Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Программное игровое средство «Морской бой»

|  |  |  |
| --- | --- | --- |
| Студент |  | Н.А. Дудкин |
| Руководитель |  | А.В. Варфоломеев |

Минск 2020

Учреждение образования

## «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

## УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

(подпись)

Лапицкая Н.В. 2020г.

ЗАДАНИЕ

## по курсовому проектированию

Студенту *Дудкину Никите Александровичу*

## Тема работы Сетевая игра “Морской бой” на платформе .NET Framework

1. Срок сдачи законченной работы *05.06.2020г.*
2. Исходные данные к работе *Язык программирования С#. Создать*

### программное средство для сетевой игры по локальной сети на платформе

### .NET Framework. Предусмотреть расстановку кораблей и возможность

### подключения двух пользователей по локальной сети. Разработать

### пользовательский интерфейс для удобной работы с приложением.

### \_

## Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

### Введение

### Анализ литературных источников

### Разработка программного средства

### Руководство пользователя

### Заключение

### Список использованных источников

### \_Приложение

## Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

### Схема алгоритма в формате А1 метода AreaFillX, AreaFillY, ConnectionCatcher.

1. Консультант по курсовой работе *Варфоломеев А.В.*
2. Дата выдачи задания *10.02.2020г.*

РУКОВОДИТЕЛЬ *Варфоломеев А.В.*

### (подпись)

Задание принял к исполнению *Дудкин Н.А.*

### (дата и подпись студента)

СОДЕРЖАНИЕ

[Введение 5](#_Toc41670990)

[1 Анализ предметной области 7](#_Toc41670991)

[1.1 Анализ литературных источников 7](#_Toc41670992)

[1.2 Обзор аналогов 7](#_Toc41670993)

[1.3 Формирование требований 9](#_Toc41670994)

[2 Разработка программного средства 10](#_Toc41670995)

[2.1 Структура программы 10](#_Toc41670996)

[2.2 Интерфейс программного средства 11](#_Toc41670997)

[2.3 Работа со звуком 13](#_Toc41670998)

[2.4 Работа с сетью 14](#_Toc41670999)

[2.5 Алгоритмы синхронизации 15](#_Toc41671002)

[2.5 Отрисовка игрового мира 20](#_Toc41671005)

[2.6 Игровая логика 24](#_Toc41671008)

[3 Руководство пользователя 27](#_Toc41671009)

[3.1 Правила игры 27](#_Toc41671010)

[3.2 Управление. 27](#_Toc41671011)

[Заключение 28](#_Toc41671012)

[Список использованных источников 29](#_Toc41671013)

[Приложение А Исходный код программы. 30](#_Toc41671014)

# ВВЕДЕНИЕ

21-й век – век информационных и телекоммуникационных технологий. Всё больше отраслей специализируются в компьютерной сфере, так как это наиболее актуальное и современное направление. Новейшие научные достижения безудержно преображают мир вокруг нас. Огромную популярность набирают компьютерные и мобильные приложения, очки виртуальной реальности и прочие развлечения, связанные с информационной деятельностью.

В далеком шестьдесят первом году двадцатого века в Массачусетском Технологическом (знаменитый MIT) на одном из мэйнфреймов впервые родилась компьютерная игра под названием SpaceWar. Это событие осталось практически никому не известным, поскольку компьютеры тех времен были, как бы это повежливее, чуть-чуть дороговаты для того, чтобы на них играть. Но, само собой, это ничуть не мешало Настоящим Программистам, работавшим на них, делать для себя игры и играть в них.

Знаменитая SpaceWar была весьма любопытна с технологической точки зрения: в те времена и на том компьютере еще не было привычных нам растровых графических дисплеев, только текстовые терминалы и, как вершина графических возможностей, векторные графические дисплеи, где электронный луч на каждом кадре не бегает по строкам, а прямо отрисовывает заданные программой контуры объектов.

Суть игры-прародительницы была весьма проста: два небольших кораблика летали по экрану и стреляли друг в друга снарядами. Так что первый в мире жанр – это, безусловно, аркада.

Само название "аркада", между прочим, происходит от архитектурной детали, эдакого крытого коридорчика, где традиционно размещались игральные автоматы, которым предстояло родиться практически точно через десять лет после появления SpaceWar.

Рассмотри популярную игру под названием “Морской бой”. Предположительно, история этой игры началась в начале прошлого века с появлением больших кораблей: броненосцев, линкоров, эсминцев, ведущих в тот период активные боевые действия. У многих людей возникло желание воспроизвести все эти боевые баталии на бумаге в клетку, тем более что она в то время стала доступной. Согласно другой версии, “Морской бой” был изобретён в 1870-х годах бурлаком Петром Кондратьевым, решившим таким образом отвлечь себя и товарищей от тяжкого изнурительного труда. Судя по всему, игра не только понравилась его товарищам, но и получила широкое распространение среди других слоёв населения и успешно дошла до наших дней. В 1930-х годах фирма Starex в США выпустила специально для игры в “Морской бой” разграфленные блокноты. В 1960-е годы компания Милтона Брэдли выпустила новую версию игры с боевыми корабликами и фишками на пластмассовой доске. В 1982 году эта игра появилась в виде пазлов. Вскоре пояилась и электронная версия этой игры, которая приобрела много аналогов и различных версий. Сегодня эта интересная игра доступна для различных игровых платформ.

Такие игры развивают способность к логическому мышлению, улучшают внимание и память, приучают к усидчивости и терпению. Играющий в “Морской бой” учится вырабатывать стратегию поведения и принимать важные решения.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Анализ литературных источников

Одним из главных источников данных в данной работе служили правила морского боя. Чтобы реализовать само программное средство, надо четко понимать правила игры и ее логику. Из-за очень многого количества аналогов необходимо было выбрать определённую версию игры. Была выбрана оригинальная (российская) версия морского боя, которая и была реализована.

Так как программное средство создавалось с помощью языка С#, то была изучена книга С# для профессионалов под авторством Джона Скита. Автор знаменит своим умениемобъяснять сложный материал очень простым и понятным языком. Книга структурированна таким образом, чтобы осведомить все возможности языка хрнологически. Таким образом читатель может проследить развитие языка. Эта книга отлично подойдёт как новичкам, так и тем, кто уже владеет данным языком программирования на довольно высокм уровне, но хочет постичь все его возможности и научиться грамотному кодингу.

Для уточнения некоторой информации касательно языка C# была использована соответствующая литература. [4]. Книга CLR via C# является высококачественным учебником по языку и позволяет пользователю приобрести специфические знания о построении грамотного и производительного кода. Автор уделяет много времени производительности кода, что является важным фактором при построении голосового помощника.

Дополнительно была рассмотрена книга о программировании на языке C# 8.0 и платформе .NET[5]. Книга затрагивает основные технологии языка C#. Источник был очень полезен благодаря своему пособию по Entity Framework.

Немаловажной особенностью данного издания является то, что оно рассматривает последнюю версию языка, что помогает писать код с использованием последних нововведений.

## Обзор аналогов

Оригинальный вариант игры предполагает наличие игрового поля размером (10×10) и четырьмя вариациями кораблей: 4 одноклеточных, 3 двухклеточных, 2 трёхклеточных и 1 четырёхклеточный (рис. 1.1). 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.

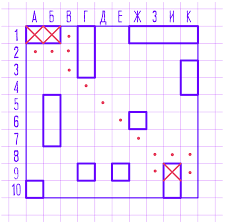


Рисунок 1.1

При стандартном размере поля (10×10) и стандартном наборе кораблей (1×4 + 2×3 + 3×2 + 4×1), в игру можно добавить одну мину (или не одну). Мина обозначается кружком, вписанным в одну клетку. Клетка с миной не должна касаться кораблей, а если мин больше одной, то и других клеток с минами (рис. 1.2).

Если игрок в результате своего хода попал на мину (на мину противника), то он должен сообщить хозяину мины (противнику) координаты одной своей непоражённой клетки, занятой любым своим кораблём (корабль может иметь сколько угодно клеток, но выдаётся только одна клетка). После этого хозяин мины имеет возможность метко выстрелить (выданная клетка не погибает в момент попадания на мину — чтоб она погибла, по ней надо выстрелить; иначе говоря, мина только сообщает координаты корабля). Хозяин мины не обязан поражать выданную клетку сразу же — он имеет право выстрелить по ней в любое время. Поскольку выстрел по выданной клетке меткий, то хозяин мины после этого выстрела получает право на повторный ход. Использованная мина «гасится» постановкой точки в центре кружка (в центре её клетки).

Размер поля можно увеличить — например, размер 16×16 или 18×18 позволяет с удобством использовать весь размер одинарного тетрадного листа. В этом случае количество фигур можно увеличить. Тогда, в связи с увеличением численности армий и размера поля, можно увеличить количество мин (например, до трёх) и добавить в игру минный тральщик (скажем, один у каждого игрока). Минный тральщик обозначается равнобедренным треугольничком, вписанным в одну клетку, так, что основание равнобедренного треугольника совпадает с нижней стороной клетки, а противоположная основанию вершина лежит на верхней стороне клетки, деля верхнюю сторону пополам.



Рисунок 1.2

Если игрок, сделав ход, попал на минный тральщик, то он должен выдать противнику (хозяину минного тральщика) координаты одной из своих ещё не сработавших мин — чтобы хозяин минного тральщика знал, что по этим координатам выданной клетки с миной ходить не следует. Клетка с минным тральщиком не должна касаться клеток с кораблями и минами, а также, если минных тральщиков больше одного, и клеток с другими минными тральщиками. Если к моменту срабатывания минного тральщика у походившего не осталось ни одной мины, то противник походившего сообщает походившему, что он попал на минный тральщик, но походивший ему ничего не выдаёт.

Так как попадание на мину или на минный тральщик не является успехом, а является неприятностью для ходившего, то после такого неудачного хода ход переходит к хозяину сработавшей мины или сработавшего минного тральщика. Попав на мину, нельзя вместо координат клетки корабля выдавать клетку с минным тральщиком. Мины и минные тральщики являются одноклеточными фигурами. Мины и минные тральщики не считаются значащими фигурами — поэтому, если у игрока остались только мины и минные тральщики, но погибли все корабли, а у другого игрока не все корабли погибли, то игра считается оконченной, а первый игрок — проигравшим.

## Формирование требований

При формировании требований к программному средству основное внимание следует уделить удобному и понятному интерфейсу, а также качественной проработки логики программы.

Таким образом, в ходе разработки данного программного средства планируется решить следующие задачи:

* запуск приложения;
* отрисовка карты для игры;
* расстановка своих кораблей на поле боя;
* соединение двух игроков по сети;
* отрисовка своей карты и карты противника(с твоими выстрелами по его кораблям);
* отключение игрока во время игры;
* визуализация при попадании в корабль противника;
* отзывчивое управление.

Для разработки программного средства будет использоваться язык программирования C# и среда разработки Visual Studio 2019.

# 2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## 2.1 Структура программы

В данном приложении использовались следующие классы и методы:

* InfoForm–класс формы, которая появляется при нажатии кнопки “About”;
* InfoForm–метод, с помощью которого инициализируются компоненты на форме InfoForms;
* SyncInForm– класс, который отвечает за отрисовку начальной формы с меню;
* SyncInForm–метод, отвечающий за инициализацию компанентов на начальной форме;
* QuitBtn\_Ckick–обработчик нажатия кнопки, которая отвечает за выход из игры и закрытие начальной формы;
* FindBtn\_Click –обработчик нажатия кнопки, которая отвечает за вызов основной формы игры;
* AboutBtn\_Click–обработчик нажатия кнопки, которая вызывает форму с информацией об игре и создателе данного программного средства;
* GameForm–основной класс программного средства, который отвечает за логику самой игры и за работу с сетью;
* GameForm–метод, который инициализирует на игровой форме все компоненты;
* FieldGenerate–метод, который отвечает за заливку и отрисовку полей сраения;
* ExitBtn\_Click–обработчик нажатия кнопки “Surrender”, при нажатии на которую происходит выход и игры и разрыв соединения;
* GameForm\_FormClosed–обработчик, кторый отвечает за закрытие формы приложения;
* YourFleet\_MouseClick–обработчик нажатия клавишей мыши на область своего поля кораблей (служит для отрисовки своих кораблей, необходимо кликнуть в начальную точку своего корабля);
* EnemyFleet\_MouseClick–обработчик нажатия клавишей мыши на область поля противника (для бомбардировки его кораблей);
* YourFleet\_MouseDoubleClick –обработчик двойного клика мышью по области своего поля (служит для отрисовки своих кораблей, необходимо кликать на конечную точку своего корабля либо на точку расположения однопалубника);
* AreaFillX–метод, который служит для заливки области;
* AreaFillY–метод, который также служит для заливки области;
* ShipLocate–метод, который служит для отрисовки самих кораблей на поле боя;
* ReadyBtn\_Click–обработчик нажатия кнопки “Ready”, при нажатии на которую игрок подтверждает, что рассредоточил свой флот и готов к игре с противником(происходит установка соединения с другим игроком и начало боя);
* ListenerUDP–метод, который производит отправку широковещательных пакетов;
* ConnectionCatcher–метод, который получает сообщение о подключении и запускает асинхронную задачу с методом ListenerTCP;
* ListenerTCP–метод, который получает сообщения от оппонента и, в зависмости от типа сообщения, их определённым образом обрабатывает.

## 

## 2.2 Описание функциональности программного средства

Внешний вид и удобность в использовании являются одними из главных критериев качества программного средства. Поэтому взаимодействие приложения с пользователем необходимо организовать максимально интуитивно и просто.

При запуске приложения показывается главное меню, состоящее из пунктов “About ”, “Quit” и “Find an opponent ”(рис. 2.1).

При выбора пункта “About ” откроется окно с краткой информацией об игре. Здесь будут краткие описание правил игры, а также информация об расстановке кораблей на карте.Также здесь указан автор данного программного средства(рис.2.2).

При выборе пункта “Quit” пользователь выходит из программного средства.

Для того, чтобы начать игру, необходимо выбрать пункт “Find an opponent”. При выборе данного пункта нам открывается наше собственное основное поле игры (рис. 2.3).

Здесь мы видим нашу собственную карту, карту соперника, а также пункты “Ready” и “Surrender”. В верхнем левом углу есть меню для выбора типа корабля.

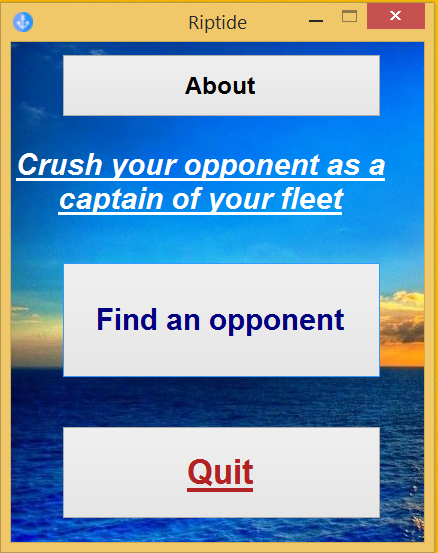


Рисунок 2.1

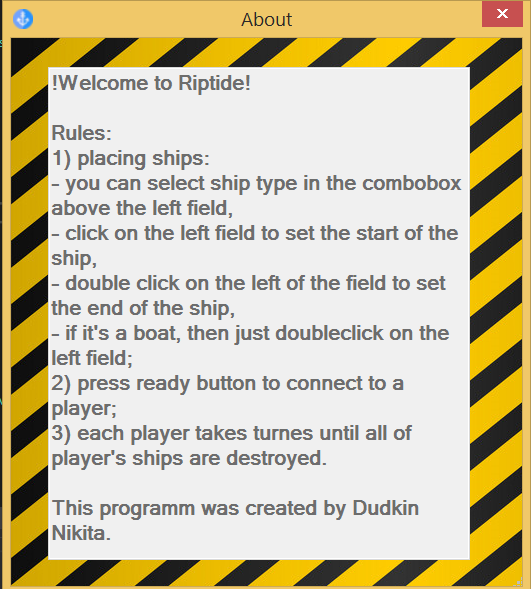


Рисунок 2.2

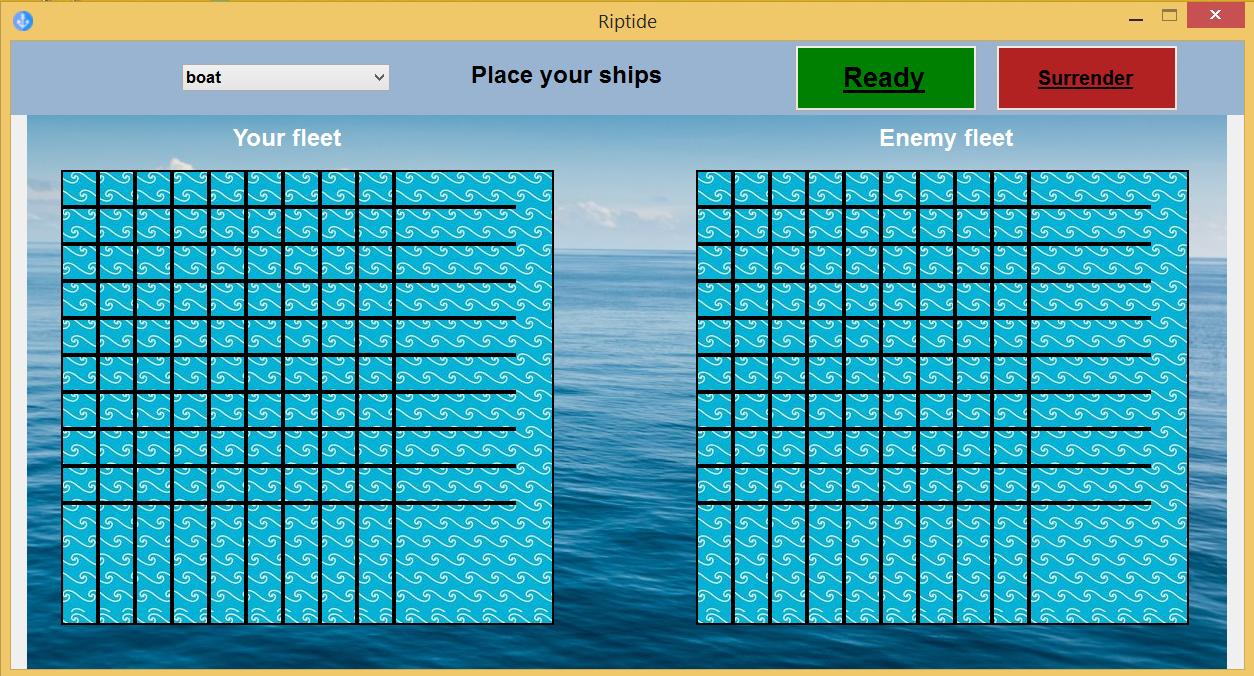


Рисунок 2.3

Пользователь должен начать размещать свой флот. Как уже говорилось

ранее, у игрока есть 4 типа кораблей (однопалубные, двухпалубных, трёхпалубные и четырёхпалубные). Пользователь выбирает нужный ему тип корабля и расставляет его у себя на карте. Для этого необходимо кликнуть левой кнопкой мыши по его началу и сделать двойной клик по месту его конца. Для однопалубных кораблей просто кликните дважды по нужной вам клетке.

Если пользователь вдруг перехотел играть или он хочет изменить расстановку флота, необходимо выбрать пункт “Surrender”. Так пользователь сможет выйти из игры.

Для подключения к оппоненту необходимо расставить все корабли, в противном случае будет выдаваться предупреждение, как показано на рисунке 2.3.

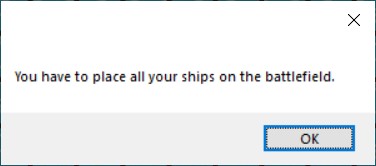


Рисунок 2.3

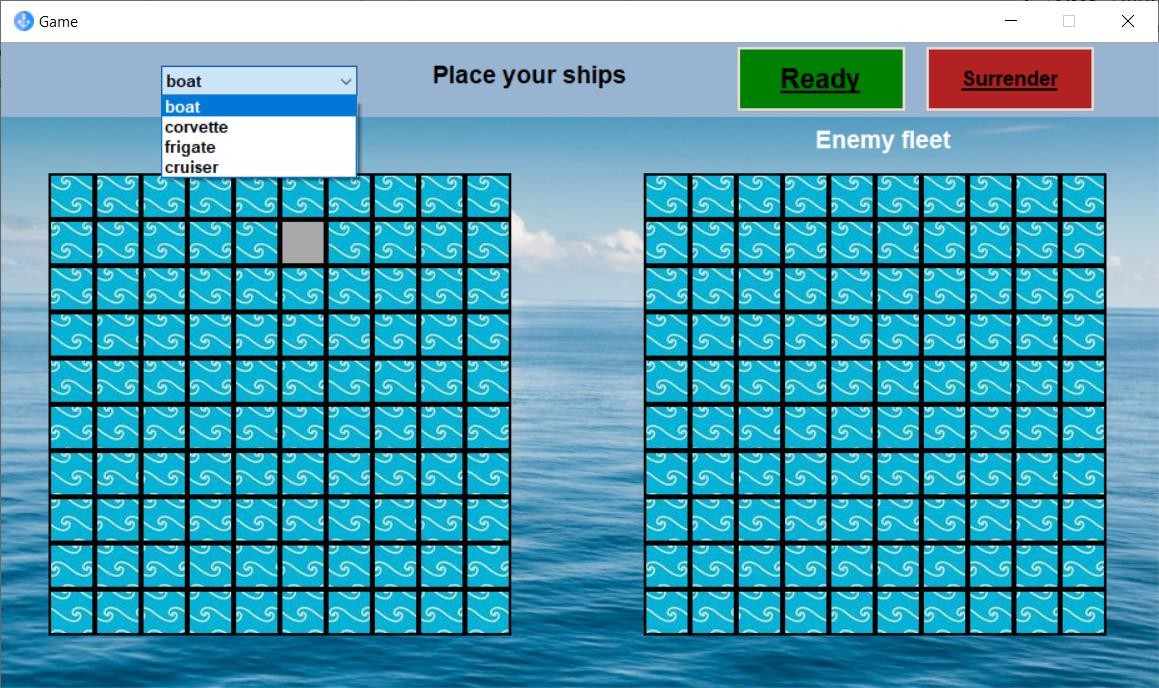
Выбрать тип корабля можно из выпадающего списка (рисунок 2.4)

Рисунок 2.4

Каждый игрок по очереди производит “выстрел”. В зависимости от того, попал ли игрок или нет, определяется его следующий ход. Если игрок попал в корабль противника, то он отображается красным на поле врага и игрок имеет право сделать ещё один ход, а если в корабль игорка попали, то он отображается красным на поле игрока и оппонент имеет право сделать ещё один ход. Рисунки 2.5,2.6.

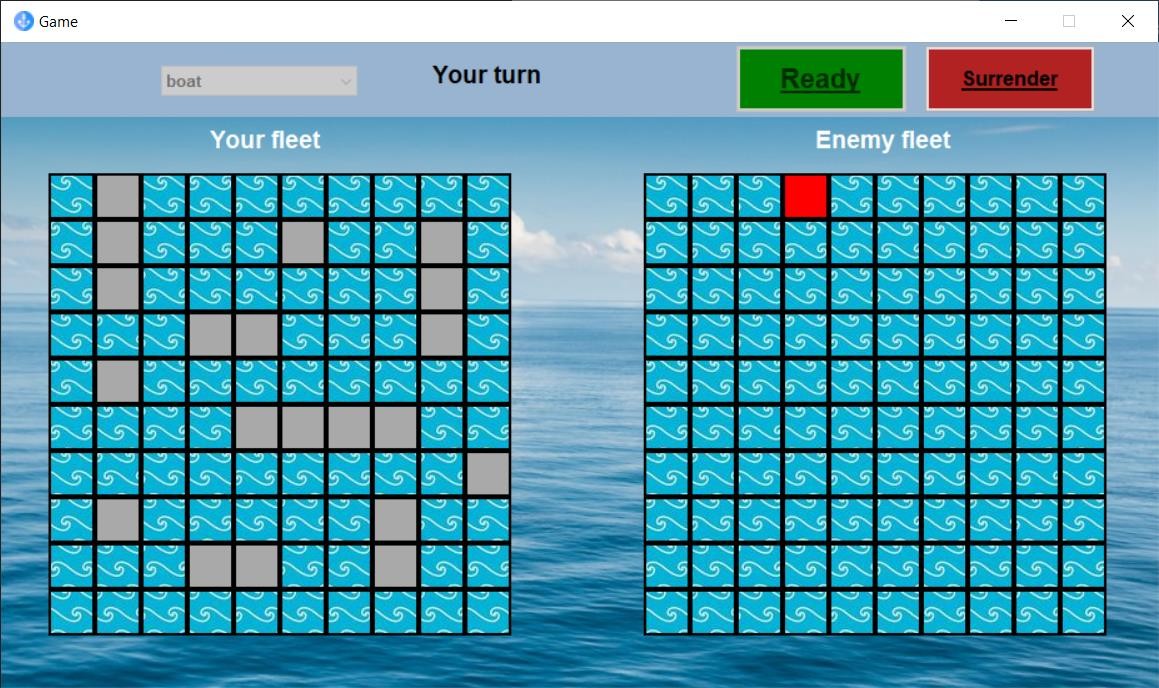


Рисунок 2.5

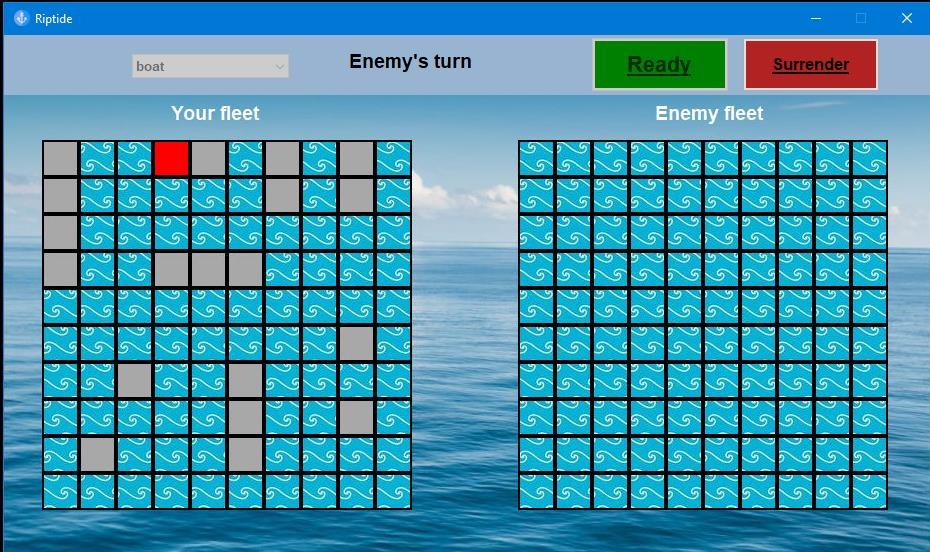
**

Рисунок 2.6

При уничтожении всех кораблей противника приходит оповещение о победе игрока, а также отправляется сообщение противнику о его проажении, рисунки 5.13 и 5.14.

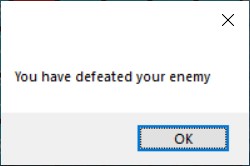


Рисунок 2.7

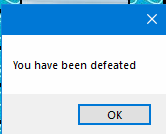


Рисунок 2.8

## 2.4 Работа с сетью

Взаимодействие с сетью для данного программного средства разработано на транспортном уровне. В качестве основы используются протоколы TCP и UDP.

В программном средстве реализованы следующие методы для работы с сетью: ListenerTCP(); ListenerUDP(); ConnectionCatcher().

ListenerUDP производит отправку шиоковещательных пакетов. При подключении устанавливает TCP соединение, отправляет сообщение о подключении и запускает асинхронную задачу с методом ListenerTCP.

## Таблица 3 - Структура данных алгоритм ListenerUDP

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| udpCatcher | UdpClient | Локальная переменная, которая  используется для широковещания |
| hostremote | IPEndPoint | Локальная переменная, которая служит для вычисления IP адреса  оппонента |
| data | byte[] | Локальная переменная, которая используется для получения  сообщения о подключении |
| ConnectMessa ge | byte[] | Локальная переменная, которая  используется для отправки сообщения о подключении |

ConnectionCatcher получает сообщение о подключении и запускает асинхронную задачу с методом ListenerTCP. Cхема приведена далее.

**Таблица 3 – Структура данных алгоритма ConnectionCatcher**

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| tcpListener | TcpListener | Локальная переменная, которая  используется для передачи данных |
| data | byte[] | Локальная переменная, которая  используется для получения сообщения о подключении |

ListenerTCP получает сообщения от оппонента и, в зависимости от типа сообщения, определенным образом обрабатывает эти сообщения.

## Таблица 3 – Структура данных алгоритма ListenerTCP

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| data | byte[] | Локальная переменная, которая используется для получения  сообщения от оппонента |
| Type | byte | Локальная переменная, которая используется для определения кода  отправленного сообщения |
| msg | byte[] | Локальная переменная, которая используется для получения координат от оппонента |

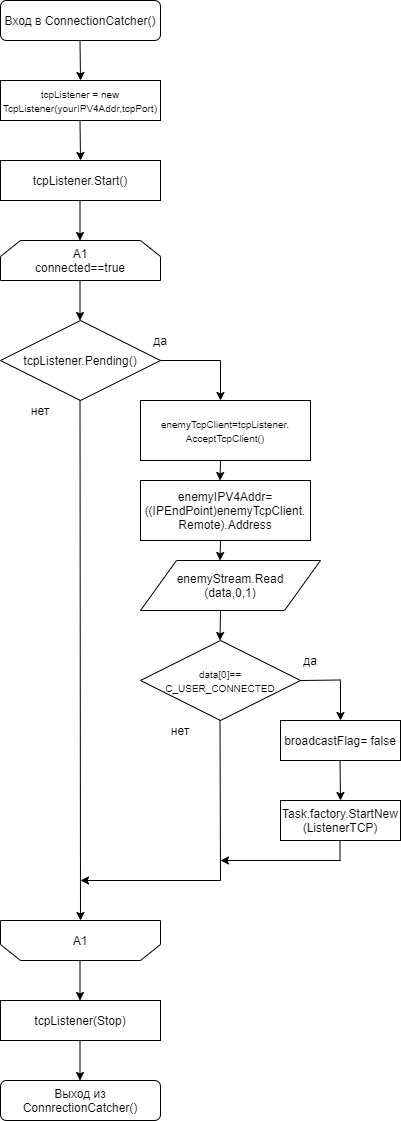


Схема ConnectionCatcher()

Продоление таблицы 3

|  |  |  |
| --- | --- | --- |
| msgx | byte | Локальная переменная, которая используется для получения координат  по оси X от оппонента |
| msgy | byte | Локальная переменная, которая  используется для получения координат по оси Y от оппонента |
| response | byte[] | Локальная переменная, которая  используется для отправки сообщения оппоненту |
| bmp | Bitmap | Локальная переменная, которая служит для создания поверхности для  рисования |
| graphics | Graphics | Локальная переменная, которая  служит для отрисовки выстрела противника |
| missCoords | byte[] | Локальная переменная, которая используется для получения координат  промаха оппонента |
| missCoordsX | int | Локальная переменная, которая используется для хранения координат  промаха оппонента по оси X |
| missCoordsY | int | Локальная переменная, которая используется для хранения координат  промаха оппонента по оси Y |
| hitCoords | byte[] | Локальная переменная, которая  используется для получения координат попадания оппонента |
| hitCoordsX | int | Локальная переменная, которая используется для хранения координат  попадания оппонента по оси X |
| hitCoordsY | int | Локальная переменная, которая  используется для хранения координат попадания оппонента по оси Y |
| victoryMsg | byte[] | Локальная переменная, которая используется для отправки сообщения  o победе |
| MForm | SyncInForm | Локальная переменная, которая  используется для закрытия главной формы |
| MainF | SyncInForm | Локальная переменная, которая  используется для закрытия главной формы |

## 2.4 Отрисовка кораблей

Для графики мы используем 4 метода.

Метод FieldGenerate() производит генерацию сетки на полях игровой формы.

Метод AreaFillX() производит заполнение трёх ячеек двумерного сассива area по индексам i и j вдоль координатной оси X при расстановке кораблей.

Метод AreaFilY() производит заполнение трёх ячеек двумерного сассива area по индексам i и j вдоль координатной оси Y при расстановке кораблей.

Метод SheepLocate() проверяет на возможность размещения корабля с количеством палуб deckNum, количеством кораблей определённого типа shipNum и координатами начала и конца. Если это возможно, то производится отрисовка корабля на поле field, пространства корабля и интервала вокруг него с помощью методов AreaFillX() и AreaFillY().

## Таблица 4 – структура алгоритма FieldGenerate

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| field | PictureBox | Параметр, который отвечает за  игровое поле для отрисовки сетки |
| bitmap | Bitmap | Локальная переменная, которая служит для создания поверхности для  рисования |
| graphics | Graphics | Локальная переменная, которая  служит для отрисовки сетки |
| i | int | Вспомогательная локальная  переменная-счётчик |

**Таблица 5-Структура алгоритма AreaFillX**

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| i | int | Параметр, который служит в качестве  индекса двумерного массива area |
| j | int | Параметр, который служит в качестве  индекса двумерного массива area |
| area | bool[,] | Параметр, который служит для  разметки кораблей |

## Таблица 6-Структура алгоритма AreaFillY

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| i | int | Параметр, который служит в качестве  индекса двумерного массива area |
| j | int | Параметр, который служит в качестве  индекса двумерного массива area |
| area | bool[,] | Параметр, который служит для  разметки кораблей |

**Таблица 7-Структура алгоритма ShipLocate**

|  |  |  |
| --- | --- | --- |
| shipNum | int | Параметр, в котором хранится информация о количестве кораблей  данного типа |
| deckNum | int | Параметр, в котором хранится  информация о количестве палуб данного типа корабля |
| start | Point | Параметр, в котором находятся  координаты начала корабля |
| end | Point | Параметр, в котором находятся  координаты конца корабля |
| area | bool[,] | Параметр, который служит для  разметки кораблей |
| iEnd | int | Локальная переменная, которая служит в качестве индекса конца  корабля в двумерном массиве area |
| jEnd | int | Локальная переменная, которая  служит в качестве индекса конца корабля в двумерном массиве area |
| iStart | int | Локальная переменная, которая служит в качестве индекса начала  корабля в двумерном массиве area |
| jStart | int | Локальная переменная, которая служит в качестве индекса начала корабля в двумерном массиве area |
| checker | bool | Локальная переменная, которая служит для проверки размещения  корабля |
| k | int | Вспомогательная локальная  переменная-счётчик |
| t | int | Локальная переменная, которая служит для индексации в двумерном  массиве area |
| YCoord | int | Локальная переменная, которая служит для расчёта координат начала корабляпо оси Y |
| field | PictureBox | Параметр, который отвечает за игровое поле для отрисовки кораблей |
| graphics | Graphics | Локальная переменная, которая  служит для отрисовки корабля |
| XCoord | int | Локальная переменная, которая  служит для расчёта координат начала корабляпо оси X |

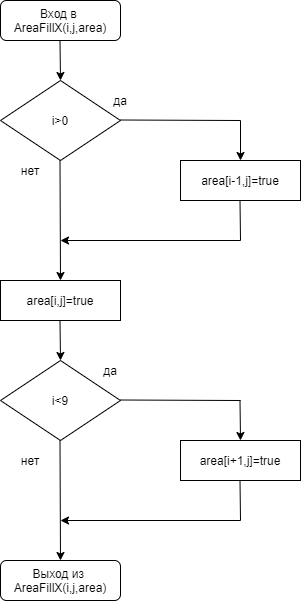


Схема AreaFillX

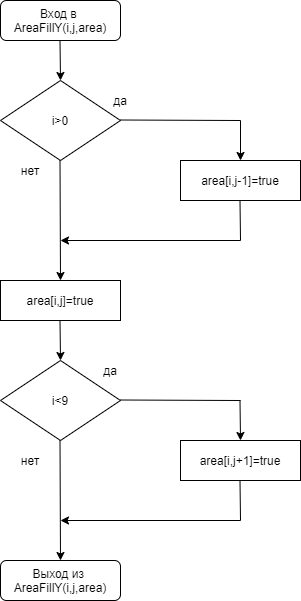


Схема AreaFillY

# 

# 3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## 3.1 Правила игры

Суть данной игры заключается в том, чтобы потопить весь флот противника до того, как он потопит весь твой флот. Если игрок попадает в корабль противника, то ему выпадает еще одна попытка выстрела. Чем меньше корабль – тем сложнее его уничтожить. Несмотря на то, что в этой игре немаловажное значение имеет просранственное мышление и логика, очень многое решает удача.

## 3.2 Управление

При расстановке кораблей необходимо 1 раз кликнуть левой кнопкой мыши по месту начала корабля и сделать двойной клик в месте окончания корабля. Если корабль однопалубный, то просто делаем двойной клик по месту, в котором хотим его расположить.При выстрелах кликаем левой кнопкой мыши в предполагаемый квадрат на карте противника.

# ЗАКЛЮЧЕНИЕ

На сегодняшний день с развитием компьютерной техники и интернета интерес к многопользовательским компьютерным играм увеличивается всё больше и больше. Большинство популярных современных многопользовательских игр имеют клиент-серверную архитектуру и являются трёхмерными. Для реализации трёхмерной графики используются различные методы, среди которых растеризация, ray-casting, ray-tracing и трассировка пути. В данном ПО был реализован один из выше перечисленных методов.

В рамках данного курсового проекта было разработано игровое программное средство «Морской бой». Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

* запуск приложения;
* отрисовка карты для игры;
* расстановка своих кораблей на поле боя;
* соединение двух игроков по сети;
* отрисовка своей карты и карты противника(с твоими выстрелами по его кораблям);
* отключение игрока во время игры;
* визуализация при попадании в корабль противника;
* отзывчивое управление.

Для успешной реализации вышеперечисленных функций потребовалось изучить работу с сокетами, объектно-ориентированную парадигму и возможности её применения в С, а также работу с формами в среде разработки VisualStudio 2019.

Существуют множество вариантов для улучшения данного программного обеспечения: например, можно добавить звуковое сопровождение, реализовать несколько одновременный партий с разными игроками, а также добавить различных помощников в уничтожении вражеских кораблей (бомбардировщиков, истребителей и торпед) и защите своих(системы противовоздушной обороны).

Данное программное обеспечение позволяет развить пространственное мышление, логику, память и просто провести время за игрой c друзьями.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Правила игры для российского морского боя [Электронный ресурс]. – Электронные данные. – Режим доступа:

<https://www.mosigra.ru/Face/Show/morskoqboir/rules/>

[2] C# для профессионалов. Тонкости программирования./Джон Скит. – 2014. -605 с.

[3] Библиотека MSDN[Электронный ресурс]. – Электронные данные. – режим доступа: <https://docs.microsoft.com/ru-ru/>

[4] CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# (Мастер-класс)/ Рихтер Дж. – 2013. -896 с.

[5] C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code/ Марк Д. Прайс . – 2019. 1017 c.

# ПРИЛОЖЕНИЕ А.

# Исходный код программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net.Sockets;

using System.Net;

namespace Riptide

{

public partial class GameForm : Form

{

public GameForm()

{

InitializeComponent();

FieldGenerate(YourFleet);

FieldGenerate(EnemyFleet);

EnemyFleet.Enabled = false;

ShipBox.SelectedIndex = 0;

}

private Point shipStart;

private Point shipEnd = new Point(-1, -1);

private bool[,] shipPosition = new bool[10, 10];

private bool[,] shipMap = new bool[10, 10];

private bool[,] shots = new bool[10, 10];

private int boatNum = 4;

private int corvetteNum = 3;

private int frigateNum = 2;

private int cruiserNum = 1;

private int tcpPort = 757;

private int udpPort = 755;

private int packetNum = 5;

private bool connected = false;

private bool enemyConnected = true;

private bool broadcastFlag = true;

private bool exitCheck = true;

private IPAddress yourIPV4Addr;

private TcpClient yourTcpClient;

private NetworkStream yourStream;

private IPAddress enemyIPV4Addr;

private TcpClient enemyTcpClient;

private NetworkStream enemyStream;

private int deckNum = 20;

private const byte C\_USER\_CONNECTED = 1;

private const byte C\_USER\_DISCONNECTED = 2;

private const byte C\_MESSAGE = 3;

private const byte C\_HIT = 4;

private const byte C\_MISS = 5;

private const byte C\_DEFEATED = 6;

private void FieldGenerate(PictureBox field)

{

var bitmap = new Bitmap(field.Width, field.Height);

field.Image = (Image)bitmap.Clone();

var graphics = Graphics.FromImage(bitmap);

graphics.DrawRectangle(new Pen(Color.Black, 4), 0, 0, field.Width, field.Height);

for (int i = 0; i < 10; i++)

{

graphics.DrawLine(new Pen(Color.Black, 4), 0, i \* 37, field.Height, i \* 37);

}

for (int i = 0; i < 10; i++)

{

graphics.DrawLine(new Pen(Color.Black, 4), i \* 37, 0, i \* 37, field.Width);

}

graphics.Dispose();

field.Image.Dispose();

field.Image = (Image)bitmap.Clone();

bitmap.Dispose();

}

private void ExitBtn\_Click(object sender, EventArgs e)

{

if (connected)

{

byte[] message = new byte[1];

message[0] = C\_USER\_DISCONNECTED;

enemyStream.Write(message, 0, 1);

exitCheck = false;

connected = false;

}

SyncInForm MF = this.Owner as SyncInForm;

MF.Close();

this.Close();

}

private void GameForm\_FormClosed(object sender, FormClosedEventArgs e)

{

if (connected)

{

if (exitCheck == true)

{

byte[] message = new byte[1];

message[0] = C\_USER\_DISCONNECTED;

enemyStream.Write(message, 0, 1);

}

connected = false;

SyncInForm MF = this.Owner as SyncInForm;

MF.Close();

}

}

private void YourFleet\_MouseClick(object sender, MouseEventArgs e)

{

shipStart = shipEnd;

shipEnd = new Point(e.X, e.Y);

int i = shipEnd.X / 37;

int j = shipEnd.Y / 37;

int xCoord = 37 \* i + 2;

int yCoord = 37 \* j + 2;

shipEnd = new Point(xCoord, yCoord);

}

private void EnemyFleet\_MouseClick(object sender, MouseEventArgs e)

{

Point target = new Point(e.X, e.Y);

int i = target.X / 37;

int j = target.Y / 37;

if (EnemyFleet.Enabled == true)

{

var sendMessage = new byte[3];

sendMessage[0] = C\_MESSAGE;

sendMessage[1] = BitConverter.GetBytes(i)[0];

sendMessage[2] = BitConverter.GetBytes(j)[0];

enemyStream.Write(sendMessage, 0, 3);

}

}

private void YourFleet\_MouseDoubleClick(object sender, MouseEventArgs e)

{

int shipType = ShipBox.SelectedIndex;

if (shipType != 0)

{

Point temp = shipStart;

if (shipEnd.X < shipStart.X)

{

shipStart.X = shipEnd.X;

shipEnd.X = temp.X;

}

else if (shipEnd.Y < shipStart.Y)

{

shipStart.Y = shipEnd.Y;

shipEnd.Y = temp.Y;

}

}

switch (shipType)

{

case 0:

if (boatNum != 0)

{

int i = (shipEnd.X - 2) / 37;

int j = (shipEnd.Y - 2) / 37;

if (shipPosition[i, j] == false)

{

Bitmap bmp = new Bitmap(YourFleet.Image, YourFleet.Width, YourFleet.Height);

var graphics = Graphics.FromImage(bmp);

graphics.FillRectangle(new SolidBrush(Color.DarkGray), shipEnd.X, shipEnd.Y, 33, 33);

graphics.Dispose();

YourFleet.Image.Dispose();

YourFleet.Image = (Image)bmp.Clone();

bmp.Dispose();

boatNum--;

if (j > 0)

{

AreaFillX(i, j - 1, shipPosition);

}

AreaFillX(i, j, shipPosition);

if (j < 9)

{

AreaFillX(i, j + 1, shipPosition);

}

shipMap[i, j] = true;

}

shipEnd = new Point(-1, -1);

}

break;

case 1:

if (corvetteNum != 0)

{

if (((shipEnd.X - shipStart.X) == 37) || ((shipEnd.Y - shipStart.Y) == 37))

{

corvetteNum = ShipLocate(YourFleet, corvetteNum, 2, shipStart, shipEnd, shipPosition);

}

shipEnd = new Point(-1, -1);

}

break;

case 2:

if (frigateNum != 0)

{

if (((shipEnd.X - shipStart.X) == 37 \* 2) || ((shipEnd.Y - shipStart.Y) == 37 \* 2))

{

frigateNum = ShipLocate(YourFleet, frigateNum, 3, shipStart, shipEnd, shipPosition);

}

shipEnd = new Point(-1, -1);

}

break;

case 3:

if (cruiserNum != 0)

{

if (((shipEnd.X - shipStart.X) == 37 \* 3) || ((shipEnd.Y - shipStart.Y) == 37 \* 3))

{

cruiserNum = ShipLocate(YourFleet, cruiserNum, 4, shipStart, shipEnd, shipPosition);

}

shipEnd = new Point(-1, -1);

}

break;

}

}

private void AreaFillX(int i, int j, bool[,] area)

{

if (i > 0)

{

area[i - 1, j] = true;

}

area[i, j] = true;

if (i < 9)

{

area[i + 1, j] = true;

}

}

private void AreaFillY(int i, int j, bool[,] area)

{

if (j > 0)

{

area[i, j - 1] = true;

}

area[i, j] = true;

if (j < 9)

{

area[i, j + 1] = true;

}

}

private int ShipLocate(PictureBox field, int shipNum, int deckNum, Point start, Point end, bool[,] area)

{

int iEnd = (end.X - 2) / 37;

int jEnd = (end.Y - 2) / 37;

int iStart = (start.X - 2) / 37;

int jStart = (start.Y - 2) / 37;

if (end.X == start.X)

{

bool checker = false;

for (int k = jStart; k <= jEnd; k++)

{

if (area[iStart, k] == true)

{

checker = true;

}

}

if (checker == false)

{

int t = jStart;

Bitmap bmp = new Bitmap(field.Image, field.Width, field.Height);

var graphics = Graphics.FromImage(bmp);

if (t > 0)

{

AreaFillX(iStart, t - 1, area);

}

for (int k = 0; k < deckNum; k++)

{

int YCoord = start.Y + k \* 37;

graphics.FillRectangle(new SolidBrush(Color.DarkGray), end.X, YCoord, 33, 33);

if (iStart > 0)

{

area[iStart - 1, t] = true;

}

area[iStart, t] = true;

shipMap[iStart, t] = true;

if (iStart < 9)

{

area[iStart + 1, t] = true;

}

t++;

}

graphics.Dispose();

field.Image.Dispose();

field.Image = (Image)bmp.Clone();

bmp.Dispose();

shipNum--;

if (t < 10)

{

AreaFillX(iStart, t, area);

}

}

}

else if (end.Y == start.Y)

{

bool checker = false;

for (int k = iStart; k <= iEnd; k++)

{

if (area[k, jStart] == true)

{

checker = true;

}

}

if (checker == false)

{

int t = iStart;

Bitmap bmp = new Bitmap(field.Image, field.Width, field.Height);

var graphics = Graphics.FromImage(bmp);

if (t > 0)

{

AreaFillY(t - 1, jStart, area);

}

for (int k = 0; k < deckNum; k++)

{

int XCoord = start.X + k \* 37;

graphics.FillRectangle(new SolidBrush(Color.DarkGray), XCoord, end.Y, 33, 33);

if (jStart > 0)

{

area[t, jStart - 1] = true;

}

area[t, jStart] = true;

shipMap[t, jStart] = true;

if (jStart < 9)

{

area[t, jStart + 1] = true;

}

t++;

}

graphics.Dispose();

field.Image.Dispose();

field.Image = (Image)bmp.Clone();

bmp.Dispose();

shipNum--;

if (t < 10)

{

AreaFillY(t, jStart, area);

}

}

}

return shipNum;

}

private void ReadyBtn\_Click(object sender, EventArgs e)

{

if ((boatNum == 0) && (corvetteNum == 0) && (frigateNum == 0) && (cruiserNum == 0))

{

YourFleet.Enabled = false;

ReadyBtn.Enabled = false;

ExitBtn.Enabled = false;

ShipBox.Enabled = false;

TurnIdent.Text = "Waiting for connections";

IPHostEntry host = Dns.GetHostEntry(Dns.GetHostName());

foreach (IPAddress tempIp in host.AddressList)

{

if (tempIp.AddressFamily == AddressFamily.InterNetwork)

{

yourIPV4Addr = tempIp;

}

}

Task.Factory.StartNew(() => ListenerUDP());

connected = true;

UdpClient udpClient = new UdpClient("255.255.255.255", udpPort);

byte[] messageIP = Encoding.Unicode.GetBytes(yourIPV4Addr.ToString());

udpClient.EnableBroadcast = true;

Task.Factory.StartNew(() => ConnectionCatcher());

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

{

udpClient.Send(messageIP, yourIPV4Addr.ToString().Length);

}

udpClient.Dispose();

}

else

{

MessageBox.Show("You have to place all your ships on the battlefield.");

}

}

private void ListenerUDP()

{

var udpCatcher = new UdpClient(udpPort);

udpCatcher.EnableBroadcast = true;

while (broadcastFlag)

{

IPEndPoint hostremote = null;

var data = udpCatcher.Receive(ref hostremote);

if (connected)

{

enemyIPV4Addr = hostremote.Address;

if (!hostremote.Address.Equals(yourIPV4Addr))

{

yourTcpClient = new TcpClient();

yourTcpClient.Connect(new IPEndPoint(enemyIPV4Addr, tcpPort));

yourStream = yourTcpClient.GetStream();

enemyStream = yourTcpClient.GetStream();

var ConnectMessage = new byte[1];

ConnectMessage[0] = C\_USER\_CONNECTED;

yourStream.Write(ConnectMessage, 0, ConnectMessage.Length);

enemyTcpClient = yourTcpClient;

this.Invoke(new MethodInvoker(() =>

{

TurnIdent.Text = "Your turn";

EnemyFleet.Enabled = true;

ExitBtn.Enabled = true;

}));

broadcastFlag = false;

Task.Factory.StartNew(() => ListenerTCP());

}

}

}

}

private void ConnectionCatcher()

{

TcpListener tcpListener = new TcpListener(yourIPV4Addr, tcpPort);

tcpListener.Start();

while (connected)

{

if (tcpListener.Pending())

{

enemyTcpClient = tcpListener.AcceptTcpClient();

enemyIPV4Addr = ((IPEndPoint)enemyTcpClient.Client.RemoteEndPoint).Address;

enemyStream = enemyTcpClient.GetStream();

byte[] data = new byte[1];

enemyStream.Read(data, 0, 1);

if (data[0] == C\_USER\_CONNECTED)

{

this.Invoke(new MethodInvoker(() =>

{

TurnIdent.Text = "Enemy's turn";

ExitBtn.Enabled = true;

}));

broadcastFlag = false;

Task.Factory.StartNew(() => ListenerTCP());

}

}

}

tcpListener.Stop();

}

private void ListenerTCP()

{

while (enemyConnected)

{

if (enemyStream.DataAvailable)

{

byte[] data = new byte[1];

enemyStream.Read(data, 0, 1);

byte Type = data[0];

switch (Type)

{

case C\_MESSAGE:

byte[] msg = new byte[2];

enemyStream.Read(msg, 0, 1);

enemyStream.Read(msg, 1, 1);

int msgx = msg[0];

int msgy = msg[1];

byte[] response = new byte[3];

this.Invoke(new MethodInvoker(() =>

{

Bitmap bmp = new Bitmap(YourFleet.Image, YourFleet.Width, YourFleet.Height);

var graphics = Graphics.FromImage(bmp);

if ((shipMap[msgx, msgy] == true) && (shots[msgx, msgy] == false))

{

shots[msgx, msgy] = true;

response[0] = C\_HIT;

response[1] = msg[0];

response[2] = msg[1];

enemyStream.Write(response, 0, 3);

graphics.FillRectangle(new SolidBrush(Color.Red), msgx \* 37 + 2, msgy \* 37 + 2, 33, 33);

}

else if (shots[msgx, msgy] == false)

{

shots[msgx, msgy] = true;

response[0] = C\_MISS;

response[1] = msg[0];

response[2] = msg[1];

enemyStream.Write(response, 0, 3);

graphics.FillRectangle(new SolidBrush(Color.Black), msgx \* 37 + 2, msgy \* 37 + 2, 33, 33);

TurnIdent.Text = "Your turn";

EnemyFleet.Enabled = true;

}

graphics.Dispose();

YourFleet.Image.Dispose();

YourFleet.Image = (Image)bmp.Clone();

bmp.Dispose();

}));

break;

case C\_MISS:

byte[] missCoords = new byte[2];

enemyStream.Read(missCoords, 0, 1);

enemyStream.Read(missCoords, 1, 1);

int missCoordsX = missCoords[0];

int missCoordsY = missCoords[1];

this.Invoke(new MethodInvoker(() =>

{

Bitmap bmp = new Bitmap(EnemyFleet.Image, EnemyFleet.Width, EnemyFleet.Height);

var graphics = Graphics.FromImage(bmp);

graphics.FillRectangle(new SolidBrush(Color.Black), missCoordsX \* 37 + 2, missCoordsY \* 37 + 2, 33, 33);

graphics.Dispose();

EnemyFleet.Image.Dispose();

EnemyFleet.Image = (Image)bmp.Clone();

bmp.Dispose();

TurnIdent.Text = "Enemy's turn";

EnemyFleet.Enabled = false;

}));

break;

case C\_HIT:

byte[] hitCoords = new byte[2];

enemyStream.Read(hitCoords, 0, 1);

enemyStream.Read(hitCoords, 1, 1);

int hitCoordsX = hitCoords[0];

int hitCoordsY = hitCoords[1];

this.Invoke(new MethodInvoker(() =>

{

Bitmap bmp = new Bitmap(EnemyFleet.Image, EnemyFleet.Width, EnemyFleet.Height);

var graphics = Graphics.FromImage(bmp);

graphics.FillRectangle(new SolidBrush(Color.Red), hitCoordsX \* 37 + 2, hitCoordsY \* 37 + 2, 33, 33);

graphics.Dispose();

EnemyFleet.Image.Dispose();

EnemyFleet.Image = (Image)bmp.Clone();

bmp.Dispose();

deckNum--;

if (deckNum == 0)

{

byte[] victoryMsg = new byte[1];

victoryMsg[0] = C\_DEFEATED;

enemyStream.Write(victoryMsg, 0, 1);

MessageBox.Show("You have defeated your enemy");

enemyConnected = false;

connected = false;

exitCheck = false;

SyncInForm MForm = this.Owner as SyncInForm;

MForm.Close();

this.Close();

}

}));

break;

case C\_DEFEATED:

enemyConnected = false;

connected = false;

this.Invoke(new MethodInvoker(() =>

{

MessageBox.Show("You have been defeated");

exitCheck = false;

SyncInForm MainF = this.Owner as SyncInForm;

MainF.Close();

this.Close();

}));

break;

case C\_USER\_DISCONNECTED:

enemyConnected = false;

connected = false;

this.Invoke(new MethodInvoker(() =>

{

MessageBox.Show("Enemy disconnected");

exitCheck = false;

SyncInForm MainF = this.Owner as SyncInForm;

MainF.Close();

this.Close();

}));

break;

}

}

}

}

}

}