype" Grid.Column="1" VerticalAlignment="Center" Background="Blue">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

</Grid>

</Grid>

</Grid>

</TabItem>

<TabItem x:Name="TabGame" Header="Game" Visibility="Collapsed">

<Grid x:Name="GameField" Margin="5">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height="auto"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid x:Name="UserPlace" Grid.Row="1" Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid x:Name="UserField" Grid.Column="1" Margin="5" Background="Blue">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

</Grid>

</Grid>

<Grid x:Name="OpponentPlace" Grid.Row="1" Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid x:Name="OpponentField" Grid.Column="1" Margin="5" Background="Blue">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

</Grid>

</Grid>

</Grid>

</TabItem>

</TabControl>

</Window>