Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные Системы и Сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

Игровое приложение «Archmage»

БГУИР КП 1-40 01 01 1 ПЗ

Студент: гр. 751003 Гринчик В.В.

Проверил: Красковский П.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 *Программное средство развития мышления и социальных навыков, в игровой форме. С возможностью одновременного участия нескольких человек «WebPoker» .*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

2. Срок сдачи законченной работы *08.06.2019г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Язык программирования C#. Среда разработки Microsoft Visual Studio 2017. Реализация обучающей игры “WebPoker” с помощью клиент-серверного приложения на базе протокола TCP. Существует возможность большого числа пользователей, также предусмотрена возможность расширения программного средства.*

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

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

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

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

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

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_*

*5 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 *Красковский П.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *15.02.2019г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования

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

*Раздел 1. Введение к 15.03.2019г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 30.03.2019г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 15.04.2019г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 30.04.2019г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5.Заключение. Приложения к 9.05.2019г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 30.05.2019г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 10.05.2019г. по 08.06.2019г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Красковский П.Н.*

*(подпись)*

Задание принял к исполнению *Гринчик В.В. 15.02.2019г.*

*(дата и подпись студента)*

Содержание

[Введение 5](#_Toc10554891)

[1 Анализ литературных источников 6](#_Toc10554892)

[1.1 Анализ существующих приложений 6](#_Toc10554893)

[2 моделирование предметной области 9](#_Toc10554894)

[2.1 Цель ….…………………………………………………………………………………..9](#_Toc10554895)

[2.2 Задачи 9](#_Toc10554896)

[2.3 Обоснование выбора языка программирования 9](#_Toc10554897)

[2.4 Описание процесса соединения игроков 10](#_Toc10554898)

[2.5 Описание процесса прорисовки 10](#_Toc10554899)

[3 Разработка программного средства 11](#_Toc10554900)

[3.1 Визуальное представление приложения 11](#_Toc10554901)

[3. 2 Схема алгоритмов решения задачи по ГОСТ 19.701-90 15](#_Toc10554902)

[4 Тестирование и проверка работоспособности программного средства 20](#_Toc10554903)

[5 Руководство по установке и использованию программного средства 29](#_Toc10554904)

[5.1 Начало работы 29](#_Toc10554905)

[Заключение 30](#_Toc10554906)

[Список использованной литературы 31](#_Toc10554907)

[6 ПРИЛОЖЕНИЕ 32](#_Toc10554908)

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. Сетевая компьютерная игра с ограниченным числом у частников это некое виртуальное помещение, предназначенное в первую очередь для развития социальных навыков.

В наше время сложно найти человека, который бы ни разу не играл в компьютерные игры, и с каждым днём становится всё больше людей, которые постоянно играют в игры, и даже(!!!) зарабатывают на этом деньги. Будь то одиночная игра на игровой приставке или многопользовательская игра с огромным количеством игроков на одном сервере, заработать на ней деньги которые в ином случае, скорее всего, оказались бы на счету у компании разработчика. И потому заработок на играх является довольно тяжелой и непредсказуемой работой. По статистике в онлайн играх люди Гораздо чаще теряют чем зарабатывают. Однако большинство компаний разработчиков придерживаются тактики игр однодневок, в которых еще вчера подзаборный побирушка, уже завтра может стать бароном или даже герцогом. Но заработать на таких игра практически невозможно, эти созданы с целью обогатить разработчика , а не разделить богатство. Однако некоторые игры, относящиеся к так называемой не стареющей классике и в самумом деле позволяют заработать денег и даже найти постоянную работу. В популярные игры может одновременно играть более миллиона человек. По некоторым из них проводятся турниры с огромными призовыми фондами и сотнями тысяч зрителей на каждом матче. Благодаря таким играм, как Counter-Strike, Dota2, Overwatch, и многим другим зародился киберспорт. В то же время существует множество киберспортсменов и киберспортивных команд, которые зарабатывают огромные деньги благодаря таким турнирам.

Однако по настоящему Огромные деньги находятся в секторе азартных игр, ежедневно люди зарабатывают и теряют целые состояния в мгновение ока. Однако, по настоящему большой популярностью пользуются Обучающие игры, ведь нет ничего приятнее, чем обучаться чему-либо не прикладывая почти никаких усилий. А уж если в будущем это позволит еще и заработать… И именно на основание утверждений описанных выше я принял решение разработать игру которая-бы удовлетворяла мнению большинства вменяемых игроков об идеальной игре.

# Анализ литературных источников

## Анализ существующих приложений

Сегодня вряд ли найдётся человек, который никогда не брал бы в руки игральных карт, – он или хотя бы в детстве играл в «подкидного дурака», в «ведьму», или раскладывал уже в зрелом возрасте пасьянс, или, наконец, пытался показывать карточные фокусы.

Все придуманные человечеством карточные игры условно можно разделить на три категории. К первой категории относятся азартные игры, их насчитывается больше десятка, и среди них наиболее известна игра «двадцать одно», или «очко». Карточные игры, которые почему-то не считаются азартными, их более тридцати, относятся ко второй категории, хотя в эту группу входят преферанс, бридж, покер и они тоже, как и игры, считающиеся азартными, могут привести к потере целого состояния (но, однако, и к крупным выигрышам). В эту же категорию, по-видимому, входит и любимая игра немцев под названием «скат». В последние десятилетия преобладает мнение, что покер, самая распространённая теперь в мире карточная игра, а также бридж являются играми, близкими к спортивному типу, наподобие шахмат и шашек, и по бриджу и покеру даже проводятся международные турниры.

Суммы выигрыша в которых могут срамница с месячным бюджетом целой страны Впрочем, соревнования проводятся и по «подкидному дураку», хотя вообще-то, «подкидной дурак» относится, как считают специалисты, к третьей категории карточных игр, называемых детскими, их насчитывается тоже примерно десяток, и в это десяток входят и известные всем «ведьма» и «пьяница». Также на текущий момент довольно распространены так называемые компьютерные карточные игры, хотя в большинстве своем они являются лишь модификациями старейшей ролевой игры “Dungeons and Dragons”.

Однако я все же остановил своё внимание на такой игре как покер, а если точнее, то на самой популярной его разновидности Техасском Холдеме.

Итак Техасский Холдем, Если говорить о точной дате и месте рождения этой игры, то вряд ли где-либо можно найти достоверные цифры и факты. Наиболее приемлемой является версия, что он пришел к нам из Северной Америки (в частности, с Аляски) в том виде, в котором мы его знаем и сейчас.

Впервые правила Техасского Холдема были утверждены в 1900 г. в Техасе (Робстаун), чему собственно и обязано его название. Но по-настоящему его популярность стала набирать обороты с проведением первого турнира по этой покерной дисциплине в 1969 г.

Своей популярность холдем во многом обязан простоте своих правил.

А именно, в холдем и грают группами от пяти до десяти человек, каждому игроку раздают по две карты лицом вниз и рубашкой вверх. Это карманные карты.

Лучшими начальными картами, как говорят правила игры в Техасский Холдем являются два туза. Начинается первый круг торгов на префлопе. Как только все ставки были уравнены (то есть никто больше не поднимает), первый круг торгов завершается. После того, как первый раунд торговли завершен, по правилам игры, дилер выкладывает три общие карты в центр стола. Все оставшиеся игроки использует эти три карты вместе с их двумя начальными картами, чтобы собрать лучшую комбинацию. Эти три карты называются "флоп". На **рисунке 1.1** вы видите пример флопа:



Рисунок 1‑1– Флоп

после флопа начинается новый раунд торговли.. Обязательных ставок больше не существует, и у игрока есть три варианта – сделать чек, поставить фишки в банк Raise, ну или отказаться играть сбросив карты fold. **"Чек"** означает, что игрок остается в розыгрыше без повышения, если количество вложенных им фишек соответствует ставкам предыдущих игроков.После того, как второй раунд торговли завершен, на стол кладется еще одна карта, ее называют "терн", ну и после очередного круга торгов следует "ривер". По окончании последнего круга торгов игроки вскрываются, победитель определяется комбинацией его карт рисунок 1.2.

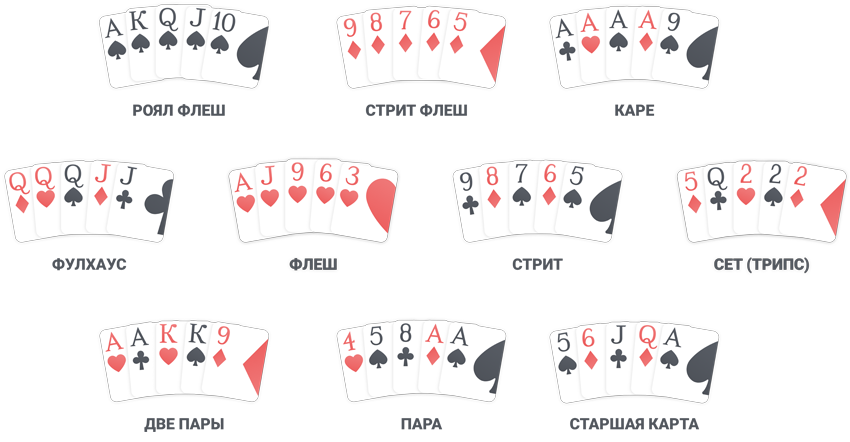


Рисунок 1‑2

Итак, промежуточные результаты благодаря популярности и простоте покер спокойно можно отнести к массовым играм таким ка MMO RPG,

Кроме того покер являясь одной из старейших карточных игр вполне заслуженно относиться к не стареющей классике. Таже благодоря небольшому числу игроков сидящих за одним столом, игроками ведется вполне оживленная беседа. Но это еще не все покер, несомненно, позволяет зарабатывать, итого 4 из 5. Итак чемуже может научить покер?

Невероятно многому, поскольку простота правил компенсируется искусством игроков. На основании данного утверждения можно также выделить еще одно достоинство покера ведь он как и любой искуство позволят развиваться не только визически( том числе и интелект) но и духовно. Итого 6 из 5. Покер является несомненно лучшим вариантом для моего программного средства.

# моделирование предметной области

## Цель

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

В процессе реализации программного средства планируется разработать игру с корректной передачей данных между многимb игроками, а также с возможностью подключения абсолютно любых интерфейсов, подходящих для запуска на любом устройстве поддерживающем формат данных JSON.

## 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, так и обучения на дому и не только, в связи с чем пользователю доступны следующие возможности:

* возможность провести партию игры с людьми независимо от используемых ими устройств и расстояния между товарищами.
* развить своё аналитическое мышление, а также тренироваться с абсолютно разными людьми.

## Обоснование выбора языка программирования

Для разработки программного средства использован язык программирования C#. Среда для разработки – Visual Studio 2017. Использование данной среды разработки дает возможность создания консольных и оконных приложений Windows. Выбор языка связан со следующими его преимуществами:

* язык программирования C# претендует на подлинную объектную  
  ориентированность (всякая языковая сущность претендует на то, чтобы  
  быть объектом);
* компонентно-ориентированный подход к программированию,  
  способствующий меньшей машинно-архитектурной зависимости  
  результирующего программного кода, гибкости, переносимости и легкости повторного использования (фрагментов) программ;
* ориентация на безопасность кода (в сравнении с С и С++);
* унифицированная система типизации;
* расширенная поддержка событийно-ориентированного  
  программирования.
* Поддержка мировых стандартов сериализации.

## Описание процесса соединения игроков

При разработке было принято решение сделать выделить независимый сервер, для обеспечения минимизации требований к пользовательским устройствам. Игрок нажимает кнопку «Старт» и совершает попытку обращения к заранее определенному порту сервера, который при наличии свободных мест в игровой комнате занимается прослушкой запросов. Другие игроки делают тоже самое. Севрер занимается поочередной обработкой первых n сигналов, и создает для каждого игрока свой сокет. При потери соединения с одним из игроков сервер по завешении партии вновь станет прослушивать порт подключения.

## Описание процесса взаимодействия сервера с клиентом

Перед отправкой все данные сиреализуются в формат JSON. Что позволяет использовать их приложениям написанных на различных видах языков программирования которые могут отличаться ка принципами хранения информации так и принципами их отправления.

При разработке было принято решение не отходить от принципов простоты и не использовать усложненные версии правил, а также не заниматся создание различных режимов игры. Для демонстрации возможности взаимодействия с различными видами интерфейсов был реализован графичиский(с использованием Windows form), консольный варианты интерфейсов.

* Одним из основных преимуществ «public»-наследования является то, что указатель на классы-наследники может быть неявно преобразован в указатель на базовый класс

# 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

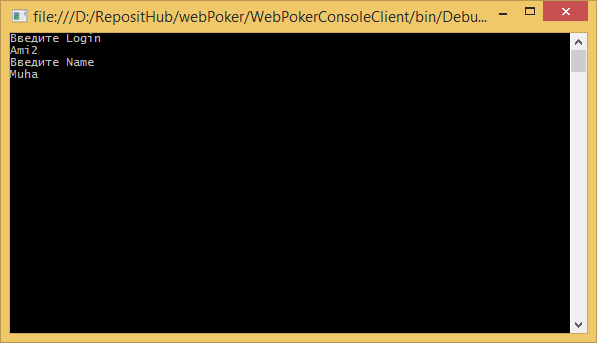


Рисунок 3.4 – Консольный интерфейс

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)

# Тестирование и проверка работоспособности программного средства

В ходе тестирования были рассмотрены в действии все функциональные возможности программного средства. Было изучено его поведение в зависимости от различных действий пользователя.

Таблица 4.1 ‑ Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация: | Тестирование работы сервера |
| Исходный набор данных: |  |
| Ожидаемый результат: | Ожидание подключения игроков |
| Фактический результат: |  |

Таблица 4.2 ‑ Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация: | Тестирование работы главного меню при нажатии на кнопку «Start» |
| Исходный набор данных: | Сервер не запущен. |
| Ожидаемый результат: | Выскочит сообщение о том, что не к кому подключиться |
| Фактический результат: |  |

Таблица 4.3 ‑ Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация: | Тестирование работы главного меню при нажатии на кнопку «Start» |
| Исходный набор данных: | Сервер запущен. |
| Ожидаемый результат: | Выполнится подключение к серверу и начнётся игра |
| Фактический результат: |  |

Таблица 4.4 ‑ Тест 4

|  |  |
| --- | --- |
| Тестовая ситуация: | Тестирование работы игры «WepPoker» при нажатии на кнопку Check/call |
| Исходный набор данных: |  |
| Ожидаемый результат: | переход хода. |
| Фактический результат: |  |

Таблица 4.5 ‑ Тест 5

|  |  |
| --- | --- |
| Тестовая ситуация: | Тестирование работы игры при нажатии на кнопку «Сбросить» |
| Исходный набор данных: |  |
| Ожидаемый результат: | Переход в режим наблюдателя(обнуление ставки) |
| Фактический результат: |  |

Таблица 4.6 ‑ Тест 6

|  |  |
| --- | --- |
| Тестовая ситуация: | Тестирование работы игры при нажатии на кнопку Raise |
| Исходный набор данных: |  |
| Ожидаемый результат: | Повышение ставки |
| Фактический результат: |  |

Таблица 4.7 ‑ Тест 7

|  |  |
| --- | --- |
| Тестовая ситуация: | Тестирование работы игры на продолжение игры |
| Исходный набор данных: |  |
| Ожидаемый результат: | Обновление игры |
| Фактический результат: |  |

# Руководство по установке и использованию программного средства

5.1 Начало работы

Для запуска игры необходимо открыть файлы WindowsFormsWebPOkerClient.exe и WebPokerServer. После чего появляется стартовое окно, представленно на рисунке 5.1.



Рисунок 5.1 – Стартовое меню

Далее необходимо ввести login name an port и нажать старт.

В случае, когда имеются два игрока, где один из них нажал «Создать игру», а другой «Подключиться к игроку», и они оба находятся в локальной сети, то будет создана партия, представленная на рисунке 5.2 (но необходимо миниму 2 клиента!):



Рисунок 5.2 – Стартовое меню

Заключение

В ходе разработки приложения был создан продукт, предоставляющий возможность провести множество тренировок с друзьями, или у строить устроить сражение с врагом.

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

Написанный код легко модифицируется, а также адаптирован к добавлению новых функций, данных.

Итогами данной работы является опыт разработки приложений на языке программирования C#, изучение и применение основных принципов объектно-ориентированного программирования, использование протокола TCP, многопоточности и создания аналитических функций.

Список использованной литературы

[1] Роберт Лафоре, Object-Oriented Programming in C#/ Роберт Лафоре. – Санкт-Петербург: Классика Computer Science, 2015, 986 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: 14.10.2018.

[4] Подключение библиотеки SFML к Visual Studio [Электронный ресурс] – Режим доступа: http://grafika.me/node/518. – Дата доступа: 14.11.2018.

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

[6] Пошаговое руководство. Добавление анимации в проект MFC [Электронный ресурс] – Режим доступа https://msdn.microsoft.com/ru-ru/library/gg466500.aspx. – Дата доступа: 29.09.2018.

[7] 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.

[8] 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();

}

}

}