

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
государственное бюджетное образовательное учреждение высшего образования
УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных систем и технологий
Кафедра «Измерительно-вычислительные комплексы»

Пояснительная записка
к коллекционной карточной игре “Project_A”

Выполнили:
ст. гр. ИСТбд-32 Сергеев П.С. / Тимошин Я.И.
(Фамилия И.О.)

Проверил:
преподаватель Кандаулов В.М.
(Фамилия И.О.)

Ульяновск
2017

Содержание

Введение	3
1. Техническое задание	4
1.1 Общее описание проекта	4
1.2 Описание серверной части и её функционала	5
1.3 Описание клиентской части и её функционала	5
2. Прикладное программное обеспечение системы	6
2.1 Общая характеристика прикладного программного обеспечения	6
2.2 Состав прикладного программного обеспечения	6
2.3 Описание основных методов приложения	7
3. Руководство пользователя	7
Заключение	12
Список источников	13
Приложение	14

Введение

Проект представляет собой многопользовательскую коллекционную карточную игру (ККИ). Данный жанр игр стал особенно популярен в последние 3 года и имеет множество различных форматов.

Впервые формат карточной игры, используемый в данном проекте, был представлен как элемент AAA-проекта 2015 года компании CD Projekt RED - The Witcher 3: Wild Hunt.

Ключ к победе - в грамотном распределении ресурсов на протяжении всей игры, с учётом необходимости доминирования на поле боя. Прояви себя талантливым стратегом - ведь зачастую, проиграв одну битву, можно выиграть войну!

Ключевая особенность игр данного жанра – их доступность, и наш проект удовлетворяет этому критерию в полной мере.

Хотя большинство подобных игр и позиционируются как free-to-play (f2p), многие из них требуют значительных временных, а иногда и денежных вложений для выхода на достойный уровень игры.

В нашем же проекте силу колоды игрока определяет только лишь его навык (и, разумеется, везение), а не сумма, вложенная в покупку наборов карт.

1. Техническое задание

1.1 Общее описание проекта








Project_A – игра, между парой соперников, разворачивающаяся на поле из 4 линий: линия карт ближнего боя (melee) и линия карт дальнего боя (range) для каждого из игроков соответственно.

Игра ведется по системе best-of-3 (bo3) – до победы в двух раундах. Исход каждого раунда определяет суммарная сила отрядов игрока на обеих линиях. В случае равной силы отрядов объявляется ничья, а каждому из игроков присуждается очко победы в раунде.

Каждый из игроков начинает игру с колодой из 8 карт составленной им самим при подготовке к матчу. При подготовке к матчу игрок составляет колоду по системе draft, состоящей из 8 этапов. На каждом этапе предоставляется выбор одной (1) из 3-х различных случайных карт. Карты в игре обладает редкостью. Всего существует 4 редкости карт: обычная, редкая, эпическая и легендарная. На первых 4-х этапах игроку могут быть предложены карты любой редкости; на 5, 6 и 7-м – минимум редкие, а на 8-м – минимум эпические.

Помимо редкости каждая карта имеет свой тип. Всего представлено 7 типов карт (см. таблицу 1).

Таблица 1.

Иконка	Описание
	Карта ближнего боя. Выставляется на линию ближнего боя игрока, разыгравшего ее.
	Карта дальнего боя. Выставляется на линию дальнего боя игрока, разыгравшего ее.
	Карта ближнего боя. Шпион. Выставляется на линию ближнего боя противника. Добавляет в вашу руку копию случайной карты из его руки.
	Карта дальнего боя. Шпион. Выставляется на линию дальнего боя противника. Добавляет в вашу руку копию случайной карты из его руки.
	Карта погоды. Направлена против ВСЕХ отрядов ближнего боя, находящихся на поле. Устанавливает их силу равной силе этой карты.
	Карта погоды. Направлена против ВСЕХ отрядов дальнего боя, находящихся на поле. Устанавливает их силу равной силе этой карты.
	Карта погоды. Возвращает силу ВСЕХ отрядов на поле боя к ее стандартному значению.

В ходе матча игроки поочередно выбирают по одной (1) карте из своей руки и разыгрывают её. Если игрок не выбрал карту, или у него не осталось карт в руке, то он пасует. После паса одного игрока другой получает право выставлять карты до своего паса. Пас от обоих игроков означает завершение раунда.

1.2 Описание серверной части и её функционала

Серверная часть отвечает за подбор соперников, синхронизацию клиентов игроков в ходе матча, выбор победителя в раунде и матче, а также выбор случайной карты, воруемой при использовании карты шпиона.

Помимо основного обеспечения работоспособности сервер ведет файл лога для каждого матча, который содержит краткое описание хода игры.

1.3 Описание клиентской части и её функционала

Клиентская часть отвечает за составление колоды игрока, логику работы карт, а также за графическое представление игры.

2. Прикладное программное обеспечение системы

2.1 Общая характеристика прикладного программного обеспечения

Данное ПО включает серверную и клиентскую часть. Где каждая часть имеет различное представление: серверная часть в виде консольного приложения; клиентская часть имеет вид приложения Windows Forms. Они представлены в виде двух приложений, что обеспечивает легкость распространения и установки.

Связь между клиентами и сервером обеспечивается специально разработанными методами, основанными на использовании протокола TCP в виде классов TcpClient и TcpListener.

Проект реализован в среде Microsoft Visual Studio Community 2015 на базе Microsoft .NET Framework 4.6.

2.2 Состав прикладного программного обеспечения

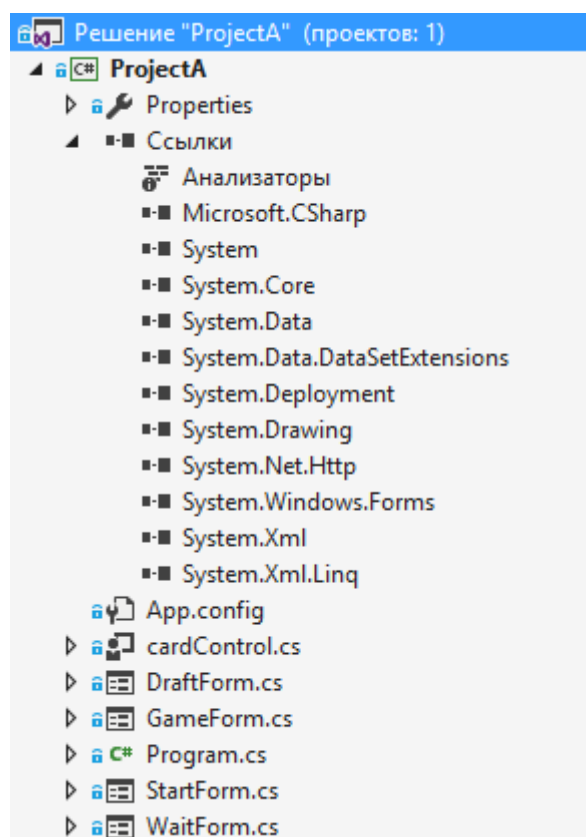


Рис. 1. Проект ProjectA – клиент игры.

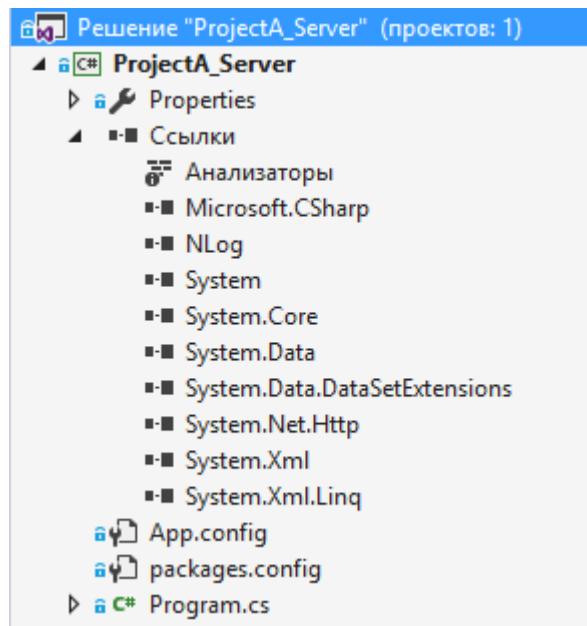


Рис. 2. Проект ProjectA_Server – сервер игры.

2.3 Описание основных методов приложения

В проекте клиента:

- TakeThree из DraftForm.cs – выбирает 3 карты для этапа драфта, с учетом редкости карт и номера этапа;
- UnpackLine из GameForm.cs – распаковывает линию карт из строки, используемой для обмена данными между сервером и клиентом;
- Game из GameForm.cs – метод отвечающий за обработку команд поступающих с сервера;
- SendLine из GameForm.cs – метод упаковывающий и передающий линию карт на сервер.

В проекте сервера:

- RemoveClients – находит пару клиентов по одному из них и отключает от сервера (с ошибкой или без);
- Game – реализует непосредственно игру, как последовательность ходов и раундов, а также определение победителя.

Общие:

- Send – передает данные TcpClient'y;
- Receive – получает данные от TcpClient'a.

3. Руководство пользователя

При запуске игры пользователь видит стартовый экран (рис. 3). По нажатию на кнопку «START» он переходит к драфту колоды.



Рис.3. Стартовый экран.

Драфт состоит из 8 этапов, на каждом из которых игрок должен выбрать одну из трех предложенных ему карт (рис. 4).

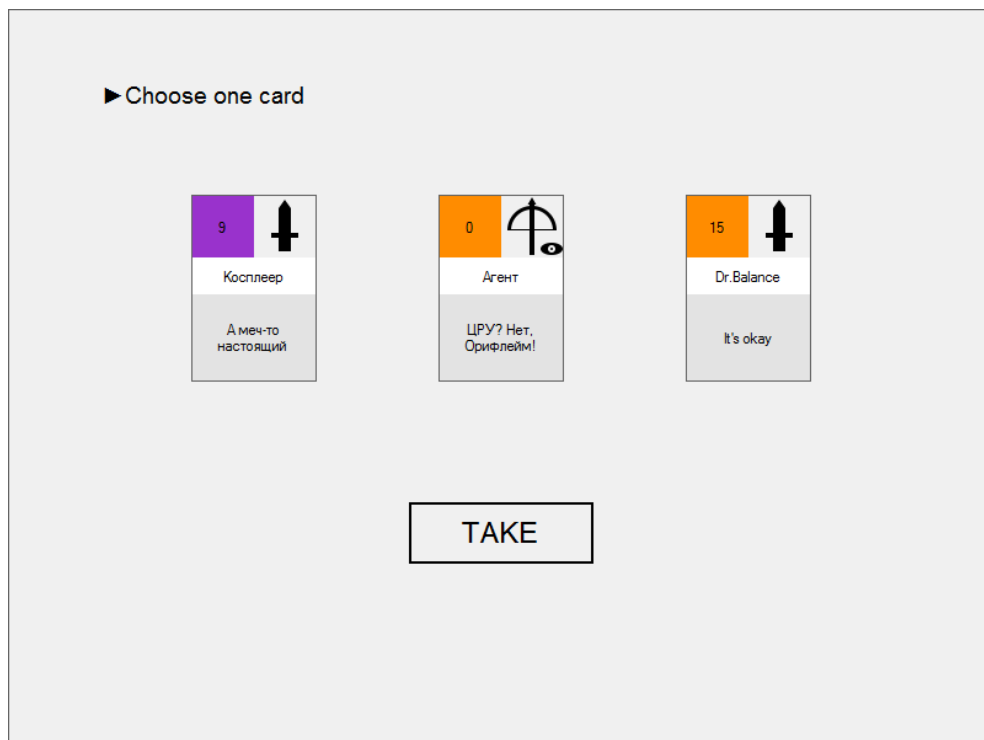


Рис. 4. Окно драфта.

После составления колоды из 8 карт начинается процесс поиска оппонента (рис. 5).

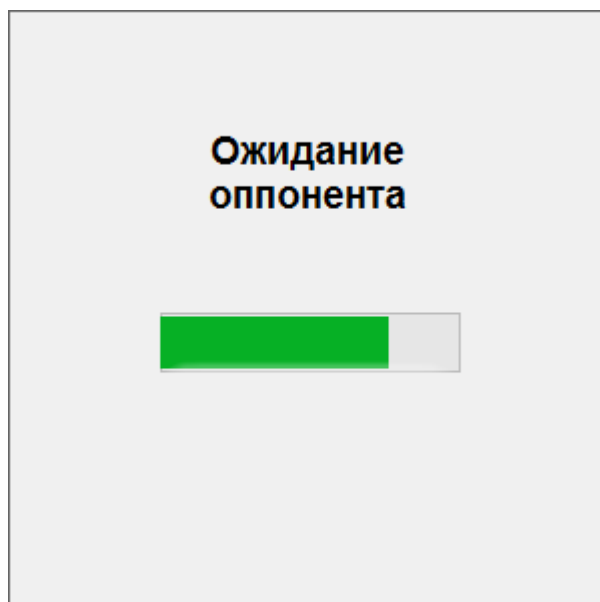


Рис. 5. Поиск оппонента.

Когда противник найден, начинается матч (рис. 6).

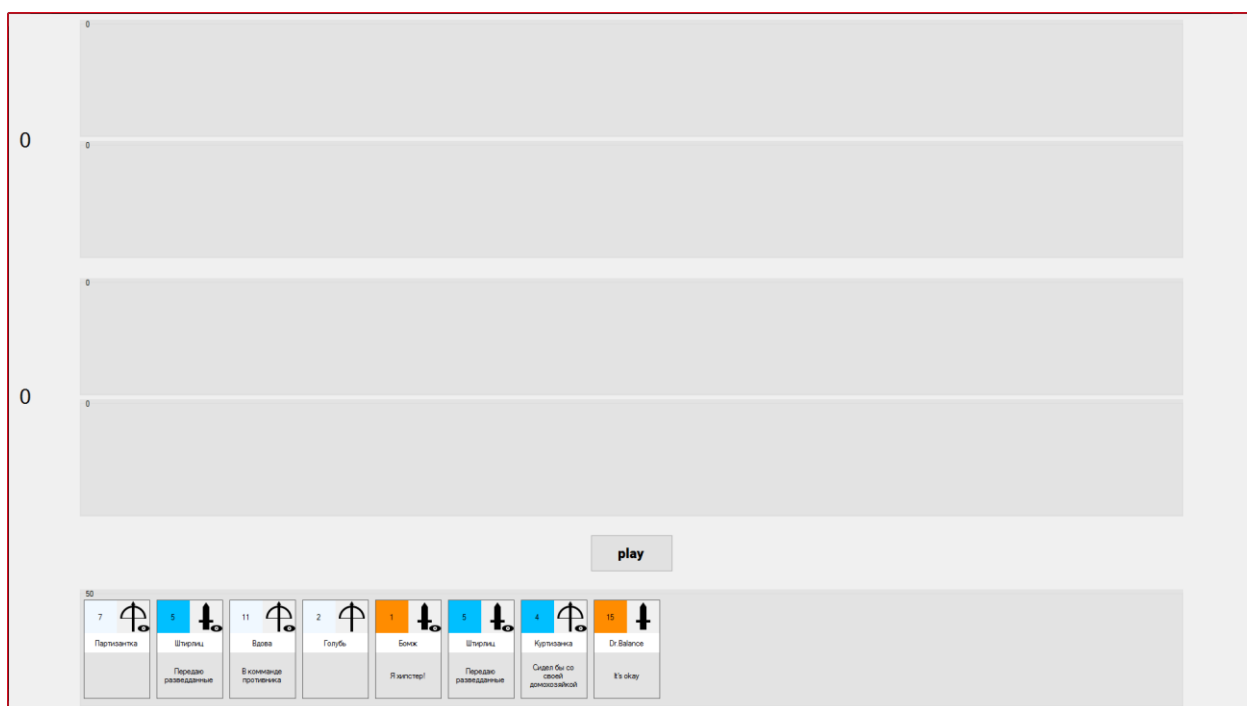


Рис. 6. Игровое поле.

Игрок во время своего хода имеет возможность выбрать карту из своей руки (самая нижняя линия) и по нажатию кнопки «play» разыграть её. Будьте внимательны – если вы не выберете карту, то ваши действия будут расценены как пас.

Когда видно, что один из игроков захватил преимущество на поле, игроки могут завершить раунд спасовав (рис. 7). И начать следующий с чистого листа (рис. 8).

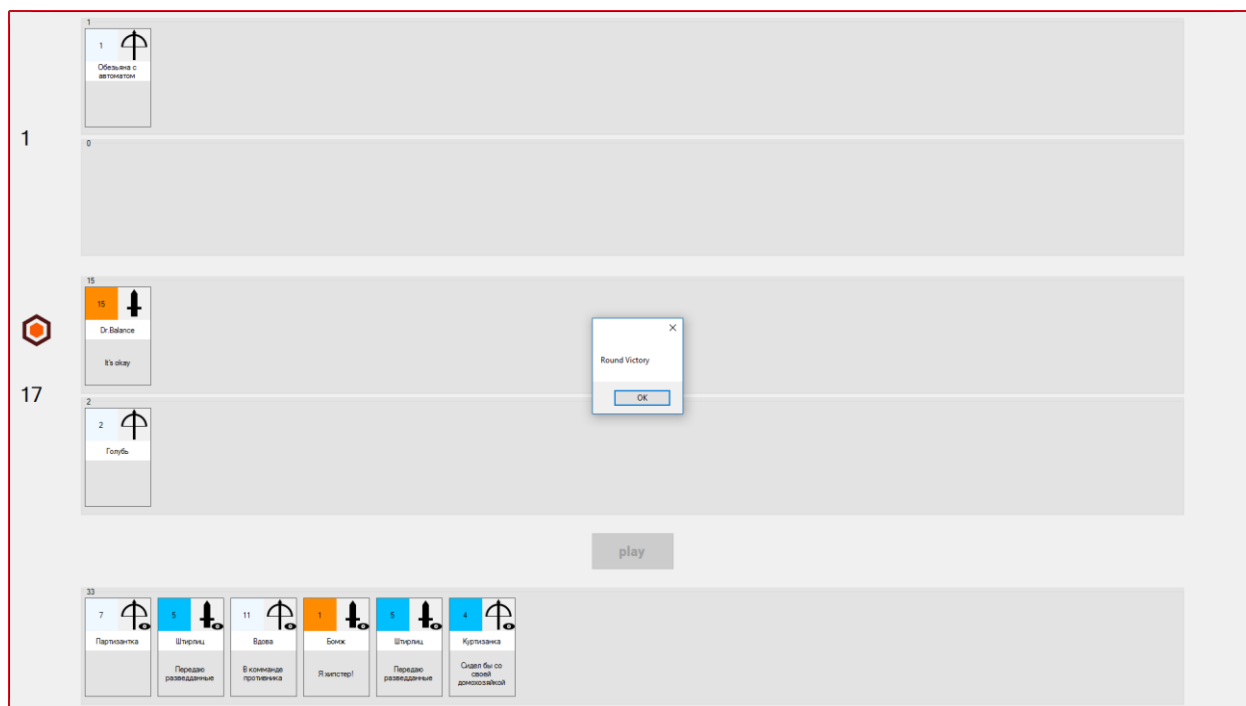


Рис. 7. Конец раунда.

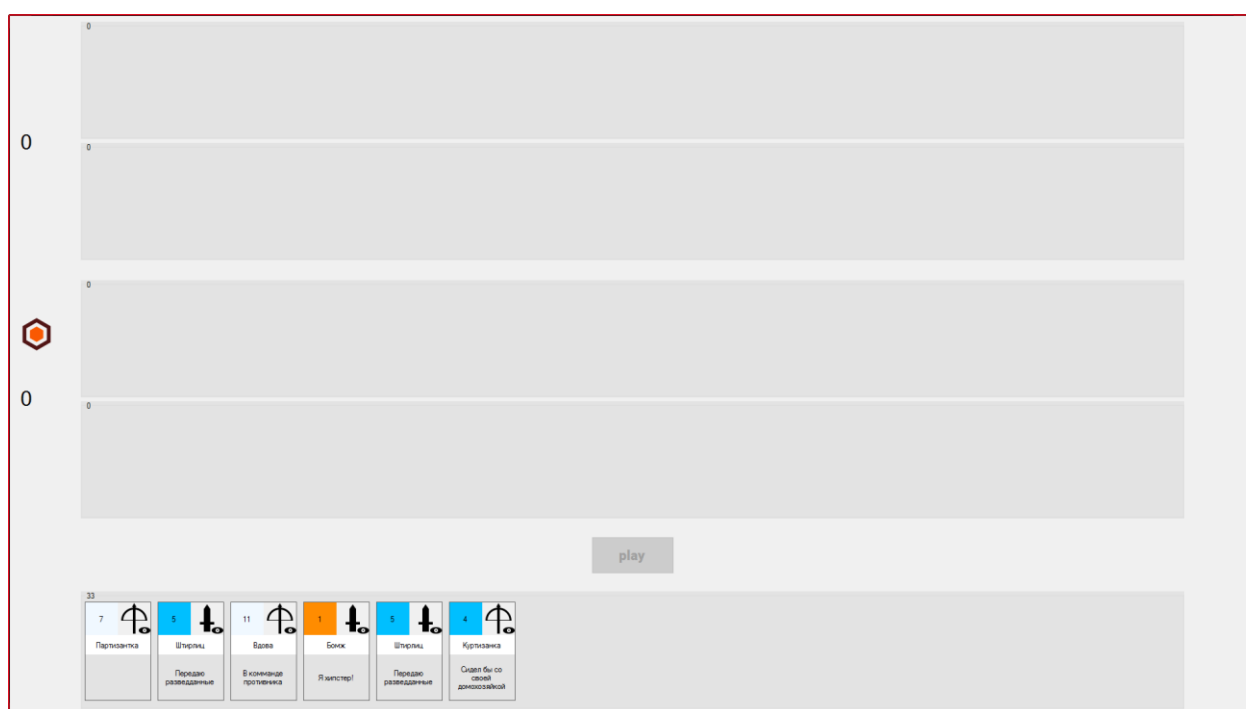


Рис. 8. Второй раунд.

Когда один из игроков (или даже оба) одерживают победу в двух раундах матч завершается (рис. 9).

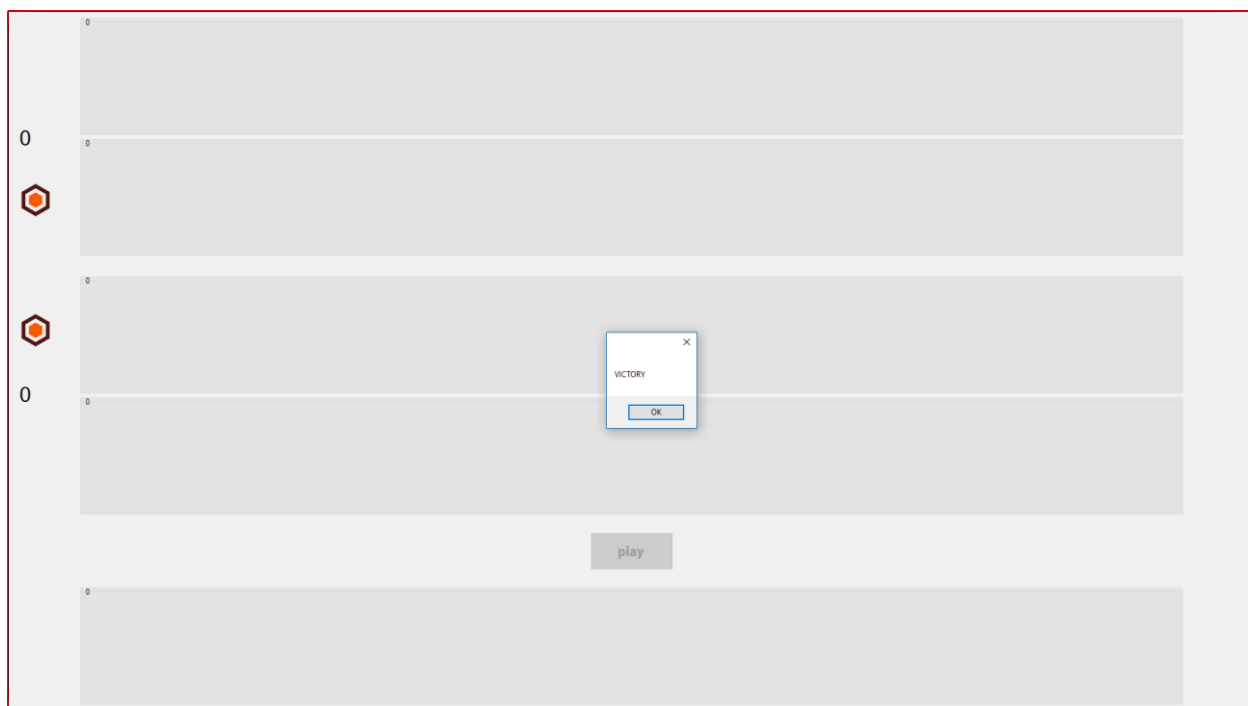


Рис. 10. Завершение игры.

Заключение

Проектирование и реализация курсовой работы были выполнены в соответствии с утвержденным техническим заданием.

Особенные сложности были вызваны незнанием некоторых аспектов Windows Forms и работы с процессами, однако данные сложности были преодолены в процессе работы над проектом.

Игра получила довольно широко развитую логику, понятный интерфейс и интересный геймплей.

К основным достоинствам можно отнести:

- простоту освоения;
- небольшой объем клиента;
- высокую реиграбельность.

Недостатки:

- привязка к платформе Windows;
- недостаточно «живой» интерфейс пользователя.

В итоге получен полноценный работоспособный продукт.

Список источников

1. MSDN – сеть разработчиков Microsoft. – URL: <https://msdn.microsoft.com/ru-ru/dn308572.aspx> (дата обращения 19.05.2017).
2. NLog. – URL: <http://nlog-project.org/> (дата обращения 19.05.2017).
3. Hello World · GitHub Guides. – URL: <https://guides.github.com/activities/hello-world/> (дата обращения 19.05.2017).

Приложение

Base64 коды изображений вырезаны для компактности кода.

cardControl.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace CardControl
{
    public partial class Card : UserControl
    {
        public static List<int> Common = new List<int> { 2, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44 };
        public static List<int> Rare = new List<int> { 3, 4, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 45, 46, 47 };
        public static List<int> Epic = new List<int> { 5, 9, 10, 11, 12, 13, 14, 15 };
        public static List<int> Legendary = new List<int> { 1, 6, 7, 8 };
        public Card()
        {
            InitializeComponent();
        }

        public Card(int id, bool inHand)
        {
            InitializeComponent();
            InHand = inHand;
            ID = id;
            switch (id)
            {
                case 1:
                    DefaultPower = 15;
                    Name = "Dr.Balance";
                    Text = "It's okay";
                    Type = AttackType.Melee;
                    break;
                case 2:
                    DefaultPower = 1;
                    Type = AttackType.Melee;
                    Name = "Хипстер";
                    Text = "";
                    break;
                case 3:
                    DefaultPower = 4;
                    Type = AttackType.Melee;
                    Name = "SMOrc";
                    Text = "Где лицо???";
                    break;
                case 4:
                    DefaultPower = 5;
                    Type = AttackType.Melee_spy;
                    Name = "Штирлиц";
                    Text = "Передаю разведданные";
                    break;
                case 5:
                    DefaultPower = 3;
                    Type = AttackType.Range_spy;
                    Name = "Домохозяйка";
                    Text = "Дорогой, я тебе постелила";
            }
        }
    }
}
```

```

        break;
case 6:
    DefaultPower = 1;
    Type = AttackType.Melee_spy;
    Name = "Бомж";
    Text = "Я хипстер!";
    break;
case 7:
    DefaultPower = 15;
    Type = AttackType.Range;
    Name = "Тунгуска-М1";
    Text = "Сама не летает и другим не дает";
    break;
case 8:
    DefaultPower = 0;
    Type = AttackType.Range_spy;
    Name = "Агент";
    Text = "ЦРУ? Нет, Орифлейм!";
    break;
case 9:
    DefaultPower = 10;
    Type = AttackType.Melee;
    Name = "хРеке";
    Text = "Невероятный игрок";
    break;
case 10:
    DefaultPower = 4;
    Type = AttackType.Range_spy;
    Name = "Папарацци";
    Text = "Улыбнитесь - вас снимают";
    break;
case 11:
    DefaultPower = 9;
    Type = AttackType.Melee;
    Name = "Косплеер";
    Text = "А меч-то настоящий";
    break;
case 12:
    DefaultPower = 8;
    Type = AttackType.Range;
    Name = "Леголас";
    Text = "Здесь стрелы все-таки кончаются";
    break;
case 13:
    DefaultPower = 9;
    Type = AttackType.Range;
    Name = "Балиста";
    Text = "Из балисты по хипстерам";
    break;
case 14:
    DefaultPower = 3;
    Type = AttackType.Melee_spy;
    Name = "V1lat";
    Text = "Есть пара инсайдов...";
    break;
case 15:
    DefaultPower = 4;
    Type = AttackType.Melee_spy;
    Name = "Сноуден";
    Text = "Публикует ваши секретные архивы";
    break;
case 16:
    DefaultPower = 7;
    Type = AttackType.Melee_spy;

```

```

        Name = "Скрытый бассейн";
        Text = "Хорошие игроки? Не у тебя!";
        break;
case 17:
    DefaultPower = 6;
    Type = AttackType.Melee_spy;
    Name = "Цыган";
    Text = "Какой конь? Не было коня!";
    break;
case 18:
    DefaultPower = 4;
    Type = AttackType.Range_spy;
    Name = "Куртизанка";
    Text = "Сидел бы со своей домохозяйкой";
    break;
case 19:
    DefaultPower = 5;
    Type = AttackType.Range_spy;
    Name = "Бабушка у подъезда";
    Text = "Ишь ты - не здороваются!";
    break;
case 20:
    DefaultPower = 6;
    Type = AttackType.Range_spy;
    Name = "IRL стример";
    Text = "Только не прыгай с крыши";
    break;
case 21:
    DefaultPower = 5;
    Type = AttackType.Melee;
    Name = "Начальник завода";
    Text = "Чего сидишь? Иди работай!";
    break;
case 22:
    DefaultPower = 7;
    Type = AttackType.Melee;
    Name = "Сисадмин";
    Text = "Я-ж программист!";
    break;
case 23:
    DefaultPower = 3;
    Type = AttackType.Range;
    Name = "Скелет-лучник";
    Text = "Ну хоть не пехота!";
    break;
case 24:
    DefaultPower = 5;
    Type = AttackType.Range;
    Name = "Имперский штурмовик";
    Text = "Я попал! (нет)";
    break;
case 25:
    DefaultPower = 6;
    Type = AttackType.Range;
    Name = "Хандзо-мейн";
    Text = "Сложная геометрия";
    break;
case 26:
    DefaultPower = 2;
    Type = AttackType.Melee;
    Name = "Скелет-мечник";
    Text = "";
    break;
case 27:

```



```

    DefaultPower = 2;
    Type = AttackType.Melee;
    Name = "Речной кроколист";
    Text = "";
    break;
case 28:
    DefaultPower = 3;
    Type = AttackType.Melee;
    Name = "Дровосек";
    Text = "";
    break;
case 29:
    DefaultPower = 1;
    Type = AttackType.Melee;
    Name = "Анимешник";
    Text = "";
    break;
case 30:
    DefaultPower = 7;
    Type = AttackType.Range_spy;
    Name = "Партизантка";
    Text = "";
    break;
case 31:
    DefaultPower = 8;
    Type = AttackType.Range_spy;
    Name = "Продажный репортер";
    Text = "Там платят больше";
    break;
case 32:
    DefaultPower = 9;
    Type = AttackType.Range_spy;
    Name = "Таргетная реклама";
    Text = "Нажми, чтобы узнать как!";
    break;
case 33:
    DefaultPower = 10;
    Type = AttackType.Range_spy;
    Name = "Windows";
    Text = "Помогите улучшить нашу ОС";
    break;
case 34:
    DefaultPower = 11;
    Type = AttackType.Range_spy;
    Name = "Вдова";
    Text = "В команде противника";
    break;
case 35:
    DefaultPower = 1;
    Type = AttackType.Range;
    Name = "Вдова";
    Text = "В твоей команде";
    break;
case 36:
    DefaultPower = 2;
    Type = AttackType.Range;
    Name = "Русский репер";
    Text = "-уши";
    break;
case 37:
    DefaultPower = 1;
    Type = AttackType.Range;
    Name = "Обезьяна с автоматом";
    Text = "";

```

```

        break;
case 38:
    DefaultPower = 2;
    Type = AttackType.Range;
    Name = "Голубь";
    Text = "";
    break;
case 39:
    DefaultPower = 3;
    Type = AttackType.Range;
    Name = "Девочка из техподдержки";
    Text = "Какая у вас ОС?";
    break;
case 40:
    DefaultPower = 9;
    Type = AttackType.Melee_spy;
    Name = "Радужный пони";
    Text = "Он только кажется милым";
    break;
case 41:
    DefaultPower = 10;
    Type = AttackType.Melee_spy;
    Name = "Огр-ниндзя";
    Text = "Два центнера незаметности";
    break;
case 42:
    DefaultPower = 9;
    Type = AttackType.Melee_spy;
    Name = "Гопник";
    Text = "Дай позвонить";
    break;
case 43:
    DefaultPower = 8;
    Type = AttackType.Melee_spy;
    Name = "Уборщица";
    Text = "Куда по помытому!";
    break;
case 44:
    DefaultPower = 7;
    Type = AttackType.Melee_spy;
    Name = "Бармен";
    Text = "Повторить?";
    break;
case 45:
    DefaultPower = 1;
    Type = AttackType.Melee_weather;
    Name = "Дождь";
    Text = "Все дороги размыло";
    break;
case 46:
    DefaultPower = 1;
    Type = AttackType.Range_weather;
    Name = "Снегопад";
    Text = "Видимость нулевая";
    break;
case 47:
    DefaultPower = 0;
    Type = AttackType.Clear_weather;
    Name = "Ясная погода";
    Text = "А солнце светит всем одинаково";
    break;
}

Power = DefaultPower;

```

```

byte[] imageBytes = null;
if (Type == AttackType.Melee)
{
    imageBytes = Convert.FromBase64String("...");
}
else if (Type == AttackType.Range)
{
    imageBytes = Convert.FromBase64String("...");
}
else if (Type == AttackType.Melee_spy)
{
    imageBytes = Convert.FromBase64String("...");
}
else if (Type == AttackType.Range_spy)
{
    imageBytes = Convert.FromBase64String("...");
}
else if (Type == AttackType.Melee_weather)
{
    imageBytes = Convert.FromBase64String("...");
}
else if (Type == AttackType.Range_weather)
{
    imageBytes = Convert.FromBase64String("...");
}
else if (Type == AttackType.Clear_weather)
{
    imageBytes = Convert.FromBase64String("...");
}
MemoryStream ms = new MemoryStream(imageBytes, 0, imageBytes.Length);
ms.Write(imageBytes, 0, imageBytes.Length);
Image image = Image.FromStream(ms, true);
picture.Image = image;

if (Rare.Contains(id))
{
    Rarity = RarityType.Rare;
}
else if (Epic.Contains(id))
{
    Rarity = RarityType.Epic;
}
else if (Legendary.Contains(id))
{
    Rarity = RarityType.Legendary;
}
else
{
    Rarity = RarityType.Common;
}
}

public int DefaultPower
{
    get
    {
        return defaultPower;
    }

    set
    {
        defaultPower = value;
    }
}

```

```

    }

    public int Power
    {
        get
        {
            return int.Parse(power.Text.ToString());
        }

        set
        {
            power.Text = value.ToString();
        }
    }

    public bool Selected
    {
        get
        {
            return selected;
        }

        set
        {
            selected = value;
            if (value)
            {
                name.BackColor = Color.MediumSpringGreen;
            }
            else
            {
                name.BackColor = Color.White;
            }
            OnSelectionChangedEvent(new EventArgs());
        }
    }

    public bool InHand
    {
        get
        {
            return inHand;
        }

        set
        {
            inHand = value;
        }
    }

    public new string Name
    {
        get
        {
            return name.Text;
        }

        set
        {
            name.Text = value;
        }
    }

    public new string Text

```

```

{
    get
    {
        return text.Text;
    }

    set
    {
        text.Text = value;
    }
}

public AttackType Type
{
    get
    {
        return type;
    }

    set
    {
        type = value;
    }
}

public int ID
{
    get
    {
        return id;
    }

    set
    {
        id = value;
    }
}

public RarityType Rarity
{
    get
    {
        return rarity;
    }

    set
    {
        if (value == RarityType.Common)
        {
            power.BackColor = Color.AliceBlue;
        }
        else if (value == RarityType.Rare)
        {
            power.BackColor = Color.DeepSkyBlue;
        }
        else if (value == RarityType.Epic)
        {
            power.BackColor = Color.DarkOrchid;
        }
        else if (value == RarityType.Legendary)
        {
            power.BackColor = Color.DarkOrange;
        }
        rarity = value;
    }
}

```

```

    }
}

public enum AttackType
{
    Null,
    Melee,
    Range,
    Melee_spy,
    Range_spy,
    Melee_weather,
    Range_weather,
    Clear_weather
}

public enum RarityType
{
    Common,
    Rare,
    Epic,
    Legendary
}

private int id = 0;

private AttackType type = AttackType.Null;

private RarityType rarity = RarityType.Common;

private bool selected = false;

private bool inHand = false;

private int defaultPower;

public delegate void SelectionChangedEventHandler(object sender, EventArgs e);

public event SelectionChangedEventHandler SelectionChangedEvent;

protected virtual void OnSelectionChangedEvent(EventArgs e)
{
    SelectionChangedEvent?.Invoke(this, e);
}

private void Card_Click(object sender, EventArgs e)
{
    (sender as Control).Enabled = false;
    if (((sender as Control).Parent as Card).InHand)
    {
        ((sender as Control).Parent as Card).Selected = !((sender as Control).Parent as Card).Selected;
    }
    (sender as Control).Enabled = true;
}
}
}

```

StartForm.cs

```

using System;
using System.Windows.Forms;

namespace ProjectA
{
    public partial class StartForm : Form

```

```

{
    public StartForm()
    {
        InitializeComponent();
    }

    private void startButton_Click(object sender, EventArgs e)
    {
        new DraftForm().Show();
        Hide();
    }

    private void StartForm_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.Escape)
        {
            Environment.Exit(0);
        }
    }
}
}

```

DraftForm.cs

```

using CardControl;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Windows.Forms;

namespace ProjectA
{
    public partial class DraftForm : Form
    {
        List<int> Deck = new List<int>();
        public DraftForm()
        {
            InitializeComponent();
            ChangeCards();
        }

        public List<int> TakeThree()
        {
            List<int> result = new List<int>();
            while (result.Count < 3)
            {
                if (Deck.Count < 4)
                {
                    int cardCount = Card.Common.Count + Card.Rare.Count + Card.Epic.Count + Card.Legendary.Count;
                    double r = new Random().NextDouble();
                    if (r < 0.5)
                    {
                        int i = new Random().Next(0, Card.Common.Count);
                        if (!result.Contains(Card.Common[i]))
                        {
                            result.Add(Card.Common[i]);
                        }
                    }
                    else if (r < 0.8)
                    {
                        int i = new Random().Next(0, Card.Rare.Count);

```

```

        if (!result.Contains(Card.Rare[i]))
        {
            result.Add(Card.Rare[i]);
        }
    }
    else if (r < 0.95)
    {
        int i = new Random().Next(0, Card.Epic.Count);
        if (!result.Contains(Card.Epic[i]))
        {
            result.Add(Card.Epic[i]);
        }
    }
    else
    {
        int i = new Random().Next(0, Card.Legendary.Count);
        if (!result.Contains(Card.Legendary[i]))
        {
            result.Add(Card.Legendary[i]);
        }
    }
}
else if (Deck.Count < 7)
{
    int cardCount = Card.Rare.Count + Card.Epic.Count + Card.Legendary.Count;
    double r = new Random().NextDouble();
    if (r < 0.6)
    {
        int i = new Random().Next(0, Card.Rare.Count);
        if (!result.Contains(Card.Rare[i]))
        {
            result.Add(Card.Rare[i]);
        }
    }
    else if (r < 0.9)
    {
        int i = new Random().Next(0, Card.Epic.Count);
        if (!result.Contains(Card.Epic[i]))
        {
            result.Add(Card.Epic[i]);
        }
    }
    else
    {
        int i = new Random().Next(0, Card.Legendary.Count);
        if (!result.Contains(Card.Legendary[i]))
        {
            result.Add(Card.Legendary[i]);
        }
    }
}
else
{
    int cardCount = Card.Epic.Count + Card.Legendary.Count;
    double r = new Random().NextDouble();
    if (r < 0.75)
    {
        int i = new Random().Next(0, Card.Epic.Count);
        if (!result.Contains(Card.Epic[i]))
        {
            result.Add(Card.Epic[i]);
        }
    }
    else

```



```

        {
            int i = new Random().Next(0, Card.Legendary.Count);
            if (!result.Contains(Card.Legendary[i]))
            {
                result.Add(Card.Legendary[i]);
            }
        }
    }
}
return result;
}
public void Card_SelectionChanged(object sender, EventArgs e)
{
    Card card = sender as Card;
    if (card.Selected)
    {
        foreach (Card c in cardsPanel.Controls)
        {
            if (c.Selected && (c != card))
            {
                c.Selected = false;
            }
        }
    }
}

public void ChangeCards()
{
    List<int> cards = TakeThree();

    Card card1 = new Card(cards[0], true);
    Card card2 = new Card(cards[1], true);
    Card card3 = new Card(cards[2], true);

    card1.SelectionChangedEvent += Card_SelectionChanged;
    card2.SelectionChangedEvent += Card_SelectionChanged;
    card3.SelectionChangedEvent += Card_SelectionChanged;

    card1.Location = new Point(148, 0);
    card2.Location = new Point(349, 0);
    card3.Location = new Point(550, 0);

    cardsPanel.Controls.Clear();

    cardsPanel.Controls.Add(card1);
    cardsPanel.Controls.Add(card2);
    cardsPanel.Controls.Add(card3);
}
private void DraftForm_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Escape)
    {
        Environment.Exit(0);
    }
}
private void takeButton_Click(object sender, EventArgs e)
{
    Card selectedCard = new Card();
    foreach (Card card in cardsPanel.Controls)
    {
        if (card.Selected)
        {
            selectedCard = card;
        }
    }
}

```

```

    }
    if (selectedCard.ID == 0)
    {
        MessageBox.Show("Выберите карту!");
    }
    else
    {
        Deck.Add(selectedCard.ID);
        if (Deck.Count == 8)
        {
            TcpClient server = new TcpClient();
            try
            {
                server.Connect("localhost", 44444);
            }
            catch
            {
                MessageBox.Show("Не удалось подключиться к серверу");
                Environment.Exit(1);
            }
            string msg = "";
            foreach (int card in Deck)
            {
                msg += card + ", " + new Card(card, false).Power + ":";
            }
            if (msg != "")
            {
                msg = msg.Substring(0, msg.Length - 1);
            }
            Send(server, msg);
            Hide();
            Form wait = new WaitForm();
            wait.Show();
            new Thread(new ThreadStart(new Action(() =>
            {
                msg = Receive(server);
                Invoke(new Action(() =>
                {
                    wait.Close();
                    new GameForm(server, msg).Show();
                }));
            }))).Start();
        }
        else
        {
            ChangeCards();
        }
    }
}

static void Send(TcpClient receiver, string message)
{
    message += "?";
    try
    {
        NetworkStream stream = receiver.GetStream();
        byte[] buff = Encoding.ASCII.GetBytes(message);
        stream.Write(BitConverter.GetBytes(buff.Length), 0, 4);
        stream.Write(buff, 0, message.Length);
    }
    catch
    {
        MessageBox.Show("Не удалось передать данные на сервер.");
        Environment.Exit(2);
    }
}

```

```

    }

    static string Receive(TcpClient sender)
    {
        byte[] buff = new byte[4];
        try
        {
            NetworkStream stream = sender.GetStream();
            stream.Read(buff, 0, 4);
            int length = BitConverter.ToInt32(buff, 0);
            buff = new byte[length];
            stream.Read(buff, 0, length);
        }
        catch
        {
            MessageBox.Show("Соединение с сервером потеряно.");
            Environment.Exit(3);
        }
        string msg = Encoding.ASCII.GetString(buff);
        msg = msg.Substring(0, msg.Length - 1);
        return msg;
    }
}

```

WaitForm.cs

```

using System;
using System.Windows.Forms;

namespace ProjectA
{
    public partial class WaitForm : Form
    {
        public WaitForm()
        {
            InitializeComponent();
        }

        private void WaitForm_Load(object sender, EventArgs e)
        {
            progressBar.Step = 1;
        }

        private void WaitForm_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Escape)
            {
                Environment.Exit(0);
            }
        }
    }
}

```

GameForm.cs

```

using System;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using CardControl;
using System.Net.Sockets;
using System.Threading;
using System.IO;

```

```

namespace ProjectA
{
    public partial class GameForm : Form
    {
        static void Send(TcpClient receiver, string message)
        {
            message += "?";
            try
            {
                NetworkStream stream = receiver.GetStream();
                byte[] buff = Encoding.ASCII.GetBytes(message);
                stream.Write(BitConverter.GetBytes(buff.Length), 0, 4);
                stream.Write(buff, 0, message.Length);
            }
            catch
            {
                MessageBox.Show("Не удалось передать данные на сервер.");
                Environment.Exit(2);
            }
        }

        static string Receive(TcpClient sender)
        {
            byte[] buff = new byte[4];
            try
            {
                NetworkStream stream = sender.GetStream();
                stream.Read(buff, 0, 4);
                int length = BitConverter.ToInt32(buff, 0);
                buff = new byte[length];
                stream.Read(buff, 0, length);
            }
            catch
            {
                MessageBox.Show("Соединение с сервером потеряно.");
                Environment.Exit(3);
            }
            string msg = Encoding.ASCII.GetString(buff);
            msg = msg.Substring(0, msg.Length - 1);
            return msg;
        }

        TcpClient server;
        string handString = "";
        public GameForm(TcpClient srv, string h)
        {
            server = srv;
            handString = h;
            InitializeComponent();
            Visible = false;
            byte[] imageBytes = Convert.FromBase64String("...");
            MemoryStream ms = new MemoryStream(imageBytes, 0, imageBytes.Length);
            ms.Write(imageBytes, 0, imageBytes.Length);
            Image image = Image.FromStream(ms, true);
            yourRound.Image = image;
            enemyRound.Image = image;
        }

        private void GameForm_Load(object sender, EventArgs e)
        {
            if (handString != "")
            {
                string[] s = handString.Split(':');
            }
        }
    }
}

```

```

        foreach (string c in s)
        {
            string[] card = c.Split(',');
            Card nc = new Card(int.Parse(card[0]), true);
            nc.Power = int.Parse(card[1]);
            hand.Controls.Add(nc);
        }
        foreach (Card c in hand.Controls)
        {
            c.SelectionChangedEvent += Card_SelectionChanged;
        }
    }
    Redraw();
    Visible = true;
    new Thread(new ThreadStart(Game)).Start();
}

```

```

public void UnpackLine(Control line, string msg, bool inhand)
{
    if (msg != "")
    {
        Invoke(new Action(() =>
        {
            line.Controls.Clear();
            string[] s = msg.Split(':');
            foreach (string c in s)
            {
                string[] card = c.Split(',');
                Card nc = new Card(int.Parse(card[0]), inhand);
                nc.Power = int.Parse(card[1]);
                line.Controls.Add(nc);
            }
        }));
    }
    else
    {
        Invoke(new Action(() =>
        {
            line.Controls.Clear();
        }));
    }
}

```

```

public void ClearBoard()
{
    enemyRange.Controls.Clear();
    enemyMelee.Controls.Clear();
    yourMelee.Controls.Clear();
    yourRange.Controls.Clear();
}
public void Game()
{
    bool eog = false;
    while (!eog)
    {
        string msg = Receive(server);
        if (msg == "game")
        {
            msg = Receive(server);
            UnpackLine(enemyRange, msg, false);

            msg = Receive(server);
            UnpackLine(enemyMelee, msg, false);
        }
    }
}

```

```

msg = Receive(server);
UnpackLine(yourMelee, msg, false);

msg = Receive(server);
UnpackLine(yourRange, msg, false);

msg = Receive(server);
UnpackLine(hand, msg, true);
Invoke(new Action(() =>
{
    foreach (Card c in hand.Controls)
    {
        c.SelectionChangedEvent += Card_SelectionChanged;
    }
    playCard.Enabled = true;
}));
}
else if (msg == "spy")
{
    msg = Receive(server);
    Invoke(new Action(() =>
    {
        if (msg != "")
        {
            string[] card = msg.Split(',');
            Card nc = new Card(int.Parse(card[0]), true);
            nc.Power = int.Parse(card[1]);
            hand.Controls.Add(nc);
        }
    }));
    new Thread(new ThreadStart(SendLines)).Start();
}
else if (msg == "score")
{
    Invoke(new Action(() =>
    {
        playCard.Enabled = false;
    }));
    Send(server, yourScore.Text);
}
else if (msg == "round_victory")
{
    Invoke(new Action(() =>
    {
        yourRound.Visible = true;
        MessageBox.Show("Round Victory");
        playCard.Enabled = false;
    }));
    ClearBoard();
}
else if (msg == "round_draw")
{
    Invoke(new Action(() =>
    {
        yourRound.Visible = true;
        enemyRound.Visible = true;
        MessageBox.Show("Round Draw");
        playCard.Enabled = false;
    }));
    ClearBoard();
}
else if (msg == "round_defeat")
{
    Invoke(new Action(() =>

```

```

        {
            enemyRound.Visible = true;
            MessageBox.Show("Round Defeat");
            playCard.Enabled = false;
        });
        ClearBoard();
    }
    else if (msg == "victory")
    {
        Invoke(new Action(() =>
        {
            MessageBox.Show("VICTORY");
            playCard.Enabled = false;
        }));
        eog = true;
    }
    else if (msg == "draw")
    {
        Invoke(new Action(() =>
        {
            MessageBox.Show("DRAW");
            playCard.Enabled = false;
        }));
        eog = true;
    }
    else if (msg == "defeat")
    {
        Invoke(new Action(() =>
        {
            MessageBox.Show("DEFEAT");
            playCard.Enabled = false;
        }));
        eog = true;
    }
    else if (msg == "error")
    {
        Invoke(new Action(() =>
        {
            MessageBox.Show("Ваш противник вышел.");
            playCard.Enabled = false;
        }));
        Environment.Exit(0);
    }
    Invoke(new Action(() =>
    {
        Redraw();
    }));
}
Environment.Exit(0);
}
public void DrawLine(GroupBox gr)
{
    if (gr.Controls.Count == 0)
    {
        gr.Text = "0";
    }
    else
    {
        int p = 0;
        int x = -106;
        int y = 16;
        foreach (Control c in gr.Controls)
        {
            p += (c as Card).Power;

```

```

        x += 112;
        c.Location = new Point(x, y);
    }
    gr.Text = p.ToString();
}
}

public void Redraw()
{
    DrawLine(enemyRange);
    DrawLine(enemyMelee);
    DrawLine(yourMelee);
    DrawLine(yourRange);
    DrawLine(hand);
    enemyScore.Text = (int.Parse(enemyRange.Text) + int.Parse(enemyMelee.Text)).ToString();
    yourScore.Text = (int.Parse(yourRange.Text) + int.Parse(yourMelee.Text)).ToString();
}

private void playCard_Click(object sender, EventArgs e)
{
    bool pass = true;
    bool spy = false;
    playCard.Enabled = false;
    foreach (Control card in hand.Controls)
    {
        Card c = card as Card;
        if (c.Selected)
        {
            c.Selected = false;
            hand.Controls.Remove(c);
            if (c.Type == Card.AttackType.Melee)
            {
                yourMelee.Controls.Add(c);
            }
            else if (c.Type == Card.AttackType.Range)
            {
                yourRange.Controls.Add(c);
            }
            else if (c.Type == Card.AttackType.Melee_spy)
            {
                enemyMelee.Controls.Add(c);
                Send(server, "spy");
                spy = true;
            }
            else if (c.Type == Card.AttackType.Range_spy)
            {
                enemyRange.Controls.Add(c);
                Send(server, "spy");
                spy = true;
            }
            else if (c.Type == Card.AttackType.Melee_weather)
            {
                foreach (Card ac in yourMelee.Controls)
                {
                    ac.Power = c.Power;
                }
                foreach (Card ac in enemyMelee.Controls)
                {
                    ac.Power = c.Power;
                }
            }
            else if (c.Type == Card.AttackType.Range_weather)
            {
                foreach (Card ac in yourRange.Controls)
                {

```



```

        ac.Power = c.Power;
    }
    foreach (Card ac in enemyRange.Controls)
    {
        ac.Power = c.Power;
    }
}
else if (c.Type == Card.AttackType.Clear_weather)
{
    foreach (Card ac in yourMelee.Controls)
    {
        ac.Power = ac.DefaultPower;
    }
    foreach (Card ac in enemyMelee.Controls)
    {
        ac.Power = ac.DefaultPower;
    }
    foreach (Card ac in yourRange.Controls)
    {
        ac.Power = ac.DefaultPower;
    }
    foreach (Card ac in enemyRange.Controls)
    {
        ac.Power = ac.DefaultPower;
    }
}
c.InHand = false;
pass = false;
break;
}
}
Redraw();
if (pass)
{
    Send(server, "pass");
}
else
{
    if (!spy)
    {
        new Thread(new ThreadStart(SendLines)).Start();
    }
}
}

public void SendLines()
{
    Send(server, "game");
    SendLine(enemyRange.Controls);
    SendLine(enemyMelee.Controls);
    SendLine(yourMelee.Controls);
    SendLine(yourRange.Controls);
    SendLine(hand.Controls);
}

public void SendLine(Control.ControlCollection controls)
{
    string msg = "";
    foreach (Control c in controls)
    {
        Card card = c as Card;
        msg += card.ID + "," + card.Power + ":";
    }
    if (msg != "")
    {

```

```

        msg = msg.Substring(0, msg.Length - 1);
    }
    Send(server, msg);
}
public void Card_SelectionChanged(object sender, EventArgs e)
{
    Card c = sender as Card;
    if (c.Selected)
    {
        foreach (Card hc in hand.Controls)
        {
            if (hc.Selected && (hc != c))
            {
                hc.Selected = false;
            }
        }
    }
}

private void GameForm_SizeChanged(object sender, EventArgs e)
{
    enemyRange.Width = Width - enemyRange.Location.X * 2;
    enemyMelee.Width = Width - enemyMelee.Location.X * 2;
    yourRange.Width = Width - yourRange.Location.X * 2;
    yourMelee.Width = Width - yourMelee.Location.X * 2;
    hand.Width = Width - hand.Location.X * 2;
    playCard.Location = new Point(Width / 2 - playCard.Width / 2, yourRange.Location.Y + yourRange.Height +
(hand.Location.Y - (yourRange.Location.Y + yourRange.Height)) / 2 - playCard.Height / 2);
}

private void GameForm_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Escape)
    {
        Environment.Exit(0);
    }
}
}
}

```

Сервер (Program.cs)

```

using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using NLog;
using NLog.Targets;
using NLog.Config;

namespace ProjectA_Server
{
    class Program
    {
        class Player
        {
            public TcpClient Client;
            public string Hand = "";
            public string Melee = "";
            public string Range = "";

```

```

    public bool Pass = false;
    public string Score = "";
    public int Rounds = 0;
}
static void Send(TcpClient receiver, string message)
{
    message += "?";
    try
    {
        NetworkStream stream = receiver.GetStream();
        byte[] buff = Encoding.ASCII.GetBytes(message);
        stream.Write(BitConverter.GetBytes(buff.Length), 0, 4);
        stream.Write(buff, 0, message.Length);
    }
    catch
    {
        Logger logger = LogManager.GetLogger("Logger" + Thread.CurrentThread.Name);
        logger.Info("Не удалось передать данные клиенту. Завершаю игру.");
        RemoveClients(receiver, true);
        Thread.CurrentThread.Abort();
    }
}

static string Receive(TcpClient sender)
{
    byte[] buff = new byte[4];
    try
    {
        NetworkStream stream = sender.GetStream();
        stream.Read(buff, 0, 4);
        int length = BitConverter.ToInt32(buff, 0);
        buff = new byte[length];
        stream.Read(buff, 0, length);
    }
    catch
    {
        Logger logger = LogManager.GetLogger("Logger" + Thread.CurrentThread.Name);
        logger.Info("Не удалось получить данные. Завершаю игру.");
        RemoveClients(sender, true);
        Thread.CurrentThread.Abort();
    }
    string msg = Encoding.ASCII.GetString(buff);
    msg = msg.Substring(0, msg.Length - 1);
    return msg;
}

static void RemoveClients(TcpClient client, bool withError)
{
    foreach (Tuple<TcpClient, TcpClient> pair in clients)
    {
        if (pair.Item1 == client)
        {
            clients.Remove(pair);
            if (withError)
            {
                Send(pair.Item2, "error");
                Logger logger = LogManager.GetLogger("Logger" + Thread.CurrentThread.Name);
                logger.Info("Послал ошибку.");
                pair.Item2.Close();
            }
        }
        else
        {
            pair.Item1.Close();
            pair.Item2.Close();
        }
    }
}

```

```

        }
        break;
    }
    else if (pair.Item2 == client)
    {
        clients.Remove(pair);
        if (withError)
        {
            Send(pair.Item1, "error");
            Logger logger = LogManager.GetLogger("Logger" + Thread.CurrentThread.Name);
            logger.Info("Послал ошибку.");
            pair.Item1.Close();
        }
        else
        {
            pair.Item1.Close();
            pair.Item2.Close();
        }
        break;
    }
}
}

static List<Tuple<TcpClient, TcpClient>> clients = new List<Tuple<TcpClient, TcpClient>>();

static int gameCount = 0;

static void Main(string[] args)
{
    List<TcpClient> list = new List<TcpClient>();
    TcpListener server = new TcpListener(IPAddress.Any, 44444);
    server.Start();
    TcpClient connection;
    while (true)
    {
        connection = server.AcceptTcpClient();
        list.Add(connection);
        int i = 0;
        while (i < list.Count)
        {
            try
            {
                list[i].GetStream().Write(new byte[1], 0, 0);
                i++;
            }
            catch
            {
                list.RemoveAt(i);
            }
        }
        if (list.Count > 1)
        {
            TcpClient first = list[0];
            TcpClient second = list[1];
            list.RemoveRange(0, 2);
            Tuple<TcpClient, TcpClient> pair = new Tuple<TcpClient, TcpClient>(first, second);
            clients.Add(pair);
            gameCount++;
            Tuple<TcpClient, TcpClient, int> parameters = new Tuple<TcpClient, TcpClient, int>(first, second,
gameCount);
            new Thread(new ParameterizedThreadStart(Game)).Start(parameters);
        }
    }
}

```

```

    }

    static void Game(object o)
    {
        int game = (o as Tuple<TcpClient, TcpClient, int>).Item3;
        LoggingConfiguration config = new LoggingConfiguration();
        FileTarget fileTarget = new FileTarget();
        fileTarget.FileName = "log_" + DateTime.Now.ToString("yyyy.MM.dd_HH.mm.ss") + "_game_" + game +
".txt";
        fileTarget.Layout = "${message}";
        fileTarget.DeleteOldFileOnStartup = true;
        LoggingRule rule = new LoggingRule("", LogLevel.Info, fileTarget);
        config.LoggingRules.Add(rule);
        LogManager.Configuration = config;
        Logger logger = LogManager.GetLogger("Logger" + game);
        Thread.CurrentThread.Name = game.ToString();

        bool eog = false;
        Player player1 = new Player();
        player1.Client = (o as Tuple<TcpClient, TcpClient, int>).Item1;
        Player player2 = new Player();
        player2.Client = (o as Tuple<TcpClient, TcpClient, int>).Item2;
        player1.Hand = Receive(player1.Client);
        logger.Info("Получил первую руку (" + player1.Hand + ")");
        player2.Hand = Receive(player2.Client);
        logger.Info("Получил вторую руку (" + player2.Hand + ")");
        Player activePlayer = player1;
        Player passivePlayer = player2;
        Send(activePlayer.Client, activePlayer.Hand);
        logger.Info("Послал первую руку (" + activePlayer.Hand + ")");
        Send(passivePlayer.Client, passivePlayer.Hand);
        logger.Info("Послал вторую руку (" + passivePlayer.Hand + ")");
        while ((activePlayer.Rounds != 2) && (passivePlayer.Rounds != 2))
        {
            while (!eog)
            {
                Send(activePlayer.Client, "game");
                logger.Info("Посылаю игру");
                Send(activePlayer.Client, passivePlayer.Range);
                logger.Info("Послал вражей ренжей (" + passivePlayer.Range + ")");
                Send(activePlayer.Client, passivePlayer.Melee);
                logger.Info("Послал вражей мили (" + passivePlayer.Melee + ")");
                Send(activePlayer.Client, activePlayer.Melee);
                logger.Info("Послал твоих мили (" + activePlayer.Melee + ")");
                Send(activePlayer.Client, activePlayer.Range);
                logger.Info("Послал твоих ренжей (" + activePlayer.Range + ")");
                Send(activePlayer.Client, activePlayer.Hand);
                logger.Info("Послал руку (" + activePlayer.Hand + ")");
                string msg = Receive(activePlayer.Client);
                if (msg == "spy")
                {
                    logger.Info("Получил запрос на карту со шпиона");
                    string card = "";
                    if (passivePlayer.Hand.Length != 0)
                    {
                        string[] cards = passivePlayer.Hand.Split(':');
                        card = cards[new Random().Next(0, cards.Length)];
                    }
                    Send(activePlayer.Client, "spy");
                    Send(activePlayer.Client, card);
                    logger.Info("Послал карту с шпиона (" + card + ")");
                    msg = Receive(activePlayer.Client);
                }
                if (msg == "game")
            }
        }
    }

```

```

{
    logger.Info("Получаю игру от активного");
    passivePlayer.Range = Receive(activePlayer.Client);
    logger.Info("Получил вражей ренжей (" + passivePlayer.Range + ")");
    passivePlayer.Melee = Receive(activePlayer.Client);
    logger.Info("Получил вражей мили (" + passivePlayer.Melee + ")");
    activePlayer.Melee = Receive(activePlayer.Client);
    logger.Info("Получил твоих мили (" + activePlayer.Melee + ")");
    activePlayer.Range = Receive(activePlayer.Client);
    logger.Info("Получил твоих ренжей (" + activePlayer.Range + ")");
    activePlayer.Hand = Receive(activePlayer.Client);
    logger.Info("Получил руку (" + activePlayer.Hand + ")");
    if (activePlayer.Hand == "")
    {
        activePlayer.Pass = true;
        logger.Info("У игрока кончились карты - пас");
    }
}
else if (msg == "pass")
{
    activePlayer.Pass = true;
    logger.Info("Игрок спасовал");
}
if (!passivePlayer.Pass)
{
    if (activePlayer == player1)
    {
        activePlayer = player2;
        passivePlayer = player1;
    }
    else
    {
        activePlayer = player1;
        passivePlayer = player2;
    }
    logger.Info("Смена активного игрока.");
}
if (activePlayer.Pass && passivePlayer.Pass)
{
    eog = true;
    logger.Info("Конец раунда");
}
}

Send(passivePlayer.Client, "game");
logger.Info("Посылаю игру пассивному игроку");
Send(passivePlayer.Client, activePlayer.Range);
logger.Info("Послал вражей ренжей (" + activePlayer.Range + ")");
Send(passivePlayer.Client, activePlayer.Melee);
logger.Info("Послал вражей мили (" + activePlayer.Melee + ")");
Send(passivePlayer.Client, passivePlayer.Melee);
logger.Info("Послал твоих мили (" + passivePlayer.Melee + ")");
Send(passivePlayer.Client, passivePlayer.Range);
logger.Info("Послал твоих ренжей (" + passivePlayer.Range + ")");
Send(passivePlayer.Client, passivePlayer.Hand);
logger.Info("Послал руку (" + passivePlayer.Hand + ")");

Send(activePlayer.Client, "score");
logger.Info("Запросил счет активного");
activePlayer.Score = Receive(activePlayer.Client);
logger.Info("Получил счет активного (" + activePlayer.Score + ")");
Send(passivePlayer.Client, "score");
logger.Info("Запросил счет пассивного");
passivePlayer.Score = Receive(passivePlayer.Client);
logger.Info("Получил счет пассивного (" + passivePlayer.Score + ")");

```

```

if (activePlayer.Score == passivePlayer.Score)
{
    Send(activePlayer.Client, "round_draw");
    activePlayer.Rounds++;
    Send(passivePlayer.Client, "round_draw");
    passivePlayer.Rounds++;
    logger.Info("Послал ничью в раунде обоим");
}
else
{
    if (int.Parse(activePlayer.Score) > int.Parse(passivePlayer.Score))
    {
        Send(activePlayer.Client, "round_victory");
        logger.Info("Послал победу в раунде активному");
        activePlayer.Rounds++;
        Send(passivePlayer.Client, "round_defeat");
        logger.Info("Послал поражение в раунде пассивному");
    }
    else
    {
        Send(passivePlayer.Client, "round_victory");
        logger.Info("Послал победу в раунде пассивному");
        passivePlayer.Rounds++;
        Send(activePlayer.Client, "round_defeat");
        logger.Info("Послал поражение в раунде активному");
    }
}
activePlayer.Melee = "";
activePlayer.Range = "";
passivePlayer.Melee = "";
passivePlayer.Range = "";
activePlayer.Pass = false;
passivePlayer.Pass = false;
eog = false;
logger.Info("Сбросил борду");
if (activePlayer == player1)
{
    activePlayer = player2;
    passivePlayer = player1;
}
else
{
    activePlayer = player1;
    passivePlayer = player2;
}
logger.Info("Сменил активного игрока");
}
logger.Info("Конец игры");
if (activePlayer.Rounds == passivePlayer.Rounds)
{
    Send(activePlayer.Client, "draw");
    Send(passivePlayer.Client, "draw");
    logger.Info("Послал ничью обоим");
}
else
{
    if (activePlayer.Rounds == 2)
    {
        Send(activePlayer.Client, "victory");
        logger.Info("Послал победу активному");
        Send(passivePlayer.Client, "defeat");
        logger.Info("Послал поражение пассивному");
    }
    else

```

```
        {
            Send(passivePlayer.Client, "victory");
            logger.Info("Послал победу пассивному");
            Send(activePlayer.Client, "defeat");
            logger.Info("Послал поражение активному");
        }
    }
    RemoveClients(player1.Client, false);
}
}
```