­Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Программное средство «Игровое приложение танки»

БГУИР КП 1-40 01 01 014 ПЗ

Студент: гр. 151004 Дубовский А.В.

Руководитель: Красковский П.Н

Минск 2022

содержание

[Введение 3](#_Toc136659721)

[1 Аналитический обзор литературы 4](#_Toc136659722)

[1.1 Обзор существующих аналогов 4](#_Toc136659723)

[1.2 Перечень функциональных требований 6](#_Toc136659724)

[1.3 Состав и параметры технических и программных средств 6](#_Toc136659725)

[2 Моделирование предметной области 7](#_Toc136659726)

[2.1 Протокол UDP 7](#_Toc136659727)

[3 Проектирование программного средства 8](#_Toc136659728)

[3.1 Структура программы 8](#_Toc136659729)

[3.2 Проектирование интерфейса программного средства 8](#_Toc136659730)

[3.3 Проектирование функционала программного средства 10](#_Toc136659731)

[4 Разработка программного средства 14](#_Toc136659732)

[4.1 Проверка столкновения танка 14](#_Toc136659733)

[4.2 Движение танка 15](#_Toc136659734)

[4.3 Создание снаряда 16](#_Toc136659735)

[5 Тестирование программного средства 17](#_Toc136659736)

[5.1 Запуск, подключение, соединение 17](#_Toc136659737)

[5.2 Игровой процесс 17](#_Toc136659738)

[5.3 Конец игры 18](#_Toc136659739)

[6 Руководство пользователя 19](#_Toc136659740)

[6.1 Минимальные системные требования 19](#_Toc136659741)

[6.2 Установка 19](#_Toc136659742)

[6.3 Главное меню 19](#_Toc136659743)

[6.4 Меню сервера 20](#_Toc136659744)

[6.5 Меню клиента 20](#_Toc136659745)

[6.6 Игровая зона 21](#_Toc136659746)

[Заключение 22](#_Toc136659747)

[Список использованной литературы 23](#_Toc136659748)

[Приложение А 24](#_Toc136659749)

# Введение

Данный курсовой проект посвящён обновлению легендарной игры «Танчики», для преобщения молодёжи к класике.

Первая версия «Танчиков» выпущина в 1980г компание «Namco».

В основе игрового процесса «Танчиков» лежит перемещение по карте с последующим отстрелом врагов.

В 1985г компания «Namco» выпустила обновлённую версию танчиков для консоли «Famicon».

В 2004г компаания «Бука» выпустила 3D версию «Танчиков».

# Аналитический обзор литературы

## Обзор существующих аналогов

### Игророве приложение «Танчики Денди 1990»

Программное средство, изображённое на рисунке 1.1, является наиболее известной итерацией игрового приложения танчики. Ввиду это являеться основным конкурентом разрабатываемого приложения.

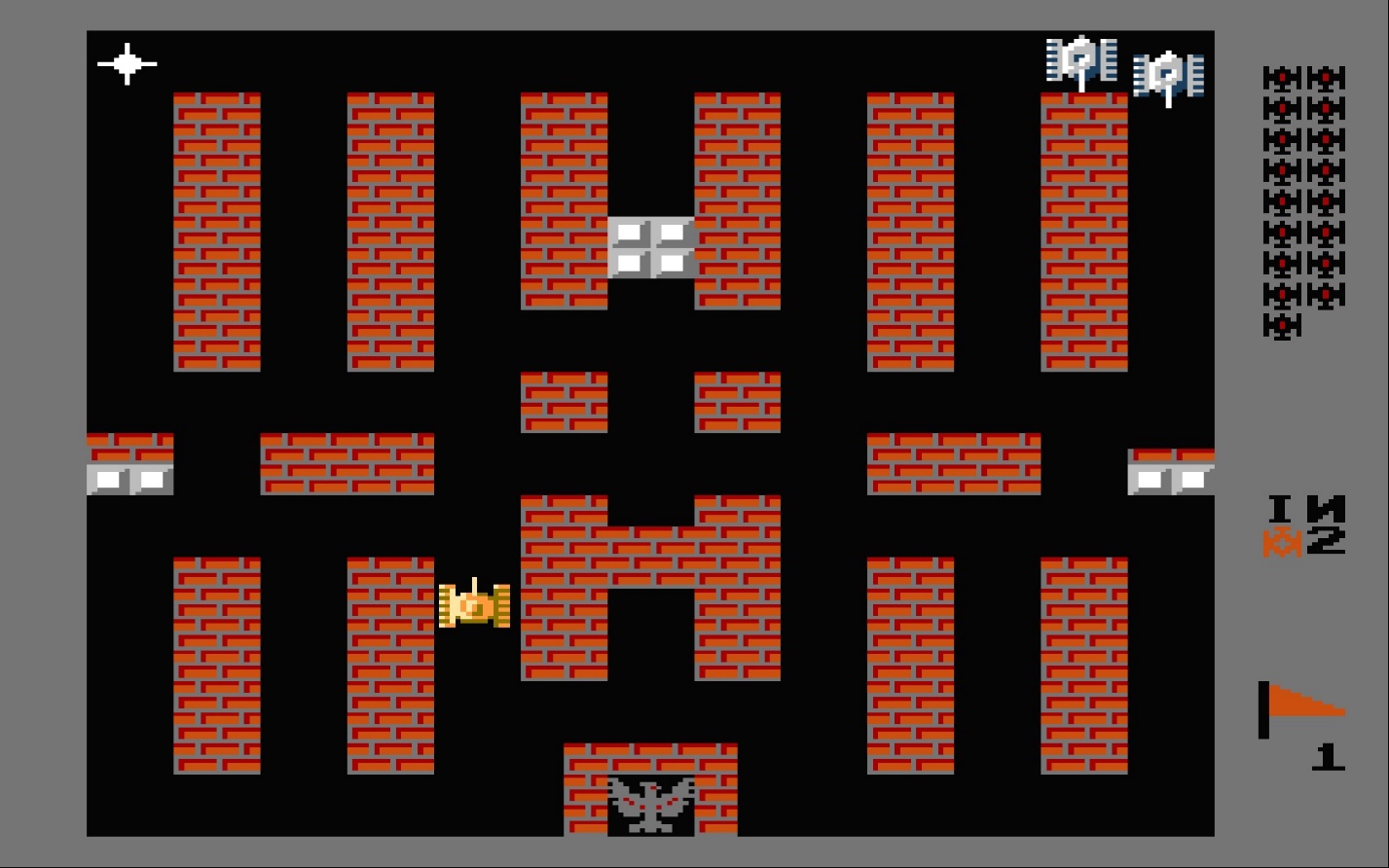


Рисунок 1.1 – Игророве приложение «Танчики Денди 1990»

Список выполняемых функций:

* управление танком;
* сражение с ботами.

Рассмотрим достоинства данного приложения:

* наличие 14 уровней;
* разнообразие врагов и препядствий;
* разруемость препядствий;

Рассмотрим недостатки данного приложения:

* запуск только на платформе Денди.

### Игровое приложение «Реалистичный симулятр танковых сражений»

Программное средство, изображённое на рисунке 1.2, сохраняет привычный гемплей танчиков, превнося в него 3D.



Рисунок 1.2 – Игровое приложение «Реалистический симулятр танковых сражений»

Список выполняемых функций:

* управление танком;
* сражение с ботами;
* Растановка танков на карте.

Рассмотрим достоинства данного приложения:

* 3D графика;
* использование браузера, что позволяет запускать игровое приложение

на любом устройстве.

Рассмотрим недостатки данного приложения:

* невозможность играть при отсутствии подключения к интернету;
* низкая производительность.

## Перечень функциональных требований

После анализа аналогов был составлен следующий список выполняемых функций:

* управление танком;
* перезапуск игры;
* перезарядка танка;
* отрисовка карты;
* режим мультиплеера.

Для разработки программного средства будет использоваться язык программирования C# и среда разработки Visual Studio 2022, а также Windows Form .

Windows Form используется с целью создания графического интерфейса.

## Состав и параметры технических и программных средств

Приложение должно функционировать на персональных компьютерах со следующими характеристиками:

* процессор Intel Pentium или лучше;
* оперативная память 1 GB или лучше;
* свободное место на диске 1 мегабайт или больше;
* операционная система Windows 7 или новее.

В данном разделе указаны минимальные технические требования для запуска программного средства. Для эксплуатации в реальных могут потребоваться более мощные технические средства. Программное средство должно корректно функционировать и на более мощном оборудовании.

# Моделирование предметной области

В качестве сетевого протокала используется протокол UDP.

## Протокол UDP

Протокол UDP (User Datagram Protocol) - это простой протокол передачи данных в компьютерных сетях, который не гарантирует доставку пакетов данных и не устанавливает соединение между устройствами.

Протокол UDP используется для передачи данных, которые не требуют гарантированной доставки или которые могут быть повторно переданы в случае потери. Это может включать потоковую передачу данных, мультимедийные данные и другие типы данных, для которых короткие задержки важнее гарантированной доставки.

UDP-пакет состоит из заголовка и данных. Заголовок включает в себя порты отправителя и получателя, длину пакета и контрольную сумму. UDP не предоставляет механизмов контроля перегрузки и управления потоком, поэтому для этого обычно используются другие протоколы, например, TCP.

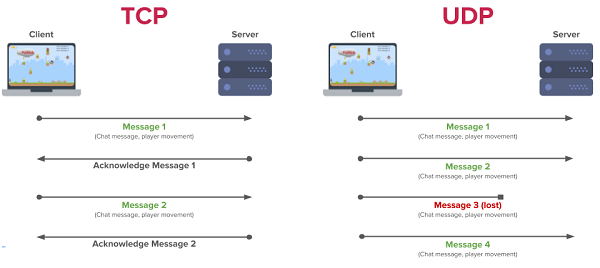


Рисунок 2.1 – Сравнение TCP и UDP

# Проектирование программного средства

## Структура программы

При разработке приложения будет использовано 9 модулей:

* MainWindow – модуль графического вывода и обработки сообщений

отправленыных окну

* Logic – модуль ии и обработки взаимодействия объектов
* Shell – модуль подпрограмм и данных снарядов
* BaseTank – модуль отвечающий за отрисовку таноков.
* Tank – модуль подпрограмм и данных танков
* Wall – модуль подпрограмм и данных стен
* UdpClient – модуль клиента.
* UdpServer – модуль сервера.
* Const – модуль хранит общие для всей программы типы и

константы.

## Проектирование интерфейса программного средства

### Главное меню

Лаунчер (главное меню приложения) должен состоять из простого окна с 3 кнопками Сервер, Клиент и Выход. Макет главного окна приложения представлен на рисунке 3.1.



Рисунок 3.1 – Главное меню

### Меню сервера

Меню сервера должно состоять из поля выбора количества игроков и полей для записи IP адресов клиентов. Также меню должно содержать кнопку возарата назад и кнопку запуска игры. Макет главного окна приложения представлен на рисунке 3.2.



Рисунок 3.2 – Меню сервера

### Меню клиента

Меню клиента должно состоять из поля для записи IP адреса сервера. Также меню должно содержать кнопку возарата назад и кнопку запуска игры. Макет главного окна приложения представлен на рисунке 3.3.



Рисунок 3.3 – Меню клиента

### Игровая зона

Игровая зона состоит из зоны отрисовки и кнопки возврата в меню. Макет главного окна приложения представлен на рисунке 3.4.



Рисунок 3.4 – Игровая зона

## Проектирование функционала программного средства

При создании игры очень важно сразу определить задачу и цели, а также хорошо составить рабочие алгоритмы. Искать ошибки в коде придётся в любом случае, но хорошо написанный алгоритм упрощает это в разы. В игре должны быть использованы следующие алгоритмы:

* Отправка данных сервером
* схема работы лаунчера;
* получение данных сервером.

### Обработчик нажатий пользователя

Отправка данных сервером не обходима для синхронизации всех пользователей . Блок-схема алгоритма данной функции приведена на рисунке 3.5.



Рисунок 3.5 – Отправка данных сервером

### Проверка попадания снаряда

Проверка попадания снаряда необходимо, чтобы избежать прохождения снарядов через объекты и их правильное движение по карте. Блок-схема алгоритма данной функции приведена на рисунке 3.6.



Рисунок 3.6 – Проверка попадания снаряда

### Получение данных сервером

Получение данны сервером для общитывания игровой логики. Блок-схема алгоритма данной функции приведена на рисунке 3.7.



Рисунок 3.7 – Получение данных сервером

# Разработка программного средства

## Проверка столкновения танка

Основой танка является алгоритм проверки столкновения с объектами.

Вызов необходимой процедуры осуществляется каждый раз, когда происходит обработка игровой логики. Сперва происходит проверка со стенами и выходы за границы поля, потом происходит проверка на столкновение с другим танком. Код процедуры будет предоставлен ниже.

private void CheckTankCollision(Tank selectTank,

Const.Direction direction)

{

bool move = true;

// Проверка на столкновение с преградами (стенами)

foreach (Wall wall in listWalls)

{

if (CheckCollision(selectTank.X1,

selectTank.Y1, selectTank.X2,

selectTank.Y2, direction, wall.X1, wall.Y1,

wall.X2, wall.Y2))

{

// Если произошло столкновение, остановить

движение

move = false;

break;

}

}

// Проверка на столкновение с другими танками

foreach (Tank tank in listTanks)

{

if (tank != selectTank && CheckCollision(

selectTank.X1, selectTank.Y1,

selectTank.X2, selectTank.Y2, direction,

tank.X1, tank.Y1, tank.X2, tank.Y2))

{

// Если произошло столкновение, остановить

движение

move = false;

break;

}

}

// Если нет столкновений, переместить танк

if (move)

{

selectTank.Move(direction);

}

}

## Движение танка

После проверки столкновений необходимо осуществить движение танка.

Сначала танк поворачивает и движется в указаную сторону, потом происходит изменение значений танка. Код функций будет предоставлен ниже.

public void Move(Const.Direction direction)

{

// Изменение координат танка в зависимости

от направления

switch (direction)

{

case Const.Direction.UP:

Y -= speed;

break;

case Const.Direction.RIGHT:

X += speed;

break;

case Const.Direction.DOWN:

Y += speed;

break;

case Const.Direction.LEFT:

X -= speed;

break;

}

// Обновление текущего направления танка

this.Direction = direction;

// Изменение состояния переменной FirstDark

для анимации гусиниц танка

FirstDark = !FirstDark;

tankGraphis.FirstDark = FirstDark;

// Обновление границ хитбокса танка

X1 = X - SIZE\_HITBOX;

X2 = X + SIZE\_HITBOX;

Y1 = Y - SIZE\_HITBOX;

Y2 = Y + SIZE\_HITBOX;

}

## Создание снаряда

Для возможности прохождения игры необходима возможность уничтожения противников или игрока.

Сначала происходит получнение параметров из танка, потом запись положения снаряда. Код данной функции будет приведён ниже.

public Shell(int x, int y, Const.Direction direction,

int damage, int speed)

{

// Установка урона, направления и скорости снаряда

this.Damage = damage;

this.Direction = direction;

this.Speed = speed \* 2;

// Установка начальных координат снаряда в

зависимости от направления

switch (direction)

{

case Const.Direction.UP:

this.X = x;

this.Y = y - 25;

break;

case Const.Direction.RIGHT:

this.X = x + 25;

this.Y = y;

break;

case Const.Direction.DOWN:

this.X = x;

this.Y = y + 25;

break;

case Const.Direction.LEFT:

this.X = x - 25;

this.Y = y;

break;

}

}

# Тестирование программного средства

## Запуск, подключение, соединение

Исходный набор данных: запуск программы, нажатие на кнопки запуска сервера. Запуск клиента.

Ожидаемый результат: открытие окон, начало игры.

Очевидно, что тест пройден успешно.

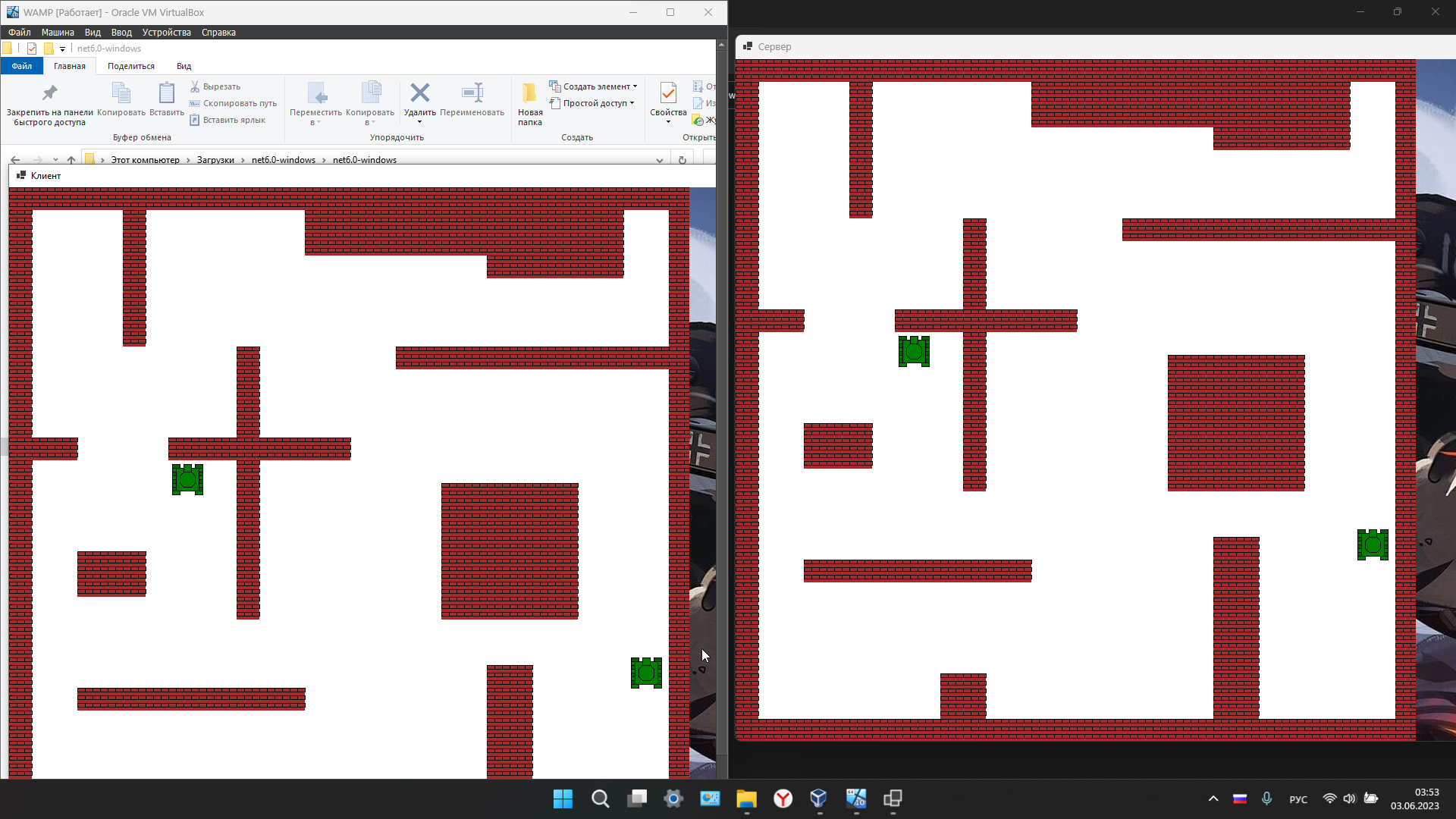


Рисунок 5.1 – Запуск игры

## Игровой процесс

Исходный набор данных: запрос на начало игры, связь клиентов.

Ожидаемый результат: два окна, игроки играют.

Очевидно, что тест пройден успешно.

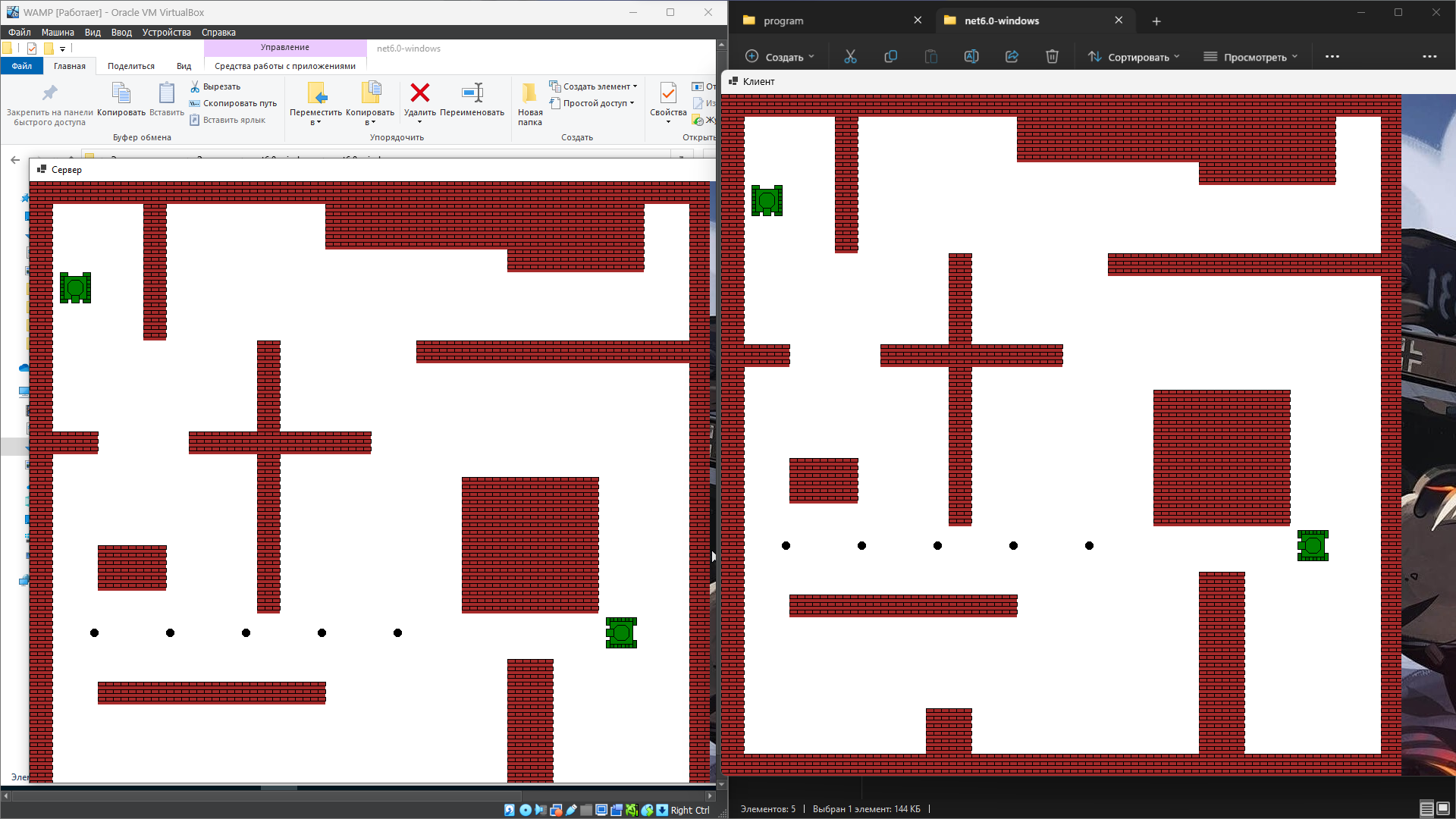


Рисунок 5.2 – Игровой процесс

## Конец игры

Исходный набор данных: 1 из игроков выигрывает.

Ожидаемый результат: сообщение о победе.

Очевидно, что тест пройден успешно.

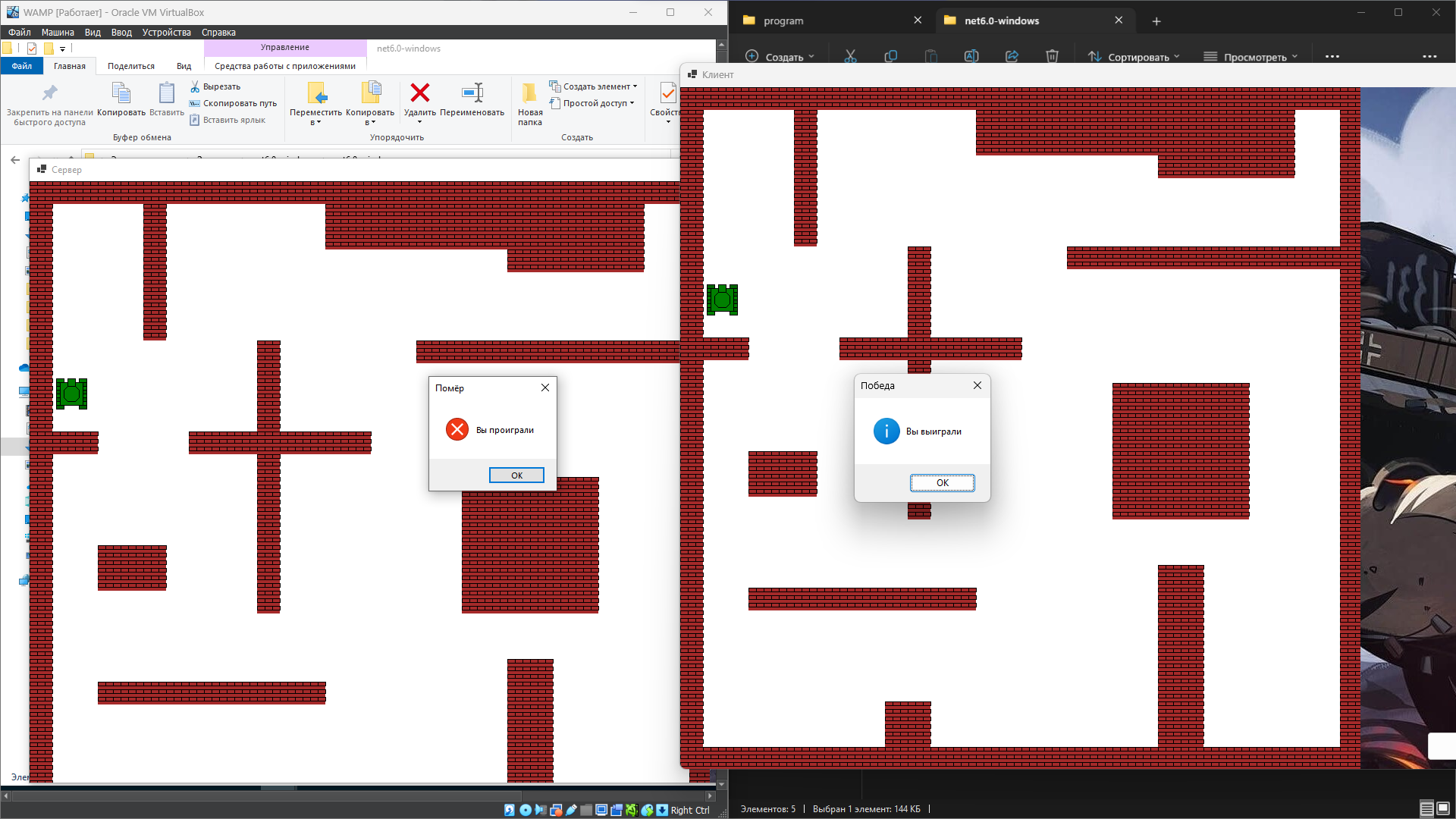


Рисунок 5.3 – Игровой процесс

# Руководство пользователя

## Минимальные системные требования

Для успешного запуска игрового приложения «точки» и комфортной работы с ним необходимо соответствие минимальным системным требованиям:

* процессор Intel Pentium или лучше;
* оперативная память 1 GB или лучше;
* свободное место на диске 1 мегабайт или больше;
* операционная система Windows 7 или новее.

## Установка

На установочном диске находится zip архив. После его распоковки программа пригодна к использованию.

## Главное меню

Главное меню представляет собой выбо использовать программу как сервер или как клиент. Также меню содержит кнопку для выхода из программы. Данное меню изображено на рисунке 6.1.

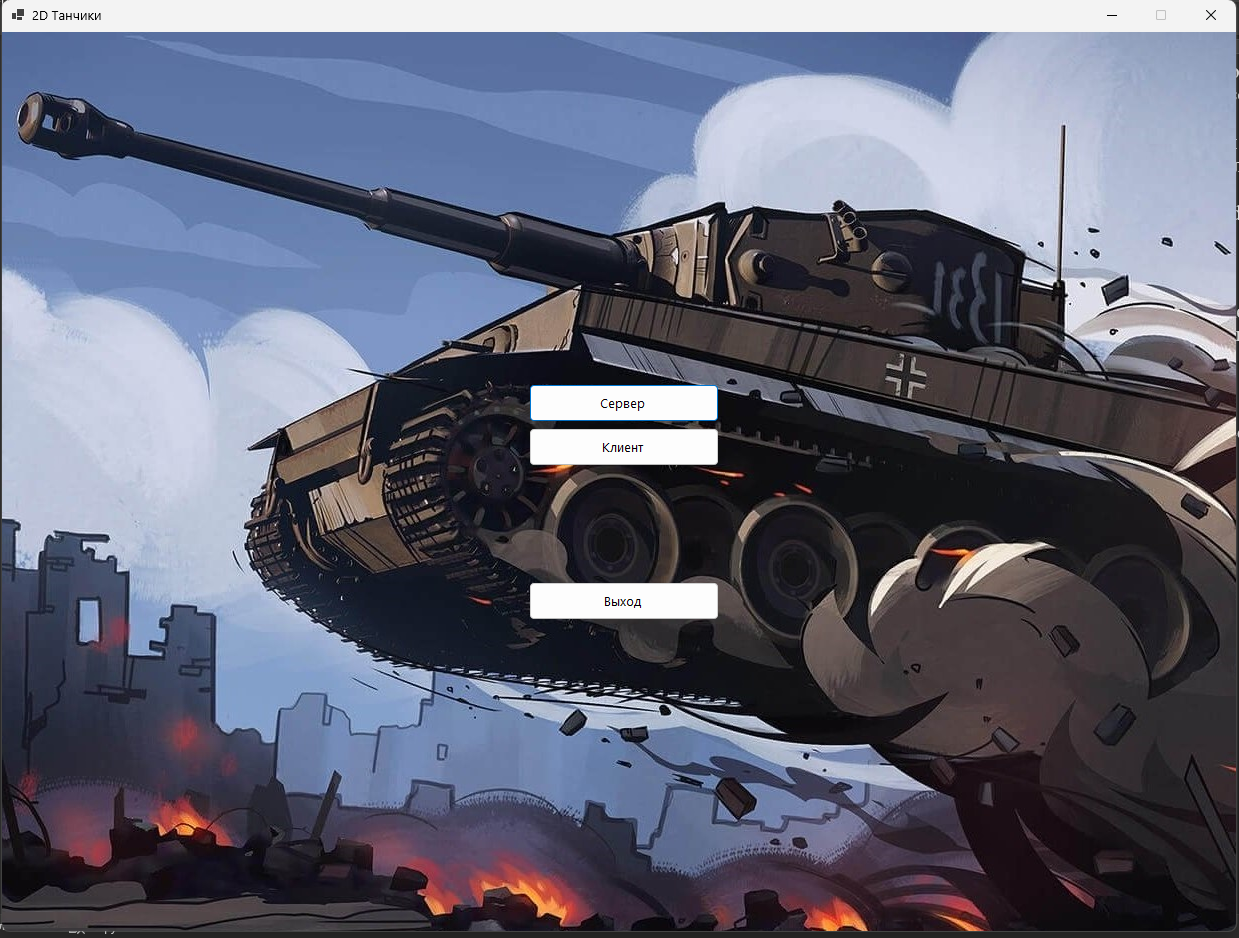


Рисунок 6.1 – Главное меню

## Меню сервера

Меню сервера представляет сабой список полей для ввода ip адресов игроков. После ввода ip адресов пользователь должен нажать кнопку начать игру. Также меню содержит кнопку для выхода из программы. Данное меню изображено на рисунке 6.2.

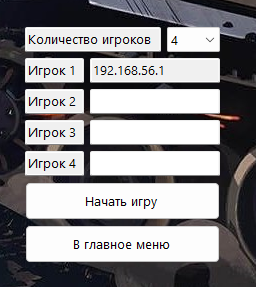


Рисунок 6.2 – Меню сервера

## Меню клиента

Меню клиента представляет сабой поле для ввода ip адреса сервера. После ввода ip адресов пользователь должен нажать кнопку начать игру. Также меню содержит кнопку для выхода из программы. Данное меню изображено на рисунке 6.3.

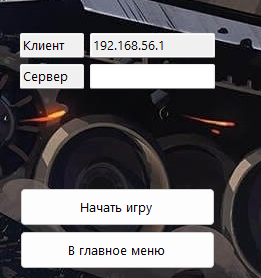


Рисунок 6.3 – Меню сервера

## Игровая зона

Игровая зона предсатвляет собой большое игровое поле слево, на котором изображаются танки, стены и снаряды. Основная задача игрока уничтожить остальные танки на карте. В правом нижнем углу находится кнопка возврата в главное меню. Данное меню изображено на рисунке 6.4.

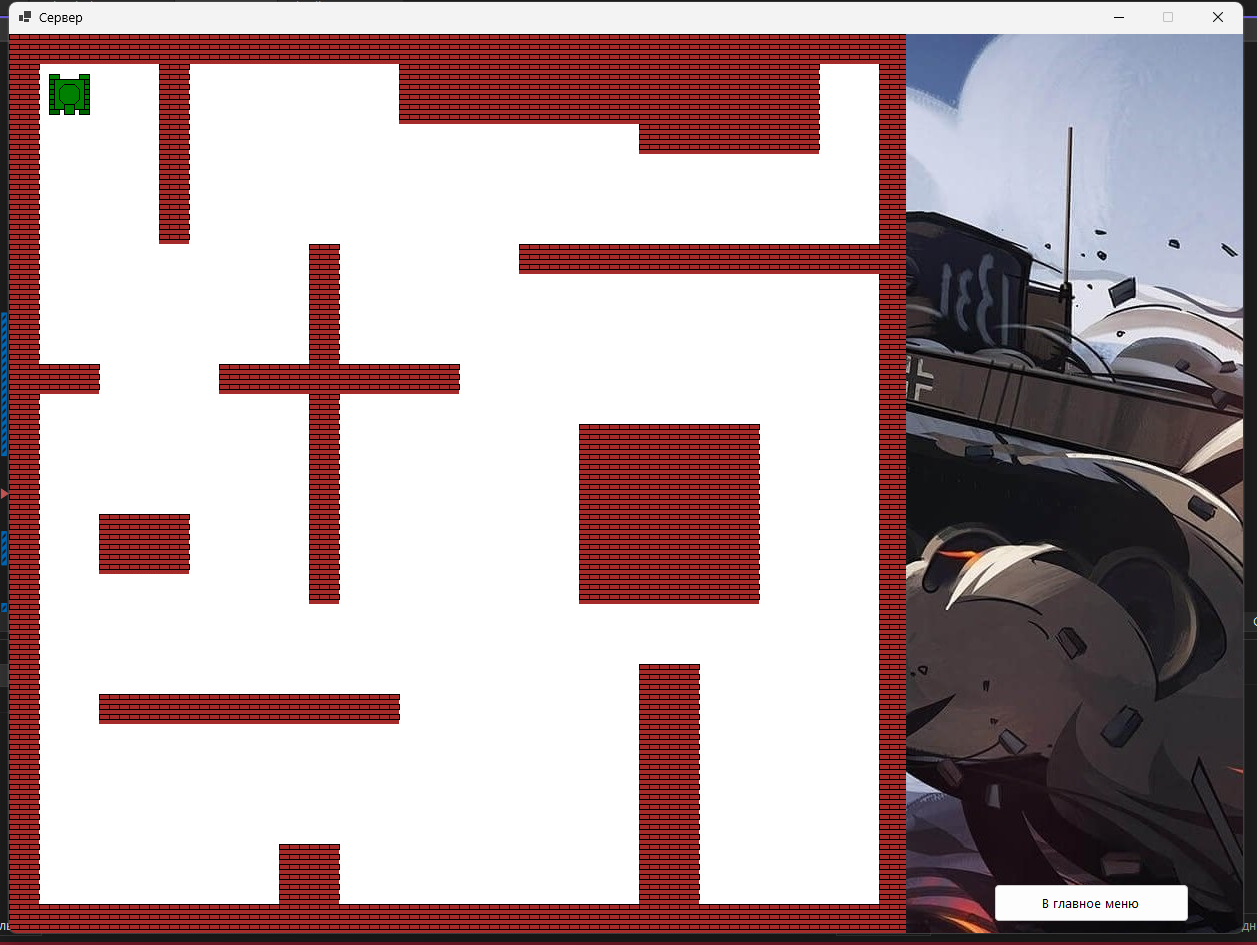


Рисунок 6.4 – Игровая зона

# Заключение

По итогу работы над курсовым проектом было разработано игровое приложение «2D танчики», обладающее графическим интерфейсом для взаимодействия с пользователем и позволяющее подключаться нескольким человекам на разных устройствах, используя протокол UDP.

Данное программное средство позволяет пользователю сыграть в игру «танчики». Среди преимуществ программы можно отметить интуитивно понятный интерфейс, простоту реализации, современный дизайн, а также низкие системные требования.

При разработке данного средства были успешно выполнены следующие поставленные задачи:

* приложение, которое позволяет сыграть с другом по сети;
* реализован сервер и клиент;
* после победы игрок получает сообщение о победе, а проигравшие о проигрыше;
* широкое использование многопоточности позволяющае приложение добится хорошей производительности.

Приложение прошло все этапы тестирования и продемонстрировало корректную быструю работу.

Бесценным результатом курсового проектирования является полученный опыт работы с компьютерными системами и сетями и Windows Forms. Был получен опыт работы с графическим интерфейсом пользователя (изучение компонентов форм, свойств и методов объектов, событий), обобщены и применены все ранее полученные знания.

B перспективе данное программное средство может быть усовершенствовано (оптимизация кода, добавление новых функций и т.п.).

# Список использованной литературы

[1] Computer Networking A Top-Down Approach 6th Edition. – Режим доступа: https://fita.vnua.edu.vn/wp-content/uploads/2013/05/TH02038\_Computer-networking-a-top-down-approach-6th.pdf – Дата доступа: 01.04.2023.

[2] "C# Network Programming" by Richard Blum. – Режим доступа: http://programming.etherealspheres.com/backup/ebooks%20-%20Other%20Programming%20Languages/C%20Sharp.Network.Programming.pdf – Дата доступа: 01.04.2023.

[3] METANIT.COM. – Режим доступа: https://metanit.com/sharp/net – Дата доступа: 01.04.2023.

[4] "C# 9.0 in a Nutshell: The Definitive Reference" by Joseph Albahari and Ben Albahari. – Режим доступа: https://archive.org/details/csharp-9-in-a-nutshell-the-definitive-reference/page/n15/mode/2up – Дата доступа: 01.04.2023.

Приложение А

(обязательное)

using Kyrsach.Game\_objects;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Text.Json;

namespace Kyrsach.Networks.Local

{

internal class UdpServer

{

// Интерфейс

// Константы

// Типы

// Поля

public string[] ClientsIP { get; set; }

// Методы

public UdpServer(int countTank)

{

ClientsIP = new string[countTank];

endPoints = new IPEndPoint[countTank];

this.countTank = countTank;

// Создание буфера для получения данных

bytes = new byte[sizeof(Int32)];

// Создание UDP сокета

socket = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

// Привязка сокета к локальному адресу и порту

socket.Bind(new IPEndPoint(IPAddress.Any,

Const.PORT\_FOR\_GAME));

}

public async void SendData(Tank[] listTank,

List<Shell> listShell)

{

// Создание пустой строки для сериализации

данных

string str = "";

// Сериализация данных о танках

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

{

str +=

JsonSerializer.Serialize<Tank>(listTank[i]);

}

// Сериализация данных о снарядах

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

{

str +=

JsonSerializer.Serialize<Shell>(listShell[i]);

}

// Отправка данных каждому клиенту, начиная с

индекса 1

for (int i = 1; i < countTank; i++)

{

// Преобразование строки в массив байтов и

отправка данных по UDP

await socket.SendToAsync(new

ArraySegment<byte>

(Encoding.UTF8.GetBytes(str)),

SocketFlags.None, endPoints[i]);

}

}

public async void GetDate(Tank[] listTank)

{

// Получение данных от каждого клиента, начиная

с индекса 1

for (int i = 1; i < countTank; i++)

{

// Создание объекта EndPoint для получения

данных от конкретного клиента

EndPoint remoteEndPoint = new

IPEndPoint(IPAddress.Parse(ClientsIP[i]),

Const.PORT\_FOR\_GAME);

// Асинхронное получение данных по UDP

await socket.ReceiveFromAsync(new

ArraySegment<byte>(bytes),

SocketFlags.None, remoteEndPoint);

// Преобразование полученных данных в

направление движения и сохранение в

объекте танка

listTank[i].keyDirection =

(Const.Direction)BitConverter.ToInt32(bytes);

}

}

public void SetIPAddres()

{

// Установка IP-адресов для каждого клиента,

начиная с индекса 1

for (int i = 1; i < countTank; i++)

{

// Создание объекта IPEndPoint с

использованием IP-адреса клиента и порта

endPoints[i] = new

IPEndPoint(IPAddress.Parse(ClientsIP[i]),

Const.PORT\_FOR\_GAME);

}

}

public void SendAuthenticationData()

{

// Создание нового сокета для отправки данных

using (var socket = new

Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp))

{

// Привязка сокета к адресу и порту

socket.Bind(new IPEndPoint(IPAddress.Any,

Const.PORT\_FOR\_INFO));

// Создание буфера для данных

byte[] buffer = new byte[sizeof(Int32) \*

2];

// Отправка аутентификационных данных

каждому клиенту

for (int i = 1; i < countTank; i++)

{

// Запись количества танков и

идентификатора клиента в буфер

Buffer.BlockCopy(

BitConverter.GetBytes(countTank),

0, buffer, 0, sizeof(int));

Buffer.BlockCopy(

BitConverter.GetBytes(i),

0, buffer, sizeof(int), sizeof(int));

// Отправка буфера по указанному

IP-адресу и порту

socket.SendTo(buffer, new

IPEndPoint(IPAddress.Parse(

ClientsIP[i]), Const.PORT\_FOR\_INFO));

}

}

}

public void Close()

{

socket.Close();

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Реализация

// Константы

// Типы

// Поля

private IPEndPoint[] endPoints;

private int countTank;

private byte[] bytes;

private Socket socket;

// Методы

}

}

using Kyrsach.Game\_objects;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Text.Json;

namespace Kyrsach.Networks.Local

{

internal class UdpClient

{

// Интерфейс

// Константы

// Типы

// Поля

public int NumbTank { get; set; } = -1;

public int CountTank { get; set; } = -1;

// Методы

public UdpClient(string serverIP, bool server)

{

this.serverIP = serverIP;

// Создание UDP сокета

socket = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

// Привязка сокета к локальному IP-адресу и

порту игры, если текущий экземпляр - клиент

if (!server)

{

socket.Bind(new IPEndPoint(IPAddress.Any,

Const.PORT\_FOR\_GAME));

}

// Создание конечной точки с указанным

IP-адресом сервера и портом игры

endPoint = new

IPEndPoint(IPAddress.Parse(serverIP),

Const.PORT\_FOR\_GAME);

}

public async void SendDate(Tank tank)

{

await socket.SendToAsync(new

ArraySegment<byte>(BitConverter.GetBytes(

Convert.ToInt32(tank.keyDirection))),

SocketFlags.None, endPoint);

}

// Получение данных по UDP

public async void GetData(Tank[] tanks,

int numbTank, List<Shell> shells, object lockShell)

{

bytes = new byte[1024];

await socket.ReceiveFromAsync(new

ArraySegment<byte>(bytes),SocketFlags.None,

endPoint);

string str = Encoding.UTF8.GetString(bytes);

// Разделение данных

Queue<int> first = new Queue<int>();

Queue<int> last = new Queue<int>();

int i;

for (i = 0; str[i] != '\0'; i++)

{

if (str[i] == '{')

{

first.Enqueue(i);

}

else if (str[i] == '}')

{

last.Enqueue(i);

}

}

// Обработка данных танков

for (i = 0; i < CountTank; i++)

{

if (first.Count > 0 && last.Count > 0)

{

string strTank =

str.Substring(first.Peek(),

last.Dequeue() - first.Dequeue() + 1);

Tank tank = JsonSerializer.Deserialize<Tank>

(strTank);

tank.Initialization();

tanks[i] = tank;

}

}

// Обработка данных снарядов

lock (lockShell)

{

shells.Clear();

while (first.Count > 0 && last.Count > 0)

{

if (last.Peek() < first.Peek())

{

last.Dequeue();

continue;

}

string strShell =

str.Substring(first.Peek(),

last.Dequeue() - first.Dequeue() + 1);

Shell shell =

JsonSerializer.Deserialize<Shell>

(strShell);

shell.Initialization();

shells.Add(shell);

}

}

}

// Получение аутентификационных данных

public void GetAuthenticationData()

{

using (var socket = new

Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp))

{

socket.Bind(new IPEndPoint(IPAddress.Any,

Const.PORT\_FOR\_INFO));

byte[] buff = new byte[sizeof(Int32) \* 2];

// Получение данных от удаленного хоста

EndPoint remoteEndPoint = new

IPEndPoint(IPAddress.Parse(serverIP),

Const.PORT\_FOR\_INFO);

socket.ReceiveFrom(buff,

ref remoteEndPoint);

// Извлечение количества танков и номера

танка из полученных данных

CountTank = BitConverter.ToInt32(buff, 0);

NumbTank = BitConverter.ToInt32(buff,

sizeof(Int32));

}

}

public void Close()

{

socket.Close();

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Реализация

// Константы

// Типы

// Поля

private string serverIP;

private IPEndPoint endPoint;

private byte[] bytes;

private Socket socket;

// Методы

}

}

using Kyrsach.Game\_objects.Base;

using System.Text.Json.Serialization;

namespace Kyrsach.Game\_objects

{

[Serializable]

internal class Tank

{

// Интерфейс

// Константы

[JsonIgnore]

public const int SIZE\_HITBOX = 20;

[JsonIgnore]

public const int SIZE\_MOVE = 5;

// Типы

// Поля

[JsonIgnore]

public int X1 { get; set; }

[JsonIgnore]

public int Y1 { get; set; }

[JsonIgnore]

public int X2 { get; set; }

[JsonIgnore]

public int Y2 { get; set; }

[JsonIgnore]

public Const.Direction keyDirection { get; set; } =

Const.Direction.DEFAULT;

public int RemainingTimeReload { get; set; }

public int HP { get; set; } = 1;

public int X { get; set; }

public int Y { get; set; }

public bool FirstDark { get; set; } = true;

public Const.Direction Direction { get; set; }

// Методы

public Tank(int x, int y,

Const.Direction direction)

{

this.X = x;

this.Y = y;

this.Direction = direction;

// Выполнение инициализации танка

Initialization();

RemainingTimeReload = timeReload;

}

public Tank()

{

}

public void Initialization()

{

tankGraphis = new BaseTank();

// Установка границ прямоугольной области

столкновения танка

X1 = this.X - SIZE\_HITBOX;

X2 = this.X + SIZE\_HITBOX;

Y1 = this.Y - SIZE\_HITBOX;

Y2 = this.Y + SIZE\_HITBOX;

}

public void Paint(Graphics graphics)

{

tankGraphis.Paint(graphics, Direction, X, Y);

}

public void Move(Const.Direction direction)

{

// Изменение координат танка в зависимости от4

направления

switch (direction)

{

case Const.Direction.UP:

Y -= speed;

break;

case Const.Direction.RIGHT:

X += speed;

break;

case Const.Direction.DOWN:

Y += speed;

break;

case Const.Direction.LEFT:

X -= speed;

break;

}

// Обновление текущего направления танка

this.Direction = direction;

// Изменение состояния переменной FirstDark для

анимации гусиниц танка

FirstDark = !FirstDark;

tankGraphis.FirstDark = FirstDark;

// Обновление границ хитбокса танка

X1 = X - SIZE\_HITBOX;

X2 = X + SIZE\_HITBOX;

Y1 = Y - SIZE\_HITBOX;

Y2 = Y + SIZE\_HITBOX;

}

public Shell Shoot()

{

return new Shell(X, Y, Direction, damage,

speed);

}

public void StartTankReload()

{

RemainingTimeReload = timeReload;

}

public void TankReload()

{

if (RemainingTimeReload > 0)

{

RemainingTimeReload--;

}

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Реализация

// Константы

private const int LENGTH\_GUN = 20;

// Типы

// Поля

private int damage = 1;

private int speed = 5;

private int timeReload = 10;

//техническая информация отображаемая

private BaseTank tankGraphis;

// Методы

}

}