СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные технологии и мобильные приложения играют ключевую роль в повседневной жизни современного человека, предоставляя разнообразные возможности для развлечения, обучения и общения. Одной из популярных многопользовательских онлайн-игр является обновленная версия классической игры «Танчики 1990», которая привлекает внимание игроков своей динамикой и соревновательным геймплеем.

«Танчики 1990» изначально была однопользовательской игрой, в которой игрок управлял танком, сражаясь с вражескими танками и защищая свою базу. Теперь, благодаря современным технологиям, игра получила новую жизнь в виде многопользовательской онлайн-версии, где игроки могут сражаться друг с другом в режиме реального времени.

Цель данной курсовой работы заключается в исследовании и разработке программного клиента для многопользовательской версии игры «Танчики 1990», который подключается к оригинальным серверам игры и позволяет пользователю играть с реальными игроками. В рамках проекта будут изучены протоколы связи с серверами «Танчики 1990», реализованы основные механики игрового процесса и разработан интуитивно понятный пользовательский интерфейс. Основное внимание уделяется обеспечению плавного и стабильного игрового процесса, а также интеграции с оригинальными серверами для поддержки многопользовательских сессий.

В данной пояснительной записке отражены следующие разделы:

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;
2. Анализ требований к программному средству и разработка функциональных требований;
3. Проектирование программного средства;
4. Создание (конструирование) программного средства;
5. Тестирование, проверка работоспособности и анализ полученных результатов;
6. Руководство по установке и использованию;

1 Анализ литературных источников

1.1 История

«Танчики 1990» — классическая аркадная игра, выпущенная в 1990 году компанией Namco для игровой консоли Nintendo Entertainment System (NES). В оригинальной версии игры игрок управлял танком, сражаясь с вражескими танками и защищая свою базу. Игра предлагала несколько уровней с увеличивающейся сложностью, что делало её захватывающей и интересной для широкого круга игроков.

Игровой процесс был простым, но увлекательным: игроку нужно было уничтожать вражеские танки, избегая попадания вражеских снарядов и защищая собственную базу от захвата. Враги появлялись волнами, и для перехода на следующий уровень необходимо было уничтожить все вражеские танки. Игра стала очень популярной благодаря своей доступности и возможности играть вдвоем, что добавляло элемент соревновательности.

Переход в онлайн:

С развитием интернета и онлайн-гейминга популярные классические игры, такие как «Танчики 1990», начали адаптироваться для многопользовательской игры в сети.Первая онлайн-версия «Танчиков 1990» появилась в начале 2000-х годов, когда разработчики начали создавать клоны и ремейки оригинальной игры для ПК и веб-браузеров. Эти версии позволяли игрокам соединяться через интернет и сражаться друг с другом в режиме реального времени. Возможность играть с реальными людьми, а не с ИИ, добавила новый уровень сложности и интереса к игре.

Многопользовательские версии и их развитие:

Многопользовательские версии «Танчиков 1990» быстро стали популярными. Разработчики продолжали улучшать графику и добавлять новые игровые механики, чтобы поддерживать интерес игроков.

Современные многопользовательские версии «Танчиков 1990» доступны на различных платформах, включая мобильные устройства, что сделало игру еще более доступной. Например, такие игры, как «Battle City» и «Tank Force», вдохновленные оригинальными «Танчиками 1990», предлагают продвинутую графику и многочисленные режимы игры, сохраняя при этом дух классической игры. [1]

1.2 Анализ прототипов

Перед разработкой программного клиента и сервера для игры «Танки Онлайн» важно провести анализ прототипов, чтобы понять их особенности, преимущества и недостатки. Этот анализ поможет определить лучшие практики и основные характеристики, которые должны быть учтены при создании программного клиента.

1.2.1 Оригинальные «Танчики 1990»

Оригинальная версия «Танчики 1990», разработанная Namco, представляет собой классическую аркадную игру, в которой игроки управляют танками, сражаясь против вражеских танков и защищая свою базу. Игровой процесс заключается в уничтожении врагов и предотвращении их проникновения в базу игрока[2]. Простота игрового процесса и возможность играть вдвоем сделали «Танчики 1990» популярной среди игроков всех возрастов. Основные особенности игры включают:
Однопользовательский и двухпользовательский режимы, позволяющие играть как в одиночку, так и вместе с другом[2].
Простое управление, подходящее для игроков любого уровня опыта.
Разнообразие уровней с возрастающей сложностью и различными типами врагов.

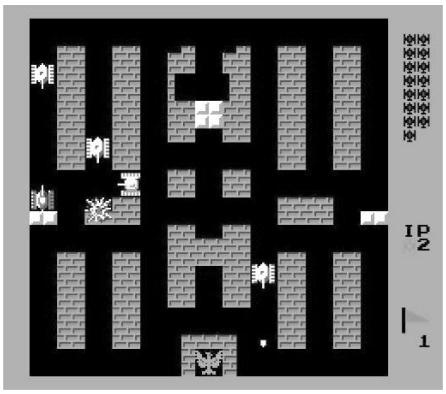

Рисунок 1.2.1 – Танчики 1990

1.2.2 Tank Force (1991)

Оригинальная версия игры "Tank Force" (1991), созданная компанией Namco для аркадных автоматов и консолей Nintendo, представляет собой классическую аркадную игру, в которой игрок управляет танком, сражаясь против вражеских танков и защищая свою базу. Простота игрового процесса и захватывающий геймплей сделали Tank Force популярной иконой в мире видеоигр. Основные особенности игры включают:

Простое управление обеспечивает игроку возможность управлять танком с помощью клавиш на клавиатуре или джойстика на аркадном автомате, что делает управление интуитивно понятным и доступным для игроков любого уровня опыта.
Защита базы становится важной частью игрового процесса, поскольку игрок должен уничтожать вражеские танки, которые пытаются захватить базу. В случае разрушения базы игра завершается.
Многоуровневая структура игры предлагает разнообразные карты с различными препятствиями и уровнями сложности, что добавляет интерес и вызов для игроков.[3]

Рисунок 1.2.1 – Tank Force

1.3 Постановка задачи

В своей курсовой работе я поставила следующие задачи:

1. Проанализировать требования, указанные в задании на курсовое проектирование.
2. Изучить существующие прототипы, их достоинства и недостатки.
3. Разработать программное средство, позволяющее пользователю запустить сервер или подключаться к существующему серверу «Танки Онлайн» и играть с реальными игроками.
4. Обеспечить плавный и стабильный игровой процесс.
5. Создать интуитивно понятный пользовательский интерфейс.
6. Отладить и проанализировать программное средство.

Для разработки программного клиента будут использоваться язык программирования C# и библиотека Windows.Forms для реализации игрового интерфейса.

2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Описание функциональных требований

На основе проведенного анализа и с учетом требований, указанных в задании на курсовое проектирование, выделяются следующие основные требования к функционалу программного средства:

1. Создание сервера игры «Танки Онлайн» и реализации на нем игровой логики.
2. Создание Клиента для подключения к запущенному серверу.
3. Реализация протокола, для взаимодействия клиента и сервера игры «Танки Онлайн».
4. Реализация механизма обновления игрового состояния с сервера на клиенте с минимальной задержкой для обеспечения плавного игрового процесса.
5. Отображение игрового поля с учетом данных, полученных от серверов игры.
6. Реализация управления танком с помощью клавиатуры.
7. Визуализация движения танка игрока и других игровых объектов на экране.
8. Создание интуитивно понятного пользовательского интерфейса, включая элементы управления, информационные панели и меню.

2.2 Спецификация функциональных требований

1. Создание игрового сервера «Танки Онлайн»:
    1. Программное средство должно поддерживать подключение 2х игроков.
    2. Требуется обработка начала игры, окончания и ее базовой логики.
2. Создание подключения к оригинальным серверу игры «Танки Онлайн»:
    1. Программное средство должно уметь устанавливать соединение с серверами игры «Танки Онлайн» по протоколу UDP/IP[4].
    2. Требуется обработка аутентификации и обмена данных между клиентом и сервером для корректной игровой сессии.
3. Реализация протокола, по которому взаимодействуют клиент и сервер игры «Танки Онлайн».
4. Реализация механизма обновления игрового состояния с сервера на клиенте с минимальной задержкой для обеспечения плавного игрового процесса:
    1. Требуется реализация эффективного механизма обновления игрового состояния с сервера на клиенте с минимальной задержкой для обеспечения плавного и непрерывного игрового процесса. Логика игры:

5. Отображение игрового поля с учетом данных, полученных от серверов игры:
    1. Программное средство должно обеспечить отображение игрового поля, а также всех игровых объектов, полученных от серверов игры «Танки Онлайн».
    2. Требуется корректная синхронизация отображаемых данных с актуальным состоянием игры на сервере.
6. Реализация управления танком с помощью клавиатуры.
7. Визуализация движения танков и других игровых объектов на экране.
    1. Программное средство должно обеспечить плавную визуализацию движения танков и других игровых объектов на экране пользователя.
    2. Для создания плавного и реалистичного визуального опыта необходима анимация движения танка и других игровых объектов при их перемещении по игровому полю.
    3. Программное средство должно корректно определять и обрабатывать коллизии между танком и другими объектами игры, например, при соприкосновении со стенами или другими танками.
8. Создание интуитивно понятного пользовательского интерфейса, включая элементы управления, информационные панели и меню.
    1. Программное средство должно иметь удобный и понятный пользовательский интерфейс с элементами управления, информационными панелями и меню.

3. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Алгоритм обработки сообщения от клиента

Этот алгоритм выполняет расшифровку сообщения от клиента. Затем он анализирует является ли он новым пользователем, после чего либо добавляет его в список пользователей, либо работает как со старым.

Рисунок 3.1 – Алгоритм обработки сообщения от клиента

3.2 Алгоритм отправки сообщения клиенту

Данный алгоритм формирует сообщение, содержащее информацию, необходимую для отрисовки игрового поля, после чего отправляет его всем пользователям.

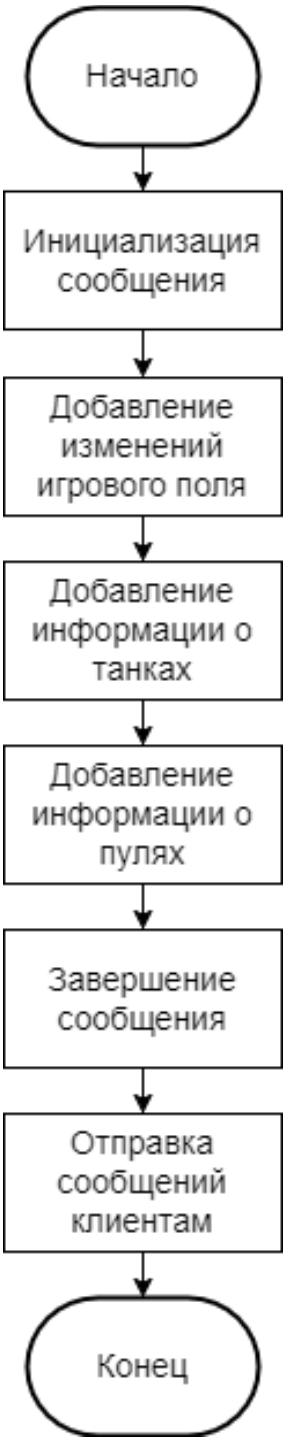

Рисунок 3.2 – Блок-схема алгоритма отправки сообщения клиенту

3.3 Алгоритм обработки сообщения от сервера

Этот алгоритм обеспечивает обновление информации об игровом поле для отрисовки и проверяет игру на окончание, так же отсеивает сообщения, отправленные другими пользователями.

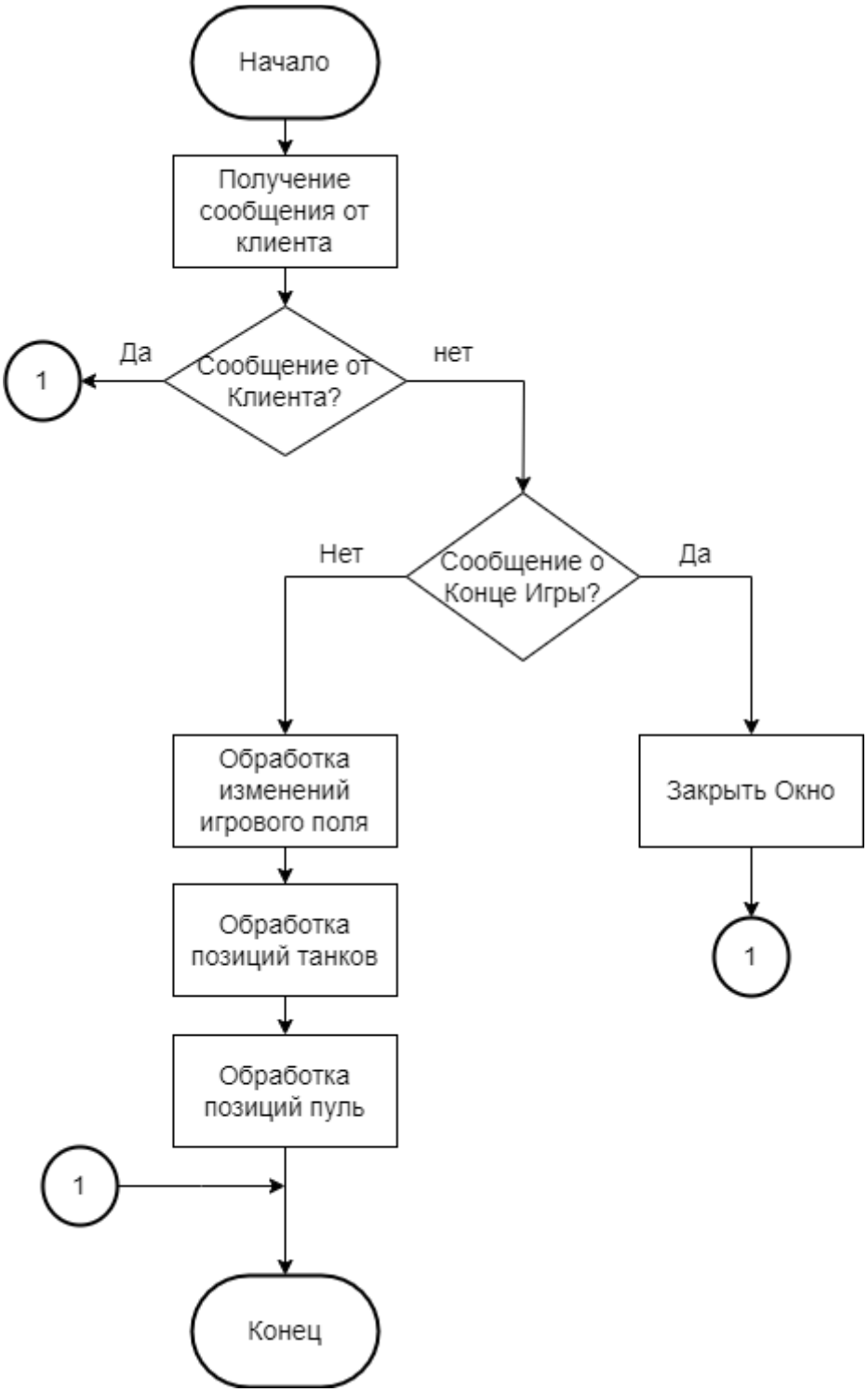

Рисунок 3.3 – Блок-схема алгоритма обработки сообщения от сервера

3.4 Алгоритм отправки сообщения серверу

Этот алгоритм создает сообщение с информацией о нажатых кнопках движения и добавленных пулях и передает его серверу, указывая свое имя, для идентификации.

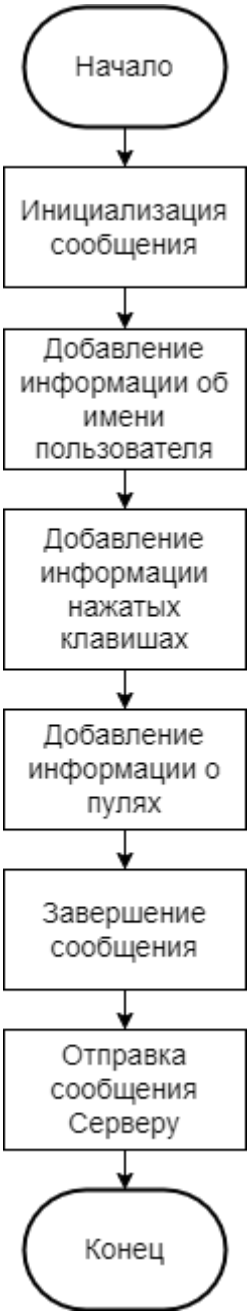

Рисунок 3.4 – Блок-схема алгоритма отправки сообщения серверу

3.5 Алгоритм движения игроков

Этот алгоритм реализует передвижение игроков и пуль, с просчетом возникающих взаимодействий, он запускается по таймеру на сервере.

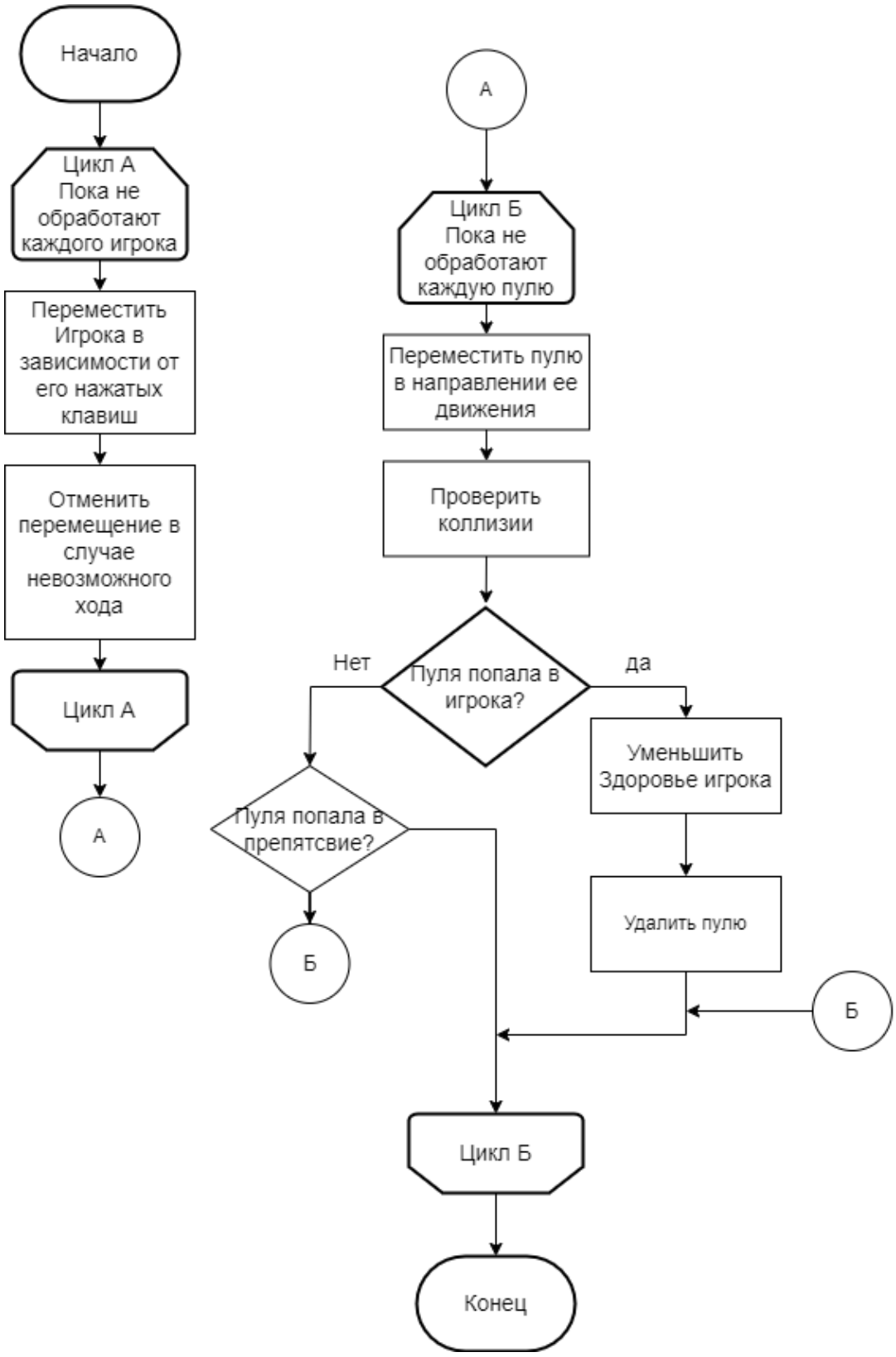

Рисунок 3.5 – Блок-схема алгоритма движения игроков

4. КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

4.1 Проектирование модулей программного средства

Разработка программного средства выполняется на основе алгоритмов, которые были представлены в разделе 3, а также на основе функциональных требований, представленных в разделе 2.

4.2 Структура модулей программы

Программа состоит из следующих файлов:

1. Файл Program.cs входной точки сервера;
2. Файл GameClasses.dll представляет собой библиотеку, которая описывает необходимые для игры объекты.
    1. Класс GameFiled представляет собой массив клеток, которые представляют собой игровое поле.
    2. Класс Block представляет собой базовый класс для клетки поля.
    3. Класс BackGround представляет собой Пустую клетку поля.
    4. Класс UnbBrick представляет собой стенку, которую нельзя сломать.
    5. Класс Brick представляет собой стенку, которую можно сломать.
    6. Класс Tank представляет собой класс, описывающий такнк.
    7. Класс TankPlayer, аналогичен Tank, но имеет другие текстуры, для отрисовки на стороне клиента.
    8. Класс bullet представляет собой структуру для описания пули на поле.
    9. Класс ThreadSafeList представляет собой потокобезопасную реализацию списка.
3. Файл UdpServer.cs представляет собой класс, который описывает работу udp севера.
4. Файл MainForm.cs представляет собой класс, который описывает объект главного окна клиента.
5. Файл InitForm.cs представляет собой класс, который описывает объект окна инициализации клиента.
6. Файл UdpClient.cs представляет собой класс, который описывает работу udp клиента.

4.3 Описание модуля Program

Модуль Main является точка входа в программу. Он запускает приложение «MySlither.io», инициализируя оконный интерфейс и начиная его выполнение.

Таблица 4.3 - Методы модуля Program

Имя метода	**Описание**	**Заголовок метода**	**Имя параметра**	**Назначение параметра**
main	Точка входа в программу. В нем создается сервер, и начинается выполнение работы программы.	ыtatic async Task Main	args	Массив аргументов командной строки

4.4 Описание модуля UdpServer

Модуль GameClasses представляет Udpсервер, который отвечает за общение с игроками и просчет игрового состояния.

Таблица 4.4 - Методы модуля UdpServer

Имя метода	**Описание**	**Заголовок метода**	**Имя параметра**	**Назначение параметра**
StartGame	Начиает игру	private void StartGame()	-	-
EndGame	Заканчивает игру, и отправляет игрокам сообщение	Public void EndGAme()	-	-
UdpServer	Метод, создающий экземпляр класса UdpServer	public UdpServer(string ipAddress, int port)	ipAddress	IP адрес сервера
			int	Порт Сервера

Продолжение таблицы 4.4

public async Task StartAsync	Запускает сервер	StartAsync	-	-
HandleIncomingMessagesAsync	Обрабатывает входящее сообщение в параллельном потоке	private async Task HandleIncomingMessagesAsync()	-	-
SendPeriodicMessages	Отправляет игрокам периодические сообщения о состоянии игры	private void SendPeriodicMessages(object state)	state	Текущее состояние данного экземпляра UdpServer
GetSubStringInd	Возвращает подстроку, которая начинается в окончании key1 и заканчивается в начале key2	static string GetSubStringInd(string input, string key1, string key2)	Input	Входная строка
			Key1	Начальный ориентир
			Key2	Конечный ориентир
MoveUpCheck	Проверяет коллизию при передвижении танка вверх	static public bool MoveUpCheck(double x, double y, GameFiled gameFiled)	x	Ожидаемая координата по x
			y	Ожидаемая координата по y
			gameFiled	Текущее состояние игрового поля

Продолжение таблицы 4.4

MoveRightCheck	Проверяет коллизию при передвижении танка вправо	static public bool MoveUpCheck(double x, double y, GameFiled gameFiled)	x	Ожидаемая координата по x
			y	Ожидаемая координата по y
			gameFiled	Текущее состояние игрового поля
MoveLeftCheck	Проверяет коллизию при передвижении танка влево	static public bool MoveUpCheck(double x, double y, GameFiled gameFiled)	x	Ожидаемая координата по x
			y	Ожидаемая координата по y
			gameFiled	Текущее состояние игрового поля
MoveDownCheck	Проверяет коллизию при передвижении танка вниз	static public bool MoveUpCheck(double x, double y, GameFiled gameFiled)	x	Ожидаемая координата по x
			y	Ожидаемая координата по y
			gameFiled	Текущее состояние игрового поля
TimerTickCallback	Производит перемещение движущихся обьектов, и проверяет коллизии	static void TimerTickCallback (Object state)	state	Текущее состояние сервера

4.5 Описание модуля MainForm

Модуль MainForm представляет собой код клиентского окна, с отрисовкой игрового поля.

Таблица 4.5 - Методы модуля MainForm

Имя метода	**Описание**	**Заголовок метода**	**Имя параметра**	**Назначение параметра**
MainForm	Создает экземпляр класса MainForm	public MainForm()	-	-
ReceiveMessageTread	Код потока для приема и обработки входящих от сервера сообщений	private async void ReceiveMessageTread()	-	-
SendToServerThread	Код для формирования и отправки сообщения сервера	private async void SendToServerThread()	-	-
MainForm_KeyDown	Код для обработки нажатия на клавишу	private void MainForm_KeyDown(object sender, KeyEventArgs e)	sender	Отправитель вызова
			e	Информация о событии
MainForm_Load	Обработчик загрузки главного окна	private async void MainForm_Load(object sender, EventArgs e)	sender	Отправитель вызова
			e	Информация о событии

Продолжение таблицы 4.5

MainForm_KeyUp	Код для обработки отжатия клавиши	private void MainForm_KeyUp (object sender, KeyEventArgs e)	sender	Отправитель вызова
			e	Информация о событии
GetSubStringInd	Возвращает подстроку, которая начинается в окончании key1 и заканчивается в начале key2	static string GetSubStringInd(string input, string key1, string key2)	Input	Входная строка
			Key1	Начальный ориентир
			Key2	Конечный ориентир
nullPressedButtons	Зануляет информацию о нажатых клавишах	private void nullPressedButtons ()	-	-
BufferedDrawField	Рисует Игровое поле	private void BufferedDrawField ()	-	-
UpdateBackGroung	Обновляет изображение заднего экрана, после изменения конфигурации поля	private void UpdateBackGroung()	-	-
BufferdDrawTankPlayer	Рисует в буфере танк игрока	private void BufferdDrawTankPlayer()	-	-
BufferdDrawTank	Рисует в буфере танк противника	private void BufferdDrawTank(Tank tank)	tank	Параметры танка для отрисовки

4.6 Описание модуля InitForm

Модуль InitForm представляет собой код окна инициализации клиента.

Таблица 4.6 - Методы модуля SnakeBodyPart

Имя метода	**Описание**	**Заголовок метода**	**Имя параметра**	**Назначение параметра**
InitForm	Возвращает экземпляр формы InitForm	public InitForm()		
buttonCancel_Click	Обрабатывает Отмена открытия клиентского окна	private void buttonCancel_Click(object sender, EventArgs e)	sender	Отправитель вызова
			e	Информация о событии
buttonSubmit_Click	Обрабатывает подтверждения открытия клиентского окна	private void buttonSubmit_Click(object sender, EventArgs e)	sender	Отправитель вызова
			e	Информация о событии

4.7 Описание модуля UdpClient

Модуль Food представляет собой еду, которую змеи могут съесть в игре.

Таблица 4.7 - Методы модуля UdpClient

Имя метода	**Описание**	**Заголовок метода**	**Имя параметра**	**Назначение параметра**
UdpClient	Возвращает экземпляр класса UdpClient	public UdpClient(string serverIpAddress, int serverPort)	serverIpAddress	IP адрес сервера
			serverPort	Порт сервера

Продолжение таблицы 4.7

SendAndReceiveAsync	Сначала отправляет сообщение, а потом ждет ответ от сервера	public async Task<string> SendAndReceiveAsync(string message)	message	Отправляемое сообщение
SendAsync	Отправляет сообщение серверу	public async Task SendAsync(string message)	message	Отправляемое сообщение
ReceiveAsync	Получает сообщение от сервера и возвращает его в виде строки	public async Task<string> ReceiveAsync()	-	-

4.8 Описание модуля GameClasses

Модуль GameClasses представляет собой библиотеку классов, необходимых для работы клиента и сервера, ниже будут приведены методы, которые присутствуют у классов этой библиотеки.

Таблица 4.8 - Методы модуля GameClasses

Имя метода	**Описание**	**Заголовок метода**	**Имя параметра**	**Назначение параметра**
GameFiled	Возвращает экзеспляр класса GameField с заданными размерами	public GameFiled(int Size, int Width, int Height)	Size	Размер в пикселях одной клетки поля
			Width	Ширина поля
			Height	Высота поля
Block	Возвращает экземпляр класса Block	public Block()	-	-

Продолжение таблицы 4.8

BackGround	Возвращает экзепляр класса BackGround с заданными размерами	public BackGround(int size)	size	Размер в пикселях одной клетки поля
UnbBrick	Возвращает экзепляр класса UnbBrick с заданными размерами	public UnbBrick(int size)	size	Размер в пикселях одной клетки поля
Brick	Возвращает экзепляр класса Brick с заданными размерами	public Brick(int size)	size	Размер в пикселях одной клетки поля
Tank	Возвращает экзепляр класса Tank с заданными размерами	public Tank(int size, string name)	size	Размер в пикселях одной клетки поля
			name	Имя танка
fire	Проверяет может ли выстрелить данный такнк	public bool fire()	-	-
timeivent	Служит для перезарядки обоемы танка со временем	public void timeivent()	-	-

Продолжение таблицы 4.8

TankPlayer	Возвращает экзепляр класса Tank с заданными размерами	public TankPlayer (int size, string name)	size	Размер в пикселях одной клетки поля
			name	Имя танка
fire	Проверяет может ли выстрелить данный TankPlayer	public bool fire()	-	-
timeivent	Служит для перезарядки обоемы TankPlayer со временем	public void timeivent()	-	-
bullet	Возвращает экзепляр класса bullet с заданными размерами	public bullet(int size, double Px, double Py, int Ppos)	Size	Размер в пикселях одной клетки поля
			Px	Положение по оси X
			Py	Положение по оси Y
			Ppos	Степень поворота
Add	Безопасно для потока добавляет новый элемент	public void Add(T item)	T	новый элемент

Продолжение таблицы 4.8

Remove	Удаляет элемент из списка, закрывая список к чтению на это время	public bool Remove(T item)	T	новый элемент
Clear	Очищает список, закрывая его к чтению на это время	public void Clear()	T	новый элемент
Count	Вохвращает количество жлементов в списке	public int Count	-	-

5. ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

В данном разделе я остановлюсь на тестировании программного средства. В таблицах 5.1, 5.2, 5.3 и 5.4 представлены результаты тестирования программного средства.

5.1 Тестирование и проверка работоспособности кнопок

Таблица 5.1 - Тестирование и проверка работоспособности кнопок

№	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	Установка IP сервера для игры вручную	Запустить приложение, ввести IP сервера игры вручную	Ввод IP сервера в соответствующее текстовое поле	Тест пройден
2	Установка желаемо ника танка для игры	Запустить приложение, ввести желаемы ник для змеи	Ввод ника в соответствующее текстовое поле	Тест пройден
3	Установка порта сервера для игры вручную	Запустить приложение, ввести порт сервера игры вручную	Ввод порта сервера в соответствующее текстовое поле	Тест пройден
4	Отмена запуска игры	Запустить приложение, нажать кнопку “отмена”	Полное закрытие клиента игры	Тест пройден
5	Подключение к оригинальному игровому серверу	Запустить приложение, нажать на кнопку «Подтвердить»	Подключение к оригинальному игровому серверу	Тест пройден

5.2 Тестирование управления игровым процессом

Таблица 5.2 - Тестирование управления игровым процессом

№	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	Движение танка вправо при нажатии кнопки “D”	Запустить приложение, нажать кнопку «Подтвердить», после подключения к серверу нажать на клавишу “D”	Танк начал двигаться вправо	Тест пройден
2	Движение танка вниз при нажатии кнопки “S”	Запустить приложение, нажать кнопку «Подтвердить», после подключения к серверу нажать на клавишу “S”	Танк начал двигаться вниз	Тест пройден
3	Движение танка влево при нажатии кнопки “A”	Запустить приложение, нажать кнопку «Подтвердить», после подключения к серверу нажать на клавишу “A”	Танк начал двигаться влево	Тест пройден
4	Движение танка вверх при нажатии кнопки “W”	Запустить приложение, нажать кнопку «Подтвердить», после подключения к серверу нажать на клавишу “W”	Танк начал двигаться вверх	Тест пройден

Продолжение таблицы 5.2

5	выстрел танка по направлению дула при нажатии кнопки “space”	Запустить приложение, нажать кнопку «Подтвердить», после подключения к серверу нажать на клавишу “space”	Появился снаряд, двигающийся по направлению дула танка в момент выстрела	Тест пройден

5.3 Тестирование игровой логики

Таблица 5.3 - Тестирование игровой логики

№	**Тестируемая функциональность**	**Последовательность действий**	**Ожидаемый результат**	**Полученный результат**
1	Появление своего танка на поле	Запустить приложение, нажать кнопку «Connect»	Появляется голубой танк	Тест пройден
2	Появление вражеского танка на поле	Запустить приложение, нажать кнопку «Connect»	Появляется серый танк	Тест пройден
3	Пропадание Пуль при попадании в край игрового поля	Запустить приложение, нажать кнопку «Connect»,выстрелить	Долетя до края поля, пуля пропадает	Тест пройден
4	Пропадание Пуль при попадании в неломаемый кирпич	Запустить приложение, нажать кнопку «Connect», выстрелить в белый кирпич	Долетя до белого кирпича, пуля пропадает	Тест пройден

Продолжение таблицы 5.3

5	Пропадание Пуль при попадании в ломаемый кирпич	Запустить приложение, нажать кнопку «Connect», выстрелить в оранжевый кирпич	Долетя до оранжевый кирпича, пуля пропадает	Тест пройден
6	При втором попадании пули в ломаемый кирпич он пропадает	Запустить приложение, нажать кнопку «Connect», выстрелить в оранжевый кирпич дважды	После прилета второй пули камень пропадает, а на его месте появляется асфальт	Тест пройден
7	При третьем попадании пули в танк, он пропадает погибает	Запустить приложение, нажать кнопку «Connect», выстрелить в вражеский танк трижды	После попадания 3й пули танк меняет свое местоположение на стандартное	Тест пройден
8	Окончание игры после трех убийств вражеского танка	Запустить приложение, нажать кнопку «Connect», уничтожить в вражеский танк трижды	После 3й смерти игра закончится и выведется сообщение о победе	Тест пройден

5.4 Итоги тестирования

Подводя итог, отмечу, что программа отвечает заданным функциональным требованиям, наблюдается стабильность в работе. Вопросов к эстетической части не имеется.

6. РУКОВОДСТВО ПО УСТАНОВКЕ И ИСОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

Для корректной работы программы необходимо распаковать архив в удобную вам папку, чтобы начать использовать программное средство, необходимо сначала запустить server.exe.

После открытия программы появляется окно с игровым полем (рис. 6.1)

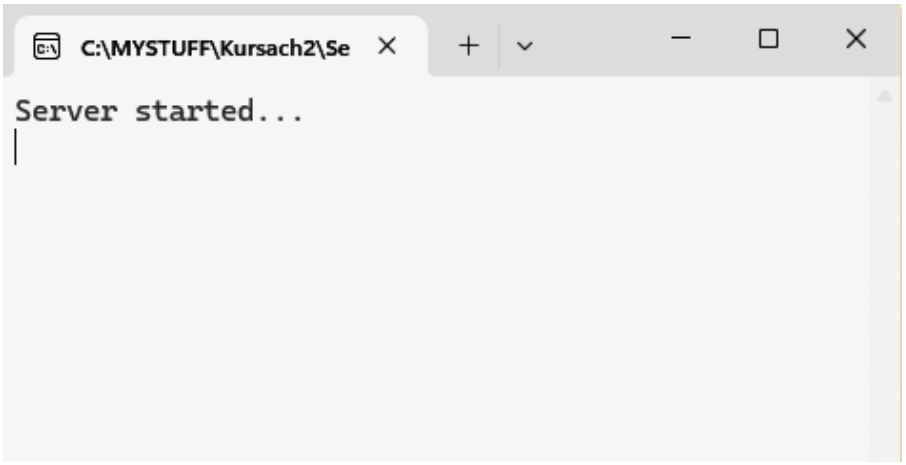

Рисунок 6.1 – Сервер

После этого необходимо запустить client.exe, где в появившиеся поля необходимо ввести IP адрес и порт сервера, а так же уникальное имя пользователя, после чего нажав кнопку “подтвердить”, вы подключитесь к серверу, когда на сервере будет 2 игрока она запустит игру, после чего появиться игровое поле.

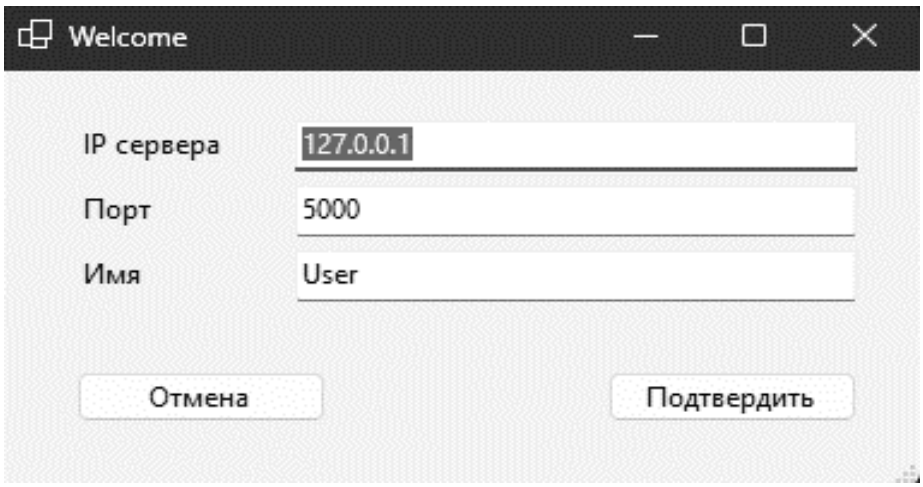

Рисунок 6.2 – Поле ввода информации о сервере

Теперь игрок может управлять танком с клавиатуры. Также можно выстрелить из пушки при нажатии пробела на клавиатуре. При попадании пули в коричневый кирпич дважды, он пропадет и на его месте откроется проход, при трех попаданиях пулей в другого игрока у того закончатся жизни, и он погибнет,

игра ведется до трех смертей, первый игрок, погибший три раза проигрывает, после этого сервер и клиенты сами отключаться друг от друга;

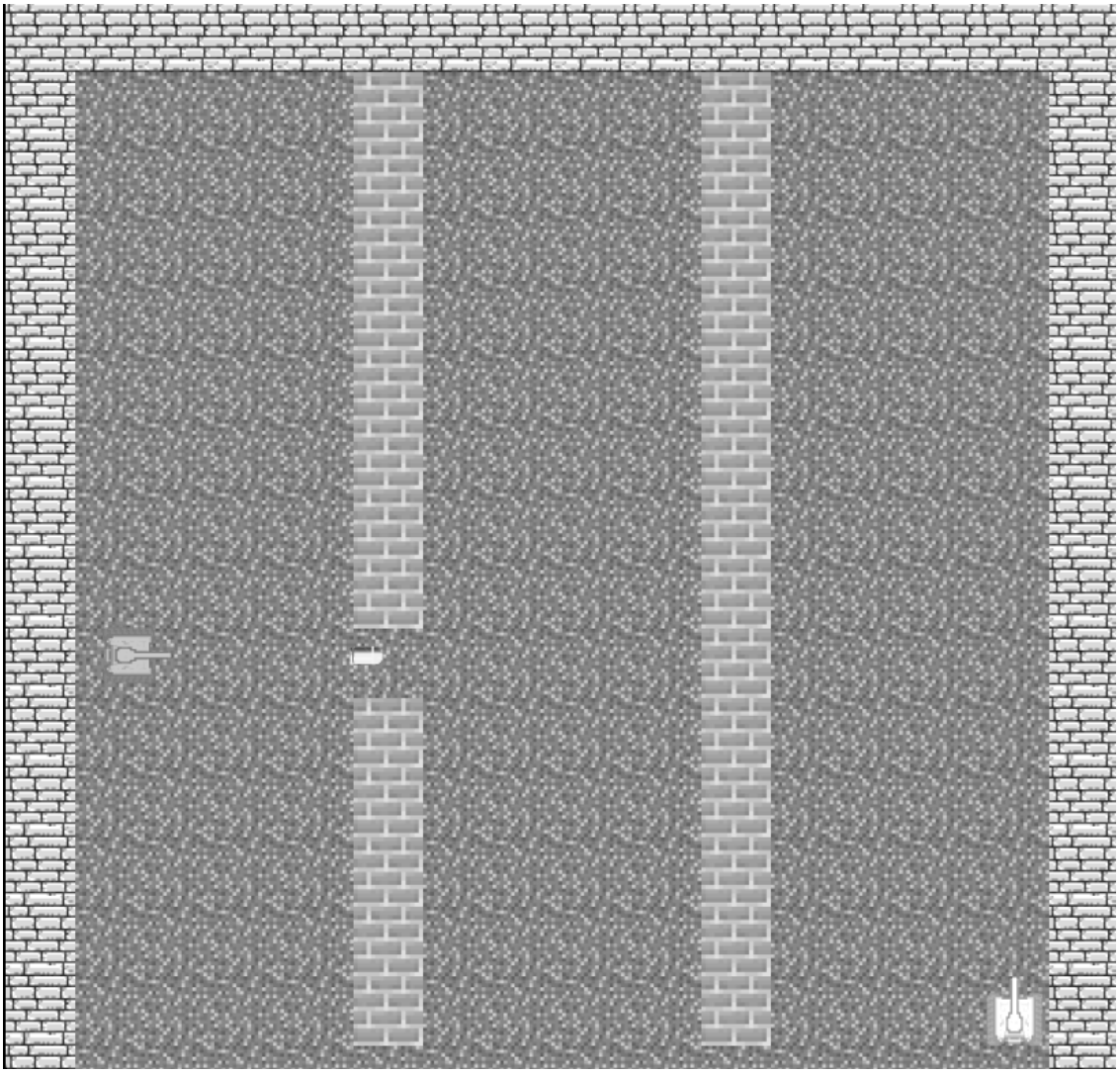

Рисунок 6.3 – Игровое поле

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта была проанализирована предметная область, рассмотрены существующие аналоги и выявлены их преимущества и недостатки. В качестве языка разработки использовался C#. На этапе проектирования были разработаны блок-схемы алгоритмов. В ходе работы был получен опыт работы со сторонними графическими библиотеками, в частности Windows.Forms..

В ходе данной работы был создан протокол, по которому общается сервер игры с клиентом, было создано программное средство «Танки онлайн», которое предоставляет собой клиент для этой игры. Так же был реализован сервер, поддерживающий одновременную игру двух человек.

Проведено тестирование работоспособности разработанной программной части. Поставленная цель была выполнена в полном объеме, работоспособность подтверждена тестированием программного средства.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

Список использованной литературы

[1] https://en.wikipedia.org/wiki/Battle_City

[2] https://bootleggames.fandom.com/wiki/Tank_1990

[3] https://en.wikipedia.org/wiki/Tank_Force

[4] https://learn.microsoft.com/dotnet/framework/network-programming/using-udp-services

Приложение А

Код класса UdpServer.cs

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;
using System.Runtime.Remoting.Messaging;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using GameClasses;

public class UdpServer
{
    private readonly IPEndPoint _endPoint;
    private readonly Socket _listener;
    //private readonly List<EndPoint> _clients;
    private Timer _timer;

    System.Threading.Timer TickTimer;
    const int FieldSize = 16;
    const int PlayersAmount = 2;

    List<Thread> playersListeners;

    ThreadSafeList<playerInfo> _players;
    ThreadSafeList<bullet> _bullets;
    ThreadSafeList<Block> GameFieldGhanges;
    public bool ChangeSaved;

    GameFiled gameFiled;
    public bool PlayerFire;
    static public bool MoveUpCheck(double x, double y, GameFiled gameFiled)
    {
        int x1, y1, x2, y2;
        x1 = (int)(x + 0.2);
        x2 = (int)(x - 0.2 + 1);
        y1 = (int)(y);
        if (x1 < gameFiled.arr.Length && x2 < gameFiled.arr.Length)
```

```
    {
        if (gameFiled.arr[x1, y1].passable && gameFiled.arr[x2, y1].passable)
            return true;
    }
    return false;
}
static public bool MoveDownCheck(double x, double y, GameFiled gameFiled)
{
    int x1, y1, x2, y2;
    x1 = (int)(x + 0.2);
    x2 = (int)(x - 0.2 + 1);
    y1 = (int)(y) + 1;
    int tmplen = (int)Math.Sqrt(gameFiled.arr.Length + 0.01);
    if (y1 >= tmplen)
        return false;
    if (x1 < gameFiled.arr.Length && x2 < gameFiled.arr.Length)
    {
        if (gameFiled.arr[x1, y1].passable && gameFiled.arr[x2, y1].passable)
            return true;
    }
    return false;
}
static public bool MoveRightCheck(double x, double y, GameFiled gameFiled)
{
    int x1, y1, x2, y2;
    y1 = (int)(y + 0.2);
    y2 = (int)(y - 0.2 + 1);
    x1 = (int)x + 1;
    int tmplen = (int)Math.Sqrt(gameFiled.arr.Length + 0.01);
    if (x1 >= tmplen)
        return false;
    if (y2 < gameFiled.arr.Length && y1 < gameFiled.arr.Length)
    {
        if (gameFiled.arr[x1, y1].passable && gameFiled.arr[x1, y2].passable)
            return true;
    }
    return false;
}
static public bool MoveLeftCheck(double x, double y, GameFiled gameFiled)
{
```

```
    int x1, y1, x2, y2;
    y1 = (int)(y + 0.2);
    y2 = (int)(y - 0.2 + 1);
    x1 = (int)x;
    if (y2 < gameFiled.arr.Length && y1 < gameFiled.arr.Length)
    {

        if (gameFiled.arr[x1, y1].passable && gameFiled.arr[x1, y2].passable)
            return true;
    }
    return false;
}
static void TimerTickCallback(Object state)
{

    UdpServer server = (UdpServer)state;
    //двигаем игроков
    foreach (var player in server._players)
    {
        if (player.WPress)
        {
            player.tank.y -= player.tank.speed;
            if (player.tank.y <= 0)
                player.tank.y = 0;
            else
            if (!MoveUpCheck(player.tank.x, player.tank.y, server.gameFiled))
                player.tank.y += player.tank.speed;
        }
        if (player.APress)
        {
            player.tank.x -= player.tank.speed;
            if (player.tank.x <= 0) player.tank.x = 0;
            else
            if (!MoveLeftCheck(player.tank.x, player.tank.y, server.gameFiled))
                player.tank.x += player.tank.speed;

        }
        if (player.SPress)
        {
```

```
            player.tank.y += player.tank.speed;
            if (player.tank.y >= FieldSize)
                player.tank.y = FieldSize;
            else
            if (!MoveDownCheck(player.tank.x, player.tank.y, server.gameFiled))
                player.tank.y -= player.tank.speed;
        }
        if (player.DPress)
        {
            player.tank.x += player.tank.speed;
            if (player.tank.x >= FieldSize)
                player.tank.x = FieldSize;
            else
                if (!MoveRightCheck(player.tank.x, player.tank.y, server.gameFiled))
                player.tank.x -= player.tank.speed;
        }
    }
    foreach (var bullet in server._bullets)
    {
        bool flag = true;
        //двигаєм пули
        switch (bullet.pos)
        {
            case 0:
                bullet.y -= bullet.speed;
                if (bullet.y < 0)
                {
                    server._bullets.Remove(bullet);
                    flag = false;
                }
                break;
            case 1:
                bullet.x += bullet.speed;
                if (bullet.x > FieldSize)
                {
                    server._bullets.Remove(bullet);
                    flag = false;
                }
                break;
            case 2:
```

```
            bullet.y += bullet.speed;
            if (bullet.y > FieldSize)
            {
                server._bullets.Remove(bullet);
                flag = false;
            }
            break;
        case 3:
            bullet.x -= bullet.speed;
            if (bullet.x < 0)
            {
                server._bullets.Remove(bullet);
                flag = false;
            }
            break;
    }

    // проверяем попадания
    if (flag)
    {
        bool flag2 = false;
        foreach (var player in server._players)
        {
            switch (bullet.pos)
            {
                case 0:
                    if (((int)(bullet.y - 1) == (int)(player.tank.y)) && (((int)bullet.x) == (int)player.tank.x))
                    {
                        player.tank.hp--;
                        if (player.tank.hp == 0)
                        {
                            player.deaths++;
                            player.tank.x = player.defaultX;
                            player.tank.y = player.defaultY;
                            player.tank.hp = 3;
                            if(player.deaths == 3)
                            {
                                server.EndGame();
                            }
                        }
```

```
            flag2 = true;
        }
        break;
    case 1:
        if (((int)bullet.y == (int)(player.tank.y)) && (((int)bullet.x + 1) == (int)player.tank.x))
        {
            player.tank.hp--;
            if (player.tank.hp == 0)
            {
                player.deaths++;
                player.tank.x = player.defaultX;
                player.tank.y = player.defaultY;
                player.tank.hp = 3;
                if (player.deaths == 3)
                {
                    server.EndGame();
                }
            }
            flag2 = true;
        }
        break;
    case 2:
        if (((int)(bullet.y + 1) == (int)(player.tank.y)) && (((int)bullet.x) == (int)player.tank.x))
        {
            player.tank.hp--;
            if (player.tank.hp == 0)
            {
                player.deaths++;
                player.tank.x = player.defaultX;
                player.tank.y = player.defaultY;
                player.tank.hp = 3;
                if (player.deaths == 3)
                {
                    server.EndGame();
                }
            }
            flag2 = true;
        }
        break;
    case 3:
```

```
            if (((int)bullet.y == (int)(player.tank.y)) && (((int)bullet.x - 1) == (int)player.tank.x))
            {
                player.tank.hp--;
                if (player.tank.hp == 0)
                {
                    player.deaths++;
                    player.tank.x = player.defaultX;
                    player.tank.y = player.defaultY;
                    player.tank.hp = 3;
                    if (player.deaths == 3)
                    {
                        server.EndGame();
                    }
                }
                flag2 = true;
            }
            break;
        }
    }
    int x, y;
    int FieldLen = server.gameFiled.arr.Length;
    FieldLen = (int)Math.Sqrt(FieldLen + 0.001);
    switch (bullet.pos)
    {
        case 0:
            x = (int)(bullet.x + 0.5);
            y = (int)(bullet.y - 1);
            if (x < FieldLen && y > 0)
            {
                if (server.gameFiled.arr[x, y] is Brick)
                {
                    Brick brick = server.gameFiled.arr[x, y] as Brick;
                    brick.hp--;
                    if (brick.hp == 0)
                    {
                        server.gameFiled.arr[x, y] = new BackGround(1);
                        BackGround tmp = new BackGround(1);
                        tmp.x = (int)x;
                        tmp.y = (int)y;
                        server.GameFieldGhanges.Add(tmp);
```

```
                server.ChangeSaved = false;
            }
            flag2 = true;
        }
    }
    break;
case 1:
    x = (int)(bullet.x + 1);
    y = (int)(bullet.y + 0.5);
    if (x < FieldLen && y < FieldLen)
    {
        if (server.gameFiled.arr[x, y] is Brick)
        {
            Brick brick = server.gameFiled.arr[x, y] as Brick;
            brick.hp--;
            if (brick.hp == 0)
            {
                server.gameFiled.arr[x, y] = new BackGround(1);
                BackGround tmp = new BackGround(1);
                tmp.x = (int)x;
                tmp.y = (int)y;
                server.GameFieldGhanges.Add(tmp);
                server.ChangeSaved = false;
            }
            flag2 = true;
        }
    }
    break;
case 2:
    x = (int)(bullet.x + 0.5);
    y = (int)(bullet.y + 1);
    if (x < FieldLen && y < FieldLen)
    {
        if (server.gameFiled.arr[x, y] is Brick)
        {
            Brick brick = server.gameFiled.arr[x, y] as Brick;
            brick.hp--;
            if (brick.hp == 0)
            {
                server.gameFiled.arr[x, y] = new BackGround(1);
```

```
                BackGround tmp = new BackGround(1);
                tmp.x = (int)x;
                tmp.y = (int)y;
                server.GameFieldGhanges.Add(tmp);
                server.ChangeSaved = false;
            }
            flag2 = true;
        }
    }
    break;
  case 3:
    x = (int)(bullet.x - 1);
    y = (int)(bullet.y + 0.5);
    if (x > 0 && y < FieldLen)
    {
      if (server.gameFiled.arr[x, y] is Brick)
      {
        Brick brick = server.gameFiled.arr[x, y] as Brick;
        brick.hp--;
        if (brick.hp == 0)
        {
          server.gameFiled.arr[x, y] = new BackGround(1);
          BackGround tmp = new BackGround(1);
          tmp.x = (int)x;
          tmp.y = (int)y;
          server.GameFieldGhanges.Add(tmp);
          server.ChangeSaved = false;
        }
        flag2 = true;
      }
    }
    break;
}

/*  if (server.gameFiled.arr[(int)bullet.x, (int)bullet.y] is UnbBrick)
  {
    flag2 = true;
  }*/
if (flag2)
  server._bullets.Remove(bullet);
```

```
                }
                else
                {
                    server._bullets.Remove(bullet);
                }
            }



        }
        private void StartGame()
        {
            string message = "GAMESTART" +_players[0].userName + "|" + _players[0].tank.x.ToString()
                + "|" + _players[0].tank.y.ToString() + "|" + _players[1].userName + "|"
                + _players[1].tank.x.ToString() + "|" + _players[1].tank.y.ToString() + "ENOOFMESSAGE";
            byte[] data = Encoding.UTF8.GetBytes(message);
            foreach (var client in _players)
            {
                Task.Run(async () =>
                {
                    try
                    {
                        await _listener.SendToAsync(new ArraySegment<byte>(data), SocketFlags.None,
client.clientEndPoint);


                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"Error occurred while sending periodic message to {client}:
{ex.Message}");
                        _players.Remove(client); // Remove client if sending fails
                    }
                });
            }


            //Thread.Sleep(100);


            _timer = new Timer(SendPeriodicMessages, null, TimeSpan.FromMilliseconds(10),
TimeSpan.FromMilliseconds(10)); ;


            // Создаем и запускаем таймер
            int dueTime = 0;  // Время задержки перед первым запуском
```

```
            int period = 15;  // Период между запусками
            TickTimer = new System.Threading.Timer(TimerTickCallback, this, dueTime, period);
            playersListeners = new List<Thread>();
        }

        public void EndGame()
        {
            string message = "GAMEEND" + _players[0].userName + "|" + _players[0].deaths + "|" +
_players[1].userName + "|"
              + _players[1].deaths + "ENOOFMESSAGE";
            byte[] data = Encoding.UTF8.GetBytes(message);
            foreach (var client in _players)
            {
                Task.Run(async () =>
                {
                    try
                    {
                        await _listener.SendToAsync(new ArraySegment<byte>(data), SocketFlags.None,
client.clientEndPoint);

                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"Error occurred while sending periodic message to {client}:
{ex.Message}");
                        _players.Remove(client); // Remove client if sending fails
                    }
                });
            }
            Thread.Sleep(1000);

        }
        public UdpServer(string ipAddress, int port)
        {
            _endPoint = new IPEndPoint(IPAddress.Parse(ipAddress), port);
            _listener = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
            _listener.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReuseAddress, true);



            _players = new ThreadSafeList<playerInfo>();
```

```
            _bullets = new ThreadSafeList<bullet>();
            GameFieldGhanges = new ThreadSafeList<Block>();
            ChangeSaved = true;
            gameFiled = new GameFiled(1, FieldSize, FieldSize);




        }

        public async Task StartAsync()
        {
            _listener.Bind(_endPoint);
            Console.WriteLine("Server started...");

            while (true)
            {
                try
                {
                    await HandleIncomingMessagesAsync();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Error occurred: {ex.Message}");
                }
            }
        }

        private async Task HandleIncomingMessagesAsync()
        {
            byte[] buffer = new byte[1024];
            EndPoint clientEndPoint = new IPEndPoint(IPAddress.Any, 0);

            var result = await _listener.ReceiveFromAsync(new ArraySegment<byte>(buffer), SocketFlags.None,
clientEndPoint);
            clientEndPoint = result.RemoteEndPoint;

            string message = Encoding.UTF8.GetString(buffer, 6, result.ReceivedBytes);
            // Console.WriteLine($"Received from {clientEndPoint}: {message}");

            // Add player to the list if it's not already present
```

```
var flag = true;
for (int i = 0; i < _players.Count; i++)
{
    if (_players[i].clientEndPoint.Equals(clientEndPoint))
    {
        flag = false;
    }
}
if (flag)//новый игрок
{
    string Username = GetSubStringInd(message, "Hello, Server!", "**");
    if (_players.Count == 0)
    {
        _players.Add(new playerInfo(Username, new Tank(1, Username), clientEndPoint));
        _players[0].tank.x = 1;
        _players[0].tank.y = 1;
        _players[0].defaultX = 1;
        _players[0].defaultY = 1;
    }
    else
    if (_players.Count == 1)
    {
        _players.Add(new playerInfo(Username, new Tank(1, Username), clientEndPoint));
        _players[1].tank.x = 14;
        _players[1].tank.y = 14;
        _players[1].defaultX = 14;
        _players[1].defaultY = 14;

        StartGame();
    }
}
else// старый игрок
{
    string username = GetSubStringInd(message, "UserName", "<");
    bool WPESS = Convert.ToBoolean(GetSubStringInd(message, "<", "@"));
    bool APESS = Convert.ToBoolean(GetSubStringInd(message, "@", "#"));
    bool SPESS = Convert.ToBoolean(GetSubStringInd(message, "#", "$"));
    bool DPESS = Convert.ToBoolean(GetSubStringInd(message, "$", "%"));
    int POS = Convert.ToInt32(GetSubStringInd(message, "%", "^"));
    bool FIRE = Convert.ToBoolean(GetSubStringInd(message, "^", "&"));
```

```
    foreach (var user in _players)
    {
        if (user.userName == username)
        {
            user.WPress = WPESS;
            user.APress = APESS;
            user.SPress = SPESS;
            user.DPress = DPESS;
            user.tank.pos = POS;
            if (FIRE)
            {
                Console.WriteLine("FIRE");
                switch (POS)
                {
                    case 0:
                        _bullets.Add(new bullet(10, user.tank.x, user.tank.y - 1.1, user.tank.pos));
                        break;
                    case 1:
                        _bullets.Add(new bullet(10, user.tank.x + 1.1, user.tank.y, user.tank.pos));
                        break;
                    case 2:
                        _bullets.Add(new bullet(10, user.tank.x, user.tank.y + 1.1, user.tank.pos));
                        break;
                    case 3:
                        _bullets.Add(new bullet(10, user.tank.x - 1.1, user.tank.y, user.tank.pos));
                        break;

                }
            }
        }
    }
}

// Echo the message back to the client in a separate task
Task.Run(async () =>
{
    try
    {
```

```
                    await _listener.SendToAsync(new ArraySegment<byte>(buffer, 0, result.ReceivedBytes),
SocketFlags.None, clientEndPoint);
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Error occurred while sending message to {clientEndPoint}: {ex.Message}");
                    //      _clients.Remove(clientEndPoint); // Remove client if sending fails
                }
            });
        }



        private void SendPeriodicMessages(object state)
        {

            string message = "FieldChanges:";
            foreach (BackGround backGround in GameFieldGhanges)
            {
                message += "BACK" + backGround.x.ToString() + "|" + backGround.y.ToString() + "#$";
                GameFieldGhanges.Remove(backGround);
            }
            message += "Tanks:";
            ThreadSafeList<playerInfo> players = new ThreadSafeList<playerInfo>();
            players = _players;
            foreach (var player in players)
            {
                message += "Tank" + player.userName + "|" + player.tank.x.ToString() + "#" +
player.tank.y.ToString() + "%" + player.tank.pos.ToString() + "@$";
            }
            message += "Bullets:";
            ThreadSafeList<bullet> bullets= new ThreadSafeList<bullet>();
            bullets = _bullets;
            foreach (var bullet in bullets)
            {
                message += "Bullet" + bullet.x.ToString() + "|" + bullet.y.ToString() + "@" + bullet.pos.ToString() +
"#$";
            }

            message += "**";
            //Console.WriteLine(message);
            byte[] data = Encoding.UTF8.GetBytes(message);
```

```
                foreach (var client in _players)
                {
                    Task.Run(async () =>
                    {
                        try
                        {
                            await _listener.SendToAsync(new ArraySegment<byte>(data), SocketFlags.None,
client.clientEndPoint);

                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine($"Error occurred while sending periodic message to {client}:
{ex.Message}");
                            _players.Remove(client); // Remove client if sending fails
                        }
                    });
                }
            }

            public class playerInfo
            {
                public double defaultX, defaultY;
                public int deaths;
                public string userName;
                public Tank tank;
                public EndPoint clientEndPoint;

                public bool WPress, APress, SPress, DPress;
                public playerInfo(string username, Tank tank, EndPoint clientEndPoint)
                {
                    this.userName = username;
                    this.tank = tank;
                    this.clientEndPoint = clientEndPoint;

                    WPress = false;
                    APress = false;
                    SPress = false;
                    DPress = false;
                    deaths = 0;
```

```
            }
        }
        static string GetSubStringInd(string input, string key1, string key2)
        {
            return (GetSubstring(input, input.IndexOf(key1) + key1.Length, input.IndexOf(key2) -
input.IndexOf(key1) - key1.Length));
        }
        static string GetSubstring(string input, int startIndex, int length)
        {
            if (startIndex < 0 || startIndex >= input.Length)
            {
                throw new ArgumentOutOfRangeException(nameof(startIndex), "Start index is out of range.");
            }

            if (length < 0 || startIndex + length > input.Length)
            {
                throw new ArgumentOutOfRangeException(nameof(length), "Length is out of range.");
            }

            return input.Substring(startIndex, length);
        }
}
```

Код класса UdpClient.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Sockets;
using System.Net;
using System.Text;
using System.Threading.Tasks;

namespace Client
{
    internal class UdpClient
    {
        private readonly IPEndPoint _serverEndPoint;
        private readonly Socket _clientSocket;
        private readonly EndPoint serverEndPoint;
        public UdpClient(string serverIpAddress, int serverPort)
        {
            _serverEndPoint = new IPEndPoint(IPAddress.Parse(serverIpAddress), serverPort);
            _clientSocket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
        }

        public async Task<string> SendAndReceiveAsync(string message)
        {
            byte[] data = Encoding.UTF8.GetBytes("client" + message);

            await _clientSocket.SendToAsync(new ArraySegment<byte>(data), SocketFlags.None, _serverEndPoint);
```

```
            Console.WriteLine("Sent: " + message);

            byte[] buffer = new byte[1024];
            EndPoint serverEndPoint = new IPEndPoint(IPAddress.Any, 0);

            var result = await _clientSocket.ReceiveFromAsync(new ArraySegment<byte>(buffer), SocketFlags.None,
serverEndPoint);
            string response = Encoding.UTF8.GetString(buffer, 0, result.ReceivedBytes);
            return response;
        }
        public async Task SendAsync(string message)
        {
            byte[] data = Encoding.UTF8.GetBytes("client" + message);

            await _clientSocket.SendToAsync(new ArraySegment<byte>(data), SocketFlags.None, _serverEndPoint);
            Console.WriteLine("Sent: " + message);
        }
        public async Task<string> ReceiveAsync()
        {
            byte[] buffer = new byte[1024];
            EndPoint serverEndPoint = new IPEndPoint(IPAddress.Any, 0);

            var result = await _clientSocket.ReceiveFromAsync(new ArraySegment<byte>(buffer), SocketFlags.None,
serverEndPoint);
            string response = Encoding.UTF8.GetString(buffer, 0, result.ReceivedBytes);
            if (response.StartsWith("client"))
                return "";

            return response;
        }

    }
}
```

Код класса MainForm.cs

```
using GameClasses;
using System.Diagnostics.Metrics;
using System.Net.Sockets;
using System.Threading.Tasks;
using System.Timers;
using System.Windows.Forms;
using System.Xml.Serialization;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.Taskbar;
namespace Client
{
    public partial class MainForm : Form
    {
        Bitmap BackgroundBMP;
        Graphics g, BufferG, BackG;
        GameFiled gameFiled;
        private int Port;
        private string Username;
        private string IP;
        UdpClient client;
        Thread ServerListener;
        Thread ServerSender;
        Thread DrawThread;
        private Bitmap buffer;
        int screenWidth, screenHeight;
        const int FieldSize = 16;
```

```
int BlockPixelAmount;
ThreadSafeList<Tank> Enemies;
ThreadSafeList<bullet> Bullets;
bool playerFireFlag;
TankPlayer Player;

public MainForm()
{
   Screen screen = Screen.PrimaryScreen;
   screenWidth = screen.Bounds.Width;
   screenHeight = screen.Bounds.Height;

   InitForm initForm = new InitForm();
   try
   {

      initForm.ShowDialog();
      Port = initForm.port;
      IP = initForm.ip;
      Username = initForm.Username;
      if (initForm.quit)
         Close();
   }
   catch
   {
      MessageBox.Show("что-то не так");
   }
   InitializeComponent();
   this.Width = (int)screenWidth + 50;
   this.Height = (int)screenHeight + 50;

   pictureBox.Width = screenWidth;
   pictureBox.Height = screenHeight;

   BlockPixelAmount = (int)screenHeight / (FieldSize + 5);
   gameFiled = new GameFiled(BlockPixelAmount, FieldSize, FieldSize);
   g = pictureBox.CreateGraphics();
   BackgroundBMP = new Bitmap(screenWidth,screenHeight);
   BackG = Graphics.FromImage(BackgroundBMP);
   buffer = new Bitmap(this.Width, this.Height);
   BufferG = Graphics.FromImage(buffer);

   Enemies = new ThreadSafeList<Tank>();
   Bullets = new ThreadSafeList<bullet> ();
   Player = new TankPlayer(BlockPixelAmount, initForm.Username);
   Player.x = 1; Player.y = 1;


   // Создаем и запускаем таймер
   int dueTime = 0;  // Время задержки перед первым запуском
   int period = 10;  // Период между запусками
   //ServerTimer = new System.Threading.Timer(TimerMoveCallback, this, dueTime, period);
}

private async void MainForm_Load(object sender, EventArgs e)
{

   try
   {
      client = new UdpClient(IP, Port);
```

```
        string message = "Hello, Server!" + Username + "**";
        message = await client.SendAndReceiveAsync(message);

        string gamestart = await client.ReceiveAsync();
        Thread.Sleep(1000);
        gamestart = GetSubStringInd(gamestart, "GAMESTART", "ENOOFMESSAGE");
        string[] paraments = gamestart.Split('|');
        if (paraments[0] == Username)
        {
            Player.x = Convert.ToInt32(paraments[1]);
            Player.y = Convert.ToInt32(paraments[2]);
            Enemies.Add(new Tank(BlockPixelAmount, paraments[3]));
            Enemies[0].x = Convert.ToInt32(paraments[4]);
            Enemies[0].y = Convert.ToInt32(paraments[5]);
        }
        else
        {
            Player.x = Convert.ToInt32(paraments[4]);
            Player.y = Convert.ToInt32(paraments[5]);
            Enemies.Add(new Tank(BlockPixelAmount, paraments[0]));
            Enemies[0].x = Convert.ToInt32(paraments[1]);
            Enemies[0].y = Convert.ToInt32(paraments[2]);
        }
        ServerListener = new Thread(new ThreadStart(ReceiveMessageTread));
        ServerListener.Start();

        ServerSender = new Thread(new ThreadStart(SendToServerThread));
        ServerSender.Start();

        DrawThread = new Thread(new ThreadStart(DrawThreadTask));
        DrawThread.Start();

        UpdateBackGroung();
    }
    catch (Exception ex)
    {
        //MessageBox.Show($"Error: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void UpdateBackGroung()
{
    BackG.FillRectangle(Brushes.Black, 0, 0, screenWidth, screenHeight);
    for (int i = 0; i < FieldSize; i++)
    {
        for (int j = 0; j < FieldSize; j++)
        {
            BackG.DrawImage(gameFiled.arr[i, j].Texture, i * (BlockPixelAmount), j * (BlockPixelAmount));
        }
    }
}
private void BufferedDrawField()
{
    //BufferG.Clear(Color.Black);
    BufferG.DrawImage(BackgroundBMP, 0, 0);
    BufferdDrawTankPlayer();
    BufferdDrawTank(Enemies[0]);

    for (int i = 0;i< Bullets.Count; i++)
    {
```

```
                BufferG.DrawImage(Bullets[i].Texture[Bullets[i].pos], (int)(Bullets[i].x * (BlockPixelAmount)),
(int)(Bullets[i].y * (BlockPixelAmount)));
            }
            g.DrawImage(buffer, 0, 0);
        }

        private async void DrawThreadTask()
        {
            while (true)
            {
                BufferedDrawField();
                Thread.Sleep(20);
            }
        }
        private async void ReceiveMessageTread()
        {
            while (true)
            {
                string message = await client.ReceiveAsync();
                if (message != "")
                    if(message.IndexOf("GAMEEND") < 0)
                    {
                        string FieldChange = GetSubStringInd(message, "FieldChanges:", "Tanks:");
                        string[] fldchanges = FieldChange.Split('$');
                        for (int i = 0; i < fldchanges.Length - 1; i++)
                        {
                            int x = Convert.ToInt32(GetSubStringInd(fldchanges[i], "BACK", "|"));
                            int y = Convert.ToInt32(GetSubStringInd(fldchanges[i], "|", "#"));
                            gameFiled.arr[x, y] = new BackGround(BlockPixelAmount);
                        }
                        if(fldchanges.Length - 1 > 0)
                        {
                            UpdateBackGroung();
                        }
                        string TanksPos = GetSubStringInd(message, "Tanks:", "Bullets:");
                        string[] tanks = TanksPos.Split('$');
                        for (int i = 0; i < tanks.Length - 1; i++)
                        {
                            string username = GetSubStringInd(tanks[i], "Tank", "|");
                            double x = Convert.ToDouble(GetSubStringInd(tanks[i], "|", "#"));
                            double y = Convert.ToDouble(GetSubStringInd(tanks[i], "#", "%"));
                            int pos = Convert.ToInt32(GetSubStringInd(tanks[i], "%", "@"));
                            if (username == Player.name)
                            {
                                Player.x = x; Player.y = y;
                                //Player.pos = pos;
                            }
                            else
                            {
                                foreach (var tank in Enemies)
                                {
                                    if (tank.name == username)
                                    {
                                        tank.x = x; tank.y = y;
                                        tank.pos = pos;
                                    }
                                }
                            }
                        }
```

```
                string BulletsPos = GetSubStringInd(message, "Bullets:", "**");
                if (BulletsPos != "")
                {
                    int t = 0;
                }
                string[] BulletsStr = BulletsPos.Split('$');
                //Bullets.Clear();
                ThreadSafeList<bullet> bulletstmp = new ThreadSafeList<bullet>();
                if (BulletsStr[0] != "")
                    for (int i = 0; i < BulletsStr.Length-1; i++)
                    {
                        double x = Convert.ToDouble(GetSubStringInd(BulletsStr[i], "Bullet", "|"));
                        double y = Convert.ToDouble(GetSubStringInd(BulletsStr[i], "|", "@"));
                        int pos = Convert.ToInt32(GetSubStringInd(BulletsStr[i], "@", "#"));
                        bullet tmp = new bullet(BlockPixelAmount, 1, 1, pos);
                        tmp.x = x; tmp.y = y;

                        bulletstmp.Add(tmp);
                    }
                Bullets = bulletstmp;
            }
            else
            {
                string[] info = GetSubStringInd(message, "GAMEEND", "ENOOFMESSAGE").Split('|');
                if (info[0] == Username)
                {
                    if (Convert.ToInt32(info[1]) == 3)
                        MessageBox.Show("Победа!");
                    else
                        MessageBox.Show("Поражение!");
                }
                else
                {
                    if (Convert.ToInt32(info[1]) == 3)
                        MessageBox.Show("Поражение!");
                    else
                        MessageBox.Show("Победа!");

                }
                this.Close();
            }

        }
    }
    private async void SendToServerThread()
    {
        while (true)
        {

            string message = "UserName" + Player.name + "<" + Player.WPress.ToString() + "@" +
Player.APress.ToString() + "#" + Player.SPress.ToString() + "$" + Player.DPress.ToString() + "%" +
Player.pos.ToString() + "^" + playerFireFlag.ToString() + "&(";
            if (playerFireFlag == true)
            {
                playerFireFlag = false;
            };
            await client.SendAsync(message);
            Thread.Sleep(50);
        }
```

```
        }
        private void BufferdDrawTankPlayer()
        {
            BufferG.DrawImage(Player.Texture[Player.pos], (int)(Player.x * BlockPixelAmount), (int)(Player.y *
BlockPixelAmount));
        }
        private void BufferdDrawTank(Tank tank)
        {
            BufferG.DrawImage(tank.Texture[tank.pos], (int)(tank.x * BlockPixelAmount), (int)(tank.y *
BlockPixelAmount));
        }
        private void pictureBox_Click(object sender, EventArgs e)
        {
            // BufferedDrawField();
        }


        private void nullPressedButtons()
        {
            Player.WPress = false;
            Player.APress = false;
            Player.SPress = false;
            Player.DPress = false;
        }
        private void MainForm_KeyDown(object sender, KeyEventArgs e)
        {
            switch (e.KeyCode)
            {
                case Keys.Escape:
                    Close();
                    break;
                case Keys.W:
                    nullPressedButtons();
                    Player.WPress = true;
                    Player.pos = 0;
                    break;
                case Keys.A:
                    nullPressedButtons();
                    Player.APress = true;
                    Player.pos = 3;
                    break;
                case Keys.S:
                    nullPressedButtons();
                    Player.SPress = true;
                    Player.pos = 2;
                    break;
                case Keys.D:
                    nullPressedButtons();
                    Player.DPress = true;
                    Player.pos = 1;
                    break;
                case Keys.Space:
                    playerFireFlag = true;
                    break;
            }
        }
        private void MainForm_KeyUp(object sender, KeyEventArgs e)
        {
            switch (e.KeyCode)
            {
```

```
            case Keys.W:
                Player.WPress = false;
                break;
            case Keys.A:
                Player.APress = false;
                break;
            case Keys.S:
                Player.SPress = false;
                break;
            case Keys.D:
                Player.DPress = false;
                break;
        }
    }
    static string GetSubStringInd(string input, string key1, string key2)
    {
        return (GetSubstring(input, input.IndexOf(key1) + key1.Length, input.IndexOf(key2) - input.IndexOf(key1) -
key1.Length));
    }
    static string GetSubstring(string input, int startIndex, int length)
    {
        if (startIndex < 0 || startIndex >= input.Length)
        {
            throw new ArgumentOutOfRangeException(nameof(startIndex), "Start index is out of range.");
        }

        if (length < 0 || startIndex + length > input.Length)
        {
            throw new ArgumentOutOfRangeException(nameof(length), "Length is out of range.");
        }

        return input.Substring(startIndex, length);
    }


  }
}
```