Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Операционные Системы и Системное Программирование (ОСиСП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

Игровое приложение «Xonix»

БГУИР КП 1-40 01 01 1 ПЗ

Студент: гр. 751003 Гринчик В.В.

Проверил: Базылев Е.Н.

Минск 2019

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2019г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Гринчику Всеволоду Владимировичу* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Игровое приложение «Xonix»* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи законченной работы *15.12.2019г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Язык программирования C++. Среда разработки Microsoft Visual Studio 2019. Реализация игрового приложения “Xonix” с использованием набора базовых функции интерфейсов программирования приложений операционных систем семейства Microsoft Windows(Windows API). Реализовать графический пользовательский интерфейс. Реализовать управление процессом игры с использованием функций обработки аппаратных сообщений интерфейса Windows API.*\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Руководство по установке и использованию программного средства\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Базылев Е.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *15.09.2019г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования

(с обозначением сроков выполнения и процентом от общего обьема работы):

*Раздел 1. Введение к 20.09.2018г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 15.10.2018г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 15.11.2018г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 10.12.2018г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5.Заключениe к 10.12.2018г. – 90% готовности работы; оформление\_ пояснительной записки и графического материала к 15.12.2018г. –100%\_готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 10.12.2018г. по 15.12.2018г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* *Базылев Е.Н.*

*(подпись)*

Задание принял к исполнению *Гринчик В.В. 15.12.2019г.*

*(дата и подпись студента)*

Содержание

[Введение 5](#_Toc25514773)

[1 Анализ литературных источников 6](#_Toc25514774)

[1.1 Анализ существующих приложений 6](#_Toc25514775)

[2 моделирование предметной области 9](#_Toc25514776)

[2.1 Цель 9](#_Toc25514777)

[2.2 Задачи 9](#_Toc25514778)

[2.3 Обоснование выбора языка программирования 9](#_Toc25514779)

[3 Разработка программного средства 10](#_Toc25514780)

[3.1 Визуальное представление приложения 10](#_Toc25514781)

[3. 2 Схема алгоритмов решения задачи по ГОСТ 19.701-90 14](#_Toc25514782)

[4 Руководство по установке и использованию программного средства 19](#_Toc25514783)

[4.1 Начало работы 19](#_Toc25514784)

[Заключение 21](#_Toc25514785)

[Список использованной литературы 22](#_Toc25514786)

[5 ПРИЛОЖЕНИЕ 23](#_Toc25514787)

Введение

Компьютерная игра — компьютерная программа, служащая для организации игрового процесса (геймплея), обучения, ускорения психологического развития, а также для создания неформальных или же на оборот особо формальных(ролеплей) связей с партнёрами по игре.

В наше время сложно найти человека, который бы ни разу не играл в компьютерные игры, и с каждым днём становится всё больше так называемых геймеров - людей которые постоянно играют в игры, и даже зарабатывают на этом деньги. Будь то новейшая игра на современной игровой приставке или старая аркада на игровом автомате или эмуляторе. По некоторым играм, относящиеся к так называемой не стареющей классике проводятся ежегодные турниры. В популярные игры может одновременно играть более миллиона человек. Однако чаще всего подобные турниры ассоциируются с такими играми, как Pac-Man, Tetris, Pong. Но существует огромное число игр которые хотя и не имеют столь громких имен, однако также представлены на различных турнирах это BreakOut, Qix, Xonix, и т. д..

Однако чтобы вспомнить игры из детства вовсе не обязательно идти в зал игровых автоматов или устанавливать эмулятор. Многие игры старые игры воссоздаются на современных платформах и не смотря на свою “старость” зачастую пользуются успехом. И именно на основание утверждений описанных выше я принял решение поучаствовать в процессе воссоздания незаслуженно забытых игр разработать версию игры “Xonix” которая бы работала на современных представителях операционных систем семейства Microsoft Windows .

# Анализ литературных источников

## Анализ существующих приложений

Итак первоначально игра Xonix была создана в 1984 году для платформы [PC](https://ru.wikipedia.org/wiki/IBM_PC-%D1%81%D0%BE%D0%B2%D0%BC%D0%B5%D1%81%D1%82%D0%B8%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) как клон игры [Qix](https://ru.wikipedia.org/wiki/Qix), появившейся ранее на [аркадных автоматах](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BA%D0%B0%D0%B4%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82). Разработана Иланом Рабом и Дэни Катцем.

Игровое поле представляет собой сетку из квадратных или прямоугольных ячеек, поэтому игра легко реализуется в текстовом режиме экрана. Ячейки могут быть двух типов: условно «суша» и «море». По полю движутся управляемый игроком курсор и управляемые программой точки. Курсор может двигаться по вертикали и горизонтали, точки — по диагонали.

Курсор может свободно передвигаться по «суше. Выходя в «море», он оставляет за собой след, уязвимый для «морских» точек, при этом в «море» нельзя изменить направление движения курсора на противоположное, можно поворачивать только на 90 градусов в любую сторону, также в «море» нельзя останавливаться. Игра считается проигранной, если курсор точка, если след курсора пересечет «морская» точка или при попытке в «море» изменить направление движение на противоположное или остановиться. Как только курсор снова оказывается на «суше», след его превращается в новую «сушу». Если при этом в «море» появилась замкнутая область, не содержащая точек, то вся эта область также превращается в «сушу».

Проще говоря, игрок, управляя курсором, пытается отсечь куски «моря», превращая их в «сушу», а точки ему в этом мешают.

К современным аналогам можно отнести такие игры как:

**1.1.1 AirXonix** – популярный 3D ремейк Xonix.

Рисунок 1.1– Внешний вид AirXonix

Цель игрока в AirXonix - используя некое летающее устройство "заполнить" как можно большую часть игрового поля, избегая столкновения с находящимися на нем шарами.

**Плюсы:**

3D графика;

Размер программы;

Несколько уровней сложности.

**Минусы:**

* Старость (игра не обновлялась с 1999 года);
* Множество отступлений от классической версии Xonix;
* AirXonix – распространяется на коммерческой основе.

В целом, я считаю, что это лучшая модификация, созданная на основе каноничной игры, но из-за большого числа изменений её уже нельзя назвать тем самым Xonix.

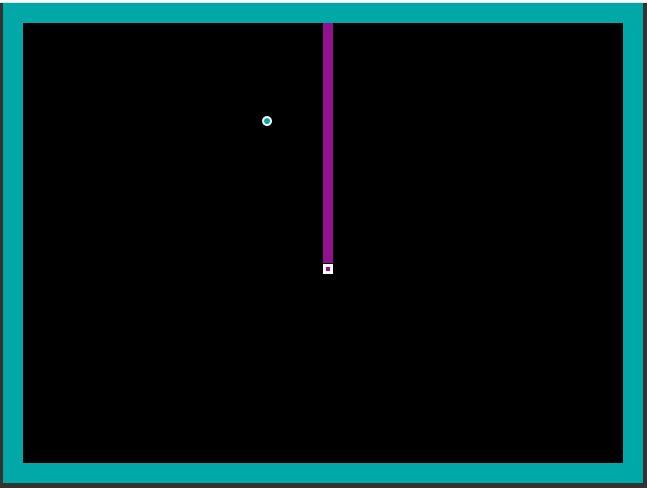
**1.1.2 xonix.pixel-tyumen** – онлайн версия Xonix.

Рисунок 1.4.2 – Внешний вид xonix.pixel-tyumen

Является полностью классической версией Xonix перенесенной на JavaScript.

**Плюсы:**

Доступность;

каноничность.

**Минусы:**

* Необходимость подключения к интернету;

Наиболее удачный каноничный ремейк Xonix. Главным недостатком которого я считаю отсутствие возможность скачивания приложения.

# моделирование предметной области

## Цель

Целью данного курсового проекта является разработка игрового программного средства «Xonix».

В процессе реализации программного средства планируется разработать игру с качественным графическим отображением процесса игры и корректной передачей команд игрока реализованных через интерфейс операционной системы Windows.

## Задачи

На основе цели работы было решено уделить особое внимание корректности передачи данных и внутренней логике программного средства.

Назначение игрового приложения заключается в возможности использования его как развлечения игроками разного уровня в связи, с чем пользователю доступны следующие возможности:

* возможность изменения скорости противников;
* возможность изменения числа противников.

## Обоснование выбора языка программирования

Для разработки программного средства использован язык программирования C++. Среда для разработки – Visual Studio 2019. Использование данной среды разработки дает возможность создания консольных и оконных приложений Windows. Выбор языка связан со следующими его преимуществами:

* язык программирования C++ претендует на объектную ориентированность
* возможность использования библиотеки windows;
* возможность взаимодействия с Windows API;
* унифицированная система типизации.

# Разработка программного средства

## Визуальное представление приложения

Главный вид программы имеет вид, представленный на рисунке 3.1 и требует предоставить номер порта а также login в приложении для сохранения результатов партии. Также имеется возможность использовать не уникальное имя



Рисунок 3.1 – Главное меню программного продукта

При подключении к серверу необходимо дождаться накопления определенного числа игроков. По окончании ожидания измениться набор кнопок для взаимодействия с приложением рисунок 3.2.



Рисунок 3.2 – Игровая комната

Игроки ходят в порядке подключения к серверу

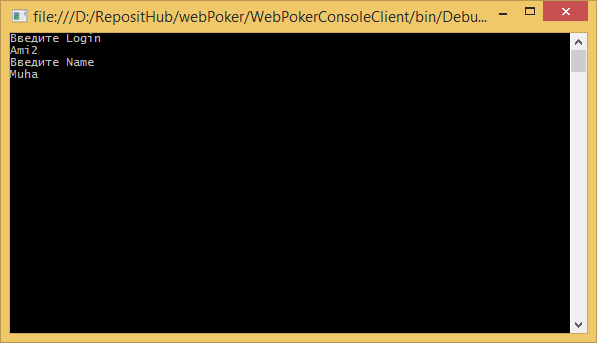
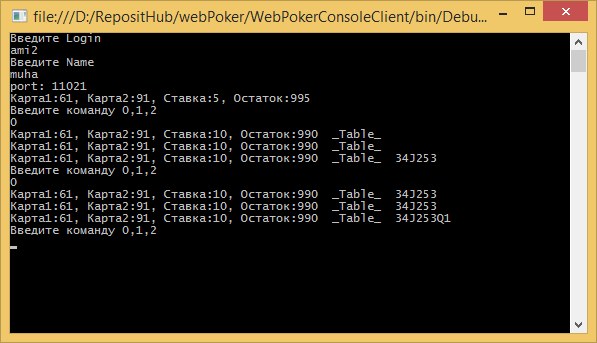


Рисунок 3.4 – Консольный интерфейс

По окончании игры объявляется победитель рисунки 3.5 3.6.



Рисунок 3.5 – Победитель

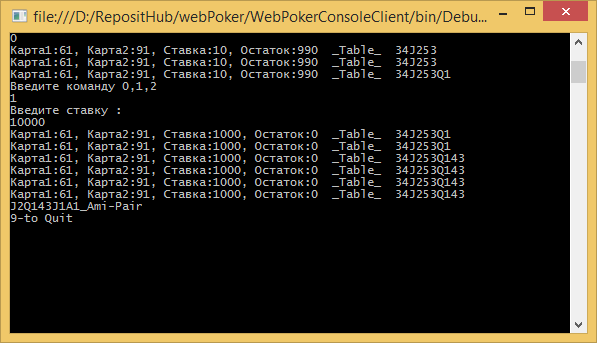


Рисунок 3.6 – Победитель консоль

## 3. 2 Схема алгоритмов решения задачи по ГОСТ 19.701-90



Рисунок 3.7 – Схема работы сервера



Рисунок 3.8 – Схема алгоритма загрузки пользователей(часть1)



Рисунок 3.9 – Схема алгоритма загрузки пользователей(часть2)



Рисунок 3.10 – Схема алгоритма начала игры (часть 1)



Рисунок 3.11 – Схема алгоритма начала игры (часть 2)

# Руководство по установке и использованию программного средства

4.1 Начало работы

Для запуска игры необходимо открыть файлы Xonix.exe. После чего появляется окно игры, представленное на рисунке 5.1.



Рисунок 5.1 – окно игры

Управление курсором производится с помощью стрелок. Для увеличения, уменьшения максимальной скорости соперников используются клавиши \*, / соответственно. Для увеличения, уменьшения числа соперников используются клавиши +, - соответственно. Для перезапуска игры используется клавиша “end”. Для победы необходимо захватить 70% поля.

Заключение

В ходе разработки приложения был создан продукт, предоставляющий возможность играть в Xonix на различных уровнях скорости и с различным числом врагов.

При тестировании приложения не было выявлено случаев некорректной работы программы и появления ошибок.

Написанный код использую функций Windows API как для вывода на экран так и для определения действий пользователя.

Итогами данной работы является опыт разработки приложений на языке программирования C++, изучение и применение основных функций Widwos API, а также опыт работы с много-поточностью и создания аналитических функций.

Список использованной литературы

[1] Рихтер Джеффри, Windows для профессионалов /Роберт Лафоре. – Санкт-Петербург: Классика Computer Science, 2001, 764 с

[2] Никита Культин Искусство программирования: в 4 т. /Никита Культин – Санкт-Петербург: БХВ-Петербург, 2009, 368 с.

[3] [Microsoft Developer Network](https://msdn.microsoft.com/ru-RU/)> Примеры [Электронный ресурс] – Режим доступа: https://code.msdn.microsoft.com. – Дата доступа: 24.11.2019.

[5] Блох Д.: Эффективное программирование, 2001– 464 с.

[6] ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 01.01.1992. – М.: Изд-во стандартов, 1991.

[7] ГОСТ 7. 53–2001. Издания. Международная стандартная нумерация книг [Текст]. – Взамен ГОСТ 7.53–86 ; введ. 2002–07–01. – Минск: Межгос. совет по стандартизации, метрологии и сертификации ; М.: Изд-во стандартов, cop. 2002. – 3 с. – 05.10.2018

# ПРИЛОЖЕНИЕ

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace Archmage

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new FMain());

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Sockets;

using System.Reflection;

using System.Text;

using System.Windows.Forms;

namespace Archmage

{

public partial class FMain : Form

{

private List<Card> qCardsList = new List<Card>();

private Socket listenSocket;

private Socket handler;

private int Port = 8888;

private IPAddress localAddr = IPAddress.Parse("127.0.0.1");

public FMain()

{

InitializeComponent();

LoadCards();

Lname.Parent = this;

Lname.BackColor = Color.Transparent;

}

private void StartGame(string whosFirst)

{

Form GameForm = new FGame(qCardsList, listenSocket, handler, whosFirst);

this.Hide();

GameForm.ShowDialog();

GameForm.Dispose();

this.Show();

if (listenSocket != null)

listenSocket.Close();

}

private void LoadCards()

{

string CardsPath = Path.Combine(Directory.GetCurrentDirectory(), "Cards");

DirectoryInfo CardsDirectory = new DirectoryInfo(CardsPath);

if (CardsDirectory.Exists)

{

var pluginFiles = Directory.GetFiles(CardsPath, "\*.jpg");

foreach (var file in pluginFiles)

{

Card newCard = new Card();

newCard.CardImage = Image.FromFile(file);

string TxtFilePath = file.Substring(0, file.Length - 4) + ".txt";

using (FileStream FileReadStream = new FileStream(TxtFilePath, FileMode.OpenOrCreate))

{

InitializeNewCard(FileReadStream, newCard);

}

qCardsList.Add(newCard);

}

}

else

{

CardsDirectory.Create();

}

}

private void InitializeNewCard(Stream getStream, Card newCard)

{

using (StreamReader ReadStream = new StreamReader(getStream))

{

string line;

string type = "";

string value = "";

while ((line = ReadStream.ReadLine()) != null)

{

type = GetTypeFromLine(line);

FieldInfo field = newCard.GetType().GetField(type);

value = GetValueFromLine(line);

if (field != null)

{

field.SetValue(newCard, Convert.ChangeType(value, field.FieldType));

}

}

}

}

private string GetTypeFromLine(string line)

{

int beg = 0;

int end = 0;

beg = line.IndexOf('"');

end = beg + 1;

while (line[end] != '"')

end++;

return line.Substring(beg + 1, end - beg - 1);

}

private string GetValueFromLine(string line)

{

int beg = 0;

beg = line.IndexOf(':');

beg += 2;

return line.Substring(beg);

}

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

{

IPEndPoint localEP = new IPEndPoint(localAddr, Port);

listenSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

try

{

listenSocket.Bind(localEP);

}

catch

{

MessageBox.Show("Произошли проблемы с установкой локальной конечной точкой. Попробуйте снова!");

return;

}

listenSocket.Listen(5);

handler = listenSocket.Accept();

StartGame("Me");

}

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

{

listenSocket = null;

IPEndPoint localEP = new IPEndPoint(localAddr, Port);

handler = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

try

{

handler.Connect(localEP);

}

catch

{

MessageBox.Show("На данный момент не к кому подключиться");

return;

}

StartGame("Not me");

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Archmage

{

public class Player

{

public int CurrentTower;

public int CurrentWall;

public int MineIncrease;

public int CurrentMinerals;

public int MonasteryIncrease;

public int CurrentMana;

public int BarrackIncrease;

public int CurrentSquads;

public bool DoesMyTurn;

public List<Card> Hand;

public Player()

{

InitializePlayer();

}

public void InitializePlayer()

{

CurrentTower = 20;

CurrentWall = 15;

MineIncrease = 3;

MonasteryIncrease = 3;

BarrackIncrease = 3;

CurrentMinerals = 10;

CurrentMana = 10;

CurrentSquads = 10;

Hand = new List<Card>(5);

}

}

}

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

namespace Archmage

{

[Serializable]

public class Card

{

public int Cost;

public string TypeOfCost;

public bool PlayAgain;

public int MyDamage;

public int EnemyDamage;

public int MyTower;

public int MyWall;

public int MyMineIncrease;

public int MyCurrentMinerals;

public int MyMonasteryIncrease;

public int MyCurrentMana;

public int MyBarrackIncrease;

public int MyCurrentSquads;

public int EnemyTower;

public int EnemyWall;

public int EnemyMineIncrease;

public int EnemyCurrentMinerals;

public int EnemyMonasteryIncrease;

public int EnemyCurrentMana;

public int EnemyBarrackIncrease;

public int EnemyCurrentSquads;

public string Condition;

public Image CardImage;

public Card()

{

MyDamage = 0;

EnemyDamage = 0;

MyTower = 0;

MyWall = 0;

MyMineIncrease = 0;

MyCurrentMinerals = 0;

MyMonasteryIncrease = 0;

MyCurrentMana = 0;

MyBarrackIncrease = 0;

MyCurrentSquads = 0;

EnemyTower = 0;

EnemyWall = 0;

EnemyMineIncrease = 0;

EnemyCurrentMinerals = 0;

EnemyMonasteryIncrease = 0;

EnemyCurrentMana = 0;

EnemyBarrackIncrease = 0;

EnemyCurrentSquads = 0;

Condition = "none";

CardImage = null;

}

}

[Serializable]

public class NetObjectToSend

{

public string Action;

public Card PlayedCard;

public NetObjectToSend()

{

Action = "Play";

PlayedCard = null;

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Net.Sockets;

using System.Reflection;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading;

using System.Windows.Forms;

using System.Timers;

using System.Net;

namespace Archmage

{

public partial class FGame : Form

{

public Socket MySocket;

public Socket ServerSocket;

static public MemoryStream GameInfoStream = new MemoryStream(new byte[MaxBuffer], 0, MaxBuffer, true, true);

public NetObjectToSend ObjectToSend;

public Player MyPlayer;

public Player EnemyPlayer;

public List<Card> CardsList = new List<Card>();

public List<Card> CurrentDeck;

public List<PictureBox> HandList = new List<PictureBox>(5);

public int CardsTaken;

public bool DoesTurnEnd;

public bool DoesItEnd;

public bool DidHeLeft;

public bool DoesEnemyPlayedCard;

public bool DoesItHost;

public const int DefaultMyTowerY = 453 - 65;

public const int DefaultMyWallY = 548 - 65;

public const int MaxTower = 90;

public const int MaxWall = 60;

public const int MaxCardsInHand = 5;

public const int MaxBuffer = 500000;

public int TowerWinCondition = 80;

public int ResourcesWinCondition = 600;

public Random MyRandom;

public FGame(List<Card> pCardsList, Socket qServerSocket, Socket getSocket, string whosFirst)

{

InitializeComponent();

this.WindowState = FormWindowState.Maximized;

CardsList = pCardsList;

MySocket = getSocket;

ServerSocket = qServerSocket;

MyPlayer = new Player();

EnemyPlayer = new Player();

if (whosFirst == "Me")

{

MyPlayer.DoesMyTurn = true;

EnemyPlayer.DoesMyTurn = false;

DoesItHost = true;

}

else

{

MyPlayer.DoesMyTurn = false;

EnemyPlayer.DoesMyTurn = true;

DoesItHost = false;

}

DoesItEnd = false;

DidHeLeft = false;

DoesEnemyPlayedCard = false;

RedrawPlayersInfo();

HandList.Add(PBcard1);

HandList.Add(PBcard2);

HandList.Add(PBcard3);

HandList.Add(PBcard4);

HandList.Add(PBcard5);

InitializeAllFormComponents();

//LoadCards();

CurrentDeck = new List<Card>(CardsList.Count);

if (!MyPlayer.DoesMyTurn)

{

Random buf = new Random();

MyRandom = new Random(buf.Next(0,100));

}

else

{

Random buf = new Random();

MyRandom = new Random(buf.Next(101,200));

}

DoRandomDeck(CardsList, CurrentDeck);

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

{

Card addCard = CurrentDeck[i];

MyPlayer.Hand.Add(addCard);

}

CardsTaken = 5;

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

{

HandList[i].Image = MyPlayer.Hand[i].CardImage;

//CardsPlayed++;

}

ObjectToSend = null;

BeginReceiveGameInfo(MySocket);

}

private void InitializeAllFormComponents()

{

string GameImagesPath = Path.Combine(Directory.GetCurrentDirectory(), "GamePictures");

PBbackground.Image = Image.FromFile(GameImagesPath + "\\bgField.jpg");

PBbackground.SizeMode = PictureBoxSizeMode.StretchImage;

PBtable.Image = Image.FromFile(GameImagesPath+ "\\bgTable.jpg");

PBtable.SizeMode = PictureBoxSizeMode.StretchImage;

PBmyMine.Image = Image.FromFile(GameImagesPath + "\\Mine.png");

PBmyMine.Parent = PBbackground;

PBmyMine.SizeMode = PictureBoxSizeMode.StretchImage;

PBmyMonastery.Image = Image.FromFile(GameImagesPath + "\\Monastery.png");

PBmyMonastery.Parent = PBbackground;

PBmyMonastery.SizeMode = PictureBoxSizeMode.StretchImage;

PBmyBarrack.Image = Image.FromFile(GameImagesPath + "\\Barrack.png");

PBmyBarrack.Parent = PBbackground;

PBmyBarrack.SizeMode = PictureBoxSizeMode.StretchImage;

PBmyWall.Image = Image.FromFile(GameImagesPath + "\\MyBldWall.png");

PBmyWall.SizeMode = PictureBoxSizeMode.AutoSize;

PBmyWall.Parent = PBbackground;

PBmyWall.BackColor = Color.Transparent;

PBmyTower.Image = Image.FromFile(GameImagesPath + "\\MyBldTower.png");

PBmyTower.SizeMode = PictureBoxSizeMode.AutoSize;

PBmyTower.Parent = PBbackground;

PBmyTower.BackColor = Color.Transparent;

PBenemyMine.Image = Image.FromFile(GameImagesPath + "\\Mine.png");

PBenemyMine.Image.RotateFlip(RotateFlipType.Rotate180FlipY);

PBenemyMine.Parent = PBbackground;

PBenemyMine.SizeMode = PictureBoxSizeMode.StretchImage;

PBenemyMonastery.Image = Image.FromFile(GameImagesPath + "\\Monastery.png");

PBenemyMonastery.Image.RotateFlip(RotateFlipType.Rotate180FlipY);

PBenemyMonastery.Parent = PBbackground;

PBenemyMonastery.SizeMode = PictureBoxSizeMode.StretchImage;

PBenemyBarrack.Image = Image.FromFile(GameImagesPath + "\\Barrack.png");

PBenemyBarrack.Image.RotateFlip(RotateFlipType.Rotate180FlipY);

PBenemyBarrack.Parent = PBbackground;

PBenemyBarrack.SizeMode = PictureBoxSizeMode.StretchImage;

PBenemyWall.Image = Image.FromFile(GameImagesPath + "\\EnemyBldWall.png");

PBenemyWall.SizeMode = PictureBoxSizeMode.AutoSize;

PBenemyWall.Parent = PBbackground;

PBenemyWall.BackColor = Color.Transparent;

PBenemyTower.Image = Image.FromFile(GameImagesPath + "\\EnemyBldTower.png");

PBenemyTower.SizeMode = PictureBoxSizeMode.AutoSize;

PBenemyTower.Parent = PBbackground;

PBenemyTower.BackColor = Color.Transparent;

PBplayedCardNew.SizeMode = PictureBoxSizeMode.StretchImage;

PBplayedCardNew.Visible = false;

PBplayedCardOld.SizeMode = PictureBoxSizeMode.StretchImage;

PBplayedCardOld.Visible = false;

PBplayedCardVeryOld.SizeMode = PictureBoxSizeMode.StretchImage;

PBplayedCardVeryOld.Visible = false;

LplayedCardNew.Visible = false;

LplayedCardOld.Visible = false;

LplayedCardVeryOld.Visible = false;

PBwinCondition.Image = Image.FromFile(GameImagesPath + "\\WinCondition.jpg");

PBwinCondition.SizeMode = PictureBoxSizeMode.StretchImage;

Bfake.Select();

PBcard1.SizeMode = PictureBoxSizeMode.StretchImage;

PBcard2.SizeMode = PictureBoxSizeMode.StretchImage;

PBcard3.SizeMode = PictureBoxSizeMode.StretchImage;

PBcard4.SizeMode = PictureBoxSizeMode.StretchImage;

PBcard5.SizeMode = PictureBoxSizeMode.StretchImage;

}

private void DiscardClick(object sender, EventArgs e)

{

if (MyPlayer.DoesMyTurn)

{

string SenderName = (sender as Button).Name;

int PBnumber = Convert.ToInt16(SenderName.Substring(SenderName.Length - 1));

Card PlayedCard = MyPlayer.Hand[PBnumber - 1];

AddNewCardToHand(PBnumber);

MyPlayer.DoesMyTurn = false;

EnemyPlayer.DoesMyTurn = true;

RedrawTurnHistory(PlayedCard, "Discard");

RedrawPlayersInfo();

SendGameInfo(MySocket, PlayedCard, "Discard");

Bfake.Select();

}

}

private void PBclick(object sender, EventArgs e)

{

if ((!DoesItEnd) && (MyPlayer.DoesMyTurn))

{

string SenderName = (sender as PictureBox).Name;

int PBnumber = Convert.ToInt16(SenderName.Substring(SenderName.Length - 1));

Card PlayedCard = MyPlayer.Hand[PBnumber - 1];

CardAction(PlayedCard, MyPlayer, EnemyPlayer, PBnumber);

}

}

private void CardAction(Card PlayedCard, Player whoPlayed, Player enemyPlayer , int qPBnumber)

{

bool CanIPlayThisCard = true;

MinusCardCost(PlayedCard, ref CanIPlayThisCard, whoPlayed);

if (!CanIPlayThisCard)

{

LwhichTurn.Visible = false;

LturnInfo.Text = "Вы не можете сыграть выбранную карту на данным момент";

LturnInfo.Visible = true;

return;

}

RecountMyPlayer(PlayedCard, whoPlayed, enemyPlayer);

RecountEnemyPlayer(PlayedCard, enemyPlayer);

if (MyPlayer.DoesMyTurn)

{

AddNewCardToHand(qPBnumber);

SendGameInfo(MySocket, PlayedCard, "Play");

}

CheckWhosTurn(PlayedCard, whoPlayed, enemyPlayer);

if (DoesTurnEnd)

{

AddResources(enemyPlayer);

}

RedrawPlayersInfo();

RedrawTurnHistory(PlayedCard, "Action");

DoesItEnd = DoesTheEnd(MyPlayer, EnemyPlayer);

}

private void MinusCardCost(Card pCard, ref bool pCanIPlayThisCard, Player getPlayer)

{

switch (pCard.TypeOfCost)

{

case "red":

if ((getPlayer.CurrentMinerals - pCard.Cost) >= 0)

{

getPlayer.CurrentMinerals -= pCard.Cost;

}

else

{

pCanIPlayThisCard = false;

}

break;

case "blue":

if ((getPlayer.CurrentMana - pCard.Cost) >= 0)

{

getPlayer.CurrentMana -= pCard.Cost;

}

else

{

pCanIPlayThisCard = false;

}

break;

case "green":

if ((getPlayer.CurrentSquads - pCard.Cost) >= 0)

{

getPlayer.CurrentSquads -= pCard.Cost;

}

else

{

pCanIPlayThisCard = false;

}

break;

}

}

private void RecountMyPlayer(Card pCard, Player getPlayer, Player enemyPlayer)

{

if (pCard.Condition != "none")

{

ResolveCondition(pCard, getPlayer, EnemyPlayer);

}

DoDamage(getPlayer, pCard.MyDamage);

IntFieldCount(ref getPlayer.CurrentTower, pCard.MyTower);

IntFieldCount(ref getPlayer.CurrentWall, pCard.MyWall);

IntFieldCount(ref getPlayer.MineIncrease, pCard.MyMineIncrease);

IntFieldCount(ref getPlayer.CurrentMinerals, pCard.MyCurrentMinerals);

IntFieldCount(ref getPlayer.MonasteryIncrease, pCard.MyMonasteryIncrease);

IntFieldCount(ref getPlayer.CurrentMana, pCard.MyCurrentMana);

IntFieldCount(ref getPlayer.BarrackIncrease, pCard.MyBarrackIncrease);

IntFieldCount(ref getPlayer.CurrentSquads, pCard.MyCurrentSquads);

if (getPlayer.CurrentTower > MaxTower)

getPlayer.CurrentTower = MaxTower;

if (getPlayer.CurrentWall > MaxWall)

getPlayer.CurrentWall = MaxWall;

}

private void RecountEnemyPlayer(Card pCard, Player getPlayer)

{

DoDamage(getPlayer, pCard.EnemyDamage);

IntFieldCount(ref getPlayer.CurrentTower, pCard.EnemyTower);

IntFieldCount(ref getPlayer.CurrentWall, pCard.EnemyWall);

IntFieldCount(ref getPlayer.MineIncrease, pCard.EnemyMineIncrease);

IntFieldCount(ref getPlayer.CurrentMinerals, pCard.EnemyCurrentMinerals);

IntFieldCount(ref getPlayer.MonasteryIncrease, pCard.EnemyMonasteryIncrease);

IntFieldCount(ref getPlayer.CurrentMana, pCard.EnemyCurrentMana);

IntFieldCount(ref getPlayer.BarrackIncrease, pCard.EnemyBarrackIncrease);

IntFieldCount(ref getPlayer.CurrentSquads, pCard.EnemyCurrentSquads);

if (getPlayer.CurrentTower > MaxTower)

getPlayer.CurrentTower = MaxTower;

if (getPlayer.CurrentWall > MaxWall)

getPlayer.CurrentWall = MaxWall;

}

private void IntFieldCount(ref int field, int Minus)

{

field += Minus;

if (field < 0)

field = 0;

}

private void DoDamage(Player getPlayer, int Damage)

{

if ((getPlayer.CurrentWall - Damage) >= 0)

{

getPlayer.CurrentWall -= Damage;

}

else

{

getPlayer.CurrentTower = getPlayer.CurrentTower - (Damage - getPlayer.CurrentWall);

getPlayer.CurrentWall = 0;

}

}

private void ResolveCondition(Card getCard, Player WhoDid, Player WhoGet)

{

string line = getCard.Condition;

int StartPosition = 0;

int EndPosition = 0;

string firstFieldString = GetWord(line, ref StartPosition, ref EndPosition);

FieldInfo firstField = WhoDid.GetType().GetField(firstFieldString);

string Sight = GetWord(line, ref StartPosition, ref EndPosition);

string secondFieldString = GetWord(line, ref StartPosition, ref EndPosition);

FieldInfo secondField = WhoGet.GetType().GetField(secondFieldString);

bool WhatResult = false;

int firstValue;

if (firstField != null)

{

firstValue = Convert.ToInt32(firstField.GetValue(WhoDid));

}

else

{

firstValue = Convert.ToInt32(firstFieldString);

}

int secondValue;

if (secondField != null)

{

secondValue = Convert.ToInt32(secondField.GetValue(WhoGet));

}

else

{

secondValue = Convert.ToInt32(secondFieldString);

}

switch (Sight)

{

case "=":

if (firstValue == secondValue)

{

WhatResult = true;

}

else

{

WhatResult = false;

}

break;

case ">":

if (firstValue > secondValue)

{

WhatResult = true;

}

else

{

WhatResult = false;

}

break;

case "<":

if (firstValue < secondValue)

{

WhatResult = true;

}

else

{

WhatResult = false;

}

break;

}

FieldInfo currentField;

int value;

int buf = 0;

if (WhatResult)

{

\_ = GetWord(line, ref StartPosition, ref EndPosition);

while (Sight != "?")

{

currentField = getCard.GetType().GetField(GetWord(line, ref StartPosition, ref EndPosition));

string getValue = GetWord(line, ref StartPosition, ref EndPosition);

switch (getValue)

{

case "!":

if (firstValue != 0)

value = firstValue;

else

value = buf;

buf = secondValue;

secondValue = 0;

secondField.SetValue(WhoGet, 0);

break;

case "@":

if (secondValue != 0)

value = secondValue;

else

value = buf;

buf = firstValue;

firstValue = 0;

firstField.SetValue(WhoDid, 0);

break;

default:

value = Convert.ToInt32(getValue);

break;

}

currentField.SetValue(getCard, value);

Sight = GetWord(line, ref StartPosition, ref EndPosition);

}

}

else

{

while (Sight != "?")

{

Sight = GetWord(line, ref StartPosition, ref EndPosition);

}

while (StartPosition <= line.Length - 1)

{

currentField = getCard.GetType().GetField(GetWord(line, ref StartPosition, ref EndPosition));

string getValue = GetWord(line, ref StartPosition, ref EndPosition);

switch (getValue)

{

case "!":

if (firstValue != 0)

value = firstValue;

else

value = buf;

buf = secondValue;

secondValue = 0;

secondField.SetValue(WhoGet, 0);

break;

case "@":

if (secondValue != 0)

value = secondValue;

else

value = buf;

buf = firstValue;

firstValue = 0;

firstField.SetValue(WhoDid, 0);

break;

default:

value = Convert.ToInt32(getValue);

break;

}

currentField.SetValue(getCard, value);

Sight = GetWord(line, ref StartPosition, ref EndPosition);

}

}

}

private string GetWord(string line, ref int beg, ref int end)

{

string resultString = "";

end = beg;

while ((end < line.Length) && (line[end] != ' '))

{

resultString = resultString + line[end];

end++;

}

beg = end + 1;

return resultString;

}

private void CheckWhosTurn(Card pCard, Player getPlayer, Player enemyPlayer)

{

if (pCard.PlayAgain)

{

DoesTurnEnd = false;

}

else

{

DoesTurnEnd = true;

getPlayer.DoesMyTurn = !getPlayer.DoesMyTurn;

enemyPlayer.DoesMyTurn = !enemyPlayer.DoesMyTurn;

}

}

private void AddNewCardToHand(int qPBnumber)

{

MyPlayer.Hand[qPBnumber - 1] = CurrentDeck[CardsTaken];

CardsTaken++;

if (CardsTaken == CurrentDeck.Count)

{

CurrentDeck.Clear();

DoRandomDeck(CardsList, CurrentDeck);

CardsTaken = 0;

}

}

private void AddResources (Player WhoGet)

{

WhoGet.CurrentMinerals += WhoGet.MineIncrease;

WhoGet.CurrentMana += WhoGet.MonasteryIncrease;

WhoGet.CurrentSquads += WhoGet.BarrackIncrease;

}

private bool DoesTheEnd(Player pMyPlayer, Player pEnemyPlayer)

{

if ((pMyPlayer.CurrentTower <= 0) && (pEnemyPlayer.CurrentTower <= 0))

{

MessageBox.Show("It's a draw!");

return true;

}

if ((pMyPlayer.CurrentTower <= 0) || (pEnemyPlayer.CurrentTower >= TowerWinCondition)

|| (((pEnemyPlayer.CurrentMinerals >= (ResourcesWinCondition / 3)))

&& (pEnemyPlayer.CurrentMana >= (ResourcesWinCondition / 3))

&& (pEnemyPlayer.CurrentSquads >= (ResourcesWinCondition / 3))))

{

MessageBox.Show("Ooooh no ... You've lost :(");

return true;

}

if ((pEnemyPlayer.CurrentTower <= 0) || (pMyPlayer.CurrentTower >= TowerWinCondition)

|| (((pMyPlayer.CurrentMinerals >= (ResourcesWinCondition / 3)))

&& (pMyPlayer.CurrentMana >= (ResourcesWinCondition / 3))

&& (pMyPlayer.CurrentSquads >= (ResourcesWinCondition / 3))))

{

MessageBox.Show("Congratulations!!! You've won :)");

return true;

}

return false;

}

private void DoRandomDeck(List<Card> CardLists, List<Card> NewDeck)

{

List<Card> ListForRandom = new List<Card>();

ListForRandom.AddRange(CardsList);

int amount = CardLists.Count;

int CardsRandomed = 1;

while (CardsRandomed <= amount)

{

Card addingCard = ListForRandom[MyRandom.Next(ListForRandom.Count)];

NewDeck.Add(addingCard);

ListForRandom.Remove(addingCard);

CardsRandomed++;

}

}

private void RedrawTurnHistory(Card pPlayedCard, string pAction)

{

if (PBplayedCardOld.Image != null)

{

PBplayedCardVeryOld.Image = PBplayedCardOld.Image;

PBplayedCardVeryOld.Visible = true;

}

if (PBplayedCardNew.Image != null)

{

PBplayedCardOld.Image = PBplayedCardNew.Image;

PBplayedCardOld.Visible = true;

}

PBplayedCardNew.Image = pPlayedCard.CardImage;

PBplayedCardNew.Visible = true;

LplayedCardVeryOld.Visible = LplayedCardOld.Visible;

LplayedCardOld.Visible = LplayedCardNew.Visible;

if (pAction == "Discard")

{

LplayedCardNew.Visible = true;

}

else

{

LplayedCardNew.Visible = false;

}

}

private void RedrawPlayersInfo()

{

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

{

HandList[i].Image = MyPlayer.Hand[i].CardImage;

}

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

PBmyTower.Visible = false;

PBmyTower.Location = new Point(PBmyTower.Location.X, DefaultMyTowerY - (MyPlayer.CurrentTower \* 4));

PBmyTower.Visible = true;

PBmyWall.Visible = false;

PBmyWall.Location = new Point(PBmyWall.Location.X, DefaultMyWallY - (MyPlayer.CurrentWall \* 8));

PBmyWall.Visible = true;

LmyTower.Text = Convert.ToString(MyPlayer.CurrentTower);

LmyWall.Text = Convert.ToString(MyPlayer.CurrentWall);

LmyMine.Text = "+" + Convert.ToString(MyPlayer.MineIncrease);

LmyMinerals.Text = "Руда: " + Convert.ToString(MyPlayer.CurrentMinerals);

LmyMonastery.Text = "+" + Convert.ToString(MyPlayer.MonasteryIncrease);

LmyMana.Text = "Мана: " + Convert.ToString(MyPlayer.CurrentMana);

LmyBarrack.Text = "+" + Convert.ToString(MyPlayer.BarrackIncrease);

LmySquads.Text = "Отряды: " + Convert.ToString(MyPlayer.CurrentSquads);

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

PBenemyTower.Visible = false;

PBenemyTower.Location = new Point(PBenemyTower.Location.X, DefaultMyTowerY - (EnemyPlayer.CurrentTower \* 4));

PBenemyTower.Visible = true;

PBenemyWall.Visible = false;

PBenemyWall.Location = new Point(PBenemyWall.Location.X, DefaultMyWallY - (EnemyPlayer.CurrentWall \* 8));

PBenemyWall.Visible = true;

LenemyTower.Text = Convert.ToString(EnemyPlayer.CurrentTower);

LenemyWall.Text = Convert.ToString(EnemyPlayer.CurrentWall);

LenemyMine.Text = "+" + Convert.ToString(EnemyPlayer.MineIncrease);

LenemyMinerals.Text = "Руда: " + Convert.ToString(EnemyPlayer.CurrentMinerals);

LenemyMonastery.Text = "+" + Convert.ToString(EnemyPlayer.MonasteryIncrease);

LenemyMana.Text = "Мана: " + Convert.ToString(EnemyPlayer.CurrentMana);

LenemyBarrack.Text = "+" + Convert.ToString(EnemyPlayer.BarrackIncrease);

LenemySquads.Text = "Отряды: " + Convert.ToString(EnemyPlayer.CurrentSquads);

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

LturnInfo.Visible = false;

LwhichTurn.Visible = true;

if (MyPlayer.DoesMyTurn)

LwhichTurn.Text = "Ваш ход";

else

LwhichTurn.Text = "Ход оппонента";

}

public class StateObject

{

public Socket workSocket = null;

public const int BUFFER\_SIZE = MaxBuffer;

public byte[] buffer = new byte[BUFFER\_SIZE];

public NetObjectToSend sendObject = null;

}

//Asunc function which wait while opponent dont send data

public void BeginReceiveGameInfo(Socket socket)

{

StateObject state = new StateObject();

state.workSocket = socket;

GameInfoStream.Position = 0;

socket.BeginReceive(GameInfoStream.GetBuffer(), 0, GameInfoStream.GetBuffer().Length, 0, new AsyncCallback(Read\_Callback), state);

}

//Callback function for processing receiving data

public void Read\_Callback(IAsyncResult ar)

{

if (MySocket == null) return;

StateObject state = (StateObject)ar.AsyncState;

Socket s = state.workSocket;

int receivedBytes = s.EndReceive(ar);

GameInfoStream.Position = 0;

BinaryFormatter bin = new BinaryFormatter();

NetObjectToSend getObject = (NetObjectToSend)bin.Deserialize(GameInfoStream);

ObjectToSend = getObject;

DoesEnemyPlayedCard = true;

GameInfoStream.Position = 0;

}

public void ReceiveGameInfo(Socket socket)

{

GameInfoStream.Position = 0;

int receivedBytes = socket.Receive(GameInfoStream.GetBuffer(), GameInfoStream.GetBuffer().Length, 0);

GameInfoStream.Position = 0;

BinaryFormatter bin = new BinaryFormatter();

NetObjectToSend getObject = (NetObjectToSend)bin.Deserialize(GameInfoStream);

ObjectToSend = getObject;

GameInfoStream.Position = 0;

}

public void SendGameInfo(Socket socket, Card PlayedCard, string Action)

{

GameInfoStream.Position = 0;

BinaryFormatter bin = new BinaryFormatter();

NetObjectToSend sendObject = new NetObjectToSend();

sendObject.Action = Action;

sendObject.PlayedCard = PlayedCard;

bin.Serialize(GameInfoStream, sendObject);

socket.Send(GameInfoStream.GetBuffer());

GameInfoStream.Position = 0;

}

private void PBbackground\_MouseMove(object sender, MouseEventArgs e)

{

if ((DoesEnemyPlayedCard) && (ObjectToSend != null))

{

BeginReceiveGameInfo(MySocket);

DoesEnemyPlayedCard = false;

if (ObjectToSend.Action == "Play")

{

CardAction(ObjectToSend.PlayedCard, EnemyPlayer, MyPlayer, 0);

}

else

{

if (ObjectToSend.Action == "Discard")

{

EnemyPlayer.DoesMyTurn = false;

MyPlayer.DoesMyTurn = true;

RedrawTurnHistory(ObjectToSend.PlayedCard, "Discard");

RedrawPlayersInfo();

}

else

{

MessageBox.Show("Your opponent has left");

DidHeLeft = true;

this.Close();

}

}

}

}

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

{

if (!DidHeLeft)

{

SendGameInfo(MySocket, null, "Exit");

Thread.Sleep(5000);

}

if (MySocket != null)

{

MySocket.Shutdown(SocketShutdown.Both);

MySocket.Close();

MySocket = null;

}

this.Close();

}

}

}