Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы Алгоритмизации и Программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Программное средство

«Текстовый редактор с подсветкой и

проверкой кода Delphi»

Студент: гр. 451001 Поняков Д.С.

Руководитель: Болтак С.В.

Минск, 2015

Оглавление

[ВВЕДЕНИЕ 4](#_Toc407118306)

[1. АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ 5](#_Toc407118307)

[1.1 Анализ аналогичных программ 5](#_Toc407118308)

[1.2 Обоснование выбора языка программирования 6](#_Toc407118309)

[1.3 Обоснование выбора технологии 7](#_Toc407118310)

[1.4 Проектирование архитектуры программного средства 7](#_Toc407118311)

[1.5 Нефункциональные требования 9](#_Toc407118313)

[2 МОДЕЛЬ ДАННЫХ И ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ 10](#_Toc407118323)

[2.1 Функциональные требования к программному средству 10](#_Toc407118324)

[2.2 Диаграмма вариантов использования 11](#_Toc407118325)

[3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 12](#_Toc407118326)

[3.1 Разработка графического интерфейса пользователя 12](#_Toc407118327)

[3.2 Enum Opponents 12](#_Toc407118328)

[3.3 Enum CardOwner 12](#_Toc407118329)

[3.4 Enum CardStatus 12](#_Toc407118330)

[3.5 Class CardClass 13](#_Toc407118331)

[3.6 Реализация основных функций 13](#_Toc407118332)

[4 ТЕСТИРОВАНИЕ 19](#_Toc407118333)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 21](#_Toc407118334)

[ЗАКЛЮЧЕНИЕ 25](#_Toc407118335)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc407118336)

[ПРИЛОЖЕНИЕ А 27](#_Toc407118337)

# ВВЕДЕНИЕ

На сегодняшний день существует множество карточных игр на различные тематики. Одной из них является футбол.

Карточная игра «Футбол» будет оценена любителями футбола, поскольку вместо обычных игральных карт используются специальные карты с известными игроками. На каждой карте изображен футболист, его личные очки, а также национальность.

Правила достаточно просты. Каждый из игроков кладет на стол карту, побеждает обладатель карты с наибольшим числом очков. Он забирает себе карты со стола. Целью игры является собрать сильнейшие карты за определенное число раундов. Побеждает обладатель наибольшей суммы очков 11 сильнейших карт. Несмотря на простоту правил, игра не перестает быть интересной.

# 1. АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ

## 1.1 Анализ аналогичных программ

Одним из примеров программы является flash-игра Карточный футбол с сайта flashgames.ru. Правила данного приложения абсолютно совпадают с правилами разработанной игры. Это неплохое приложение для компьютеров и планшетов. Она ориентировано на игру пользователя с компьютером.







Рисунок 1.1 Приложение «Карточный футбол»

Достоинства:

* Приятный и понятный интерфейс
* Наличие обучающей игры
* Реализация игры с компьютером;
* Игра в формате турнира

Недостатки:

* Нет возможности сыграть двум пользователям;
* Для игры нужен обязательный доступ в интернет.

## 1.2 Обоснование выбора языка программирования

Для реализации курсового проекта было отдано предпочтение платформе .Net и языку С#. На данный момент эта платформа достаточно распространена и как результат имеет ряд преимуществ:

* Язык C# и платформа .Net - активно развивающиеся язык и платформа, благодаря чему в них реализована поддержка самых новых технологий
* Наличие подробной официальной документации.
* Широкие возможности языка и платформы: платформа включает в себя сборки, охватывающие все направления работы в разработке
* Автоматическое управление памятью (наличие механизма сборки мусора)
* Малый размер исполняемых модулей за счет того что основные библиотеки поставляются в составе .Net Framework
* Нацеленность, как языка, так и платформы на объектно-ориентированное программирование, в результате чего некоторые общепринятые паттерны реализуются на уровне платформы.

Нельзя не отметить также целый ряд преимуществ языка C#:

* Поддержка лямбда-выражений
* Компиляция осуществляется в промежуточный язык, благодаря чему программа кроссплатформенна
* Удобство разработки, благодаря понятности, взвешенности синтаксиса
* Безопасность кода, написанного на C#.
* Возможность определения типа исходя из инициализации объекта.
* Реализация инкапсуляции данных с помощью концепции свойств объектов.

## 1.3 Обоснование выбора технологии

Для построения графического интерфейса пользователя C# .Net предлагает на выбор несколько технологий:

* Windows Forms;
* Silverlight;
* Windows Presentation Foundation.

Для решения данной задачи была использована технология Windows Forms

Windows Forms представляет собой одну из технологий, используемую в Visual C# для создания интеллектуальных клиентских приложения на основе Windows, выполняемых в среде .NET Framework. Технология Windows Forms специально создана для быстрой разработки приложений, в которых обширный графический пользовательский интерфейс не является приоритетом. Для создания пользовательского интерфейса используется конструктор Windows Forms, и пользователь получает доступ к другим возможностям времени разработки и времени выполнения.

## 1.4 Проектирование архитектуры программного средства

*1.4.1 Базовые принципы объектно-ориентированного программирования*

Рассмотрим базовые принципы объектно-ориентированного программирования, учтенные при проектировании архитектуры разработанного приложения.

Абстракция — в объектно-ориентированном программировании это придание объекту характеристик, которые отличают его от всех других объектов, четко определяя его концептуальные границы. Основная идея состоит в том, чтобы отделить способ использования составных объектов данных от деталей их реализации в виде более простых объектов, подобно тому, как функциональная абстракция разделяет способ использования функции и деталей её реализации в терминах более примитивных функций, таким образом, данные обрабатываются функцией высокого уровня с помощью вызова функций низкого уровня. Такой подход является основой объектно-ориентированного программирования. Это позволяет работать с объектами, не вдаваясь в особенности их реализации.

Инкапсуляция — свойство языка программирования, позволяющее пользователю не задумываться о сложности реализации используемого программного компонента, а взаимодействовать с ним посредством предоставляемого интерфейс, а также объединить и защитить жизненно важные для компонента данные. При этом пользователю предоставляется только спецификация (интерфейс) объекта. Пользователь может взаимодействовать с объектом только через этот интерфейс. Реализуется с помощью ключевого слова: public. Пользователь не может использовать закрытые данные и методы. Реализуется с помощью ключевых слов: private, protected, internal. Сокрытие реализации целесообразно применять в следующих случаях: предельная локализация изменений при необходимости таких изменений, прогнозируемость изменений (какие изменения в коде надо сделать для заданного изменения функциональности) и прогнозируемость последствий изменений.

Наследование — один из важнейших механизмов объектно-ориентированного программирования, позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом. Другими словами, класс-наследник реализует спецификацию уже существующего класса (базовый класс). Это позволяет обращаться с объектами класса-наследника точно так же, как с объектами базового класса. Простое наследование Класс, от которого произошло наследование, называется базовым или родительским. Классы, которые произошли от базового, называются потомками, наследниками или производными классам. В некоторых языках используются абстрактные классы. Абстрактный класс — это класс, содержащий хотя бы один абстрактный метод, он описан в программе, имеет поля, методы и не может использоваться для непосредственного создания объекта. То есть от абстрактного класса можно только наследовать. Объекты создаются только на основе производных классов, наследованных от абстрактного.

Полиморфизм — возможность объектов с одинаковой спецификацией иметь различную реализацию. Язык программирования поддерживает полиморфизм, если классы с одинаковой спецификацией могут иметь различную реализацию — например, реализация класса может быть изменена в процессе наследования. Полиморфизм позволяет писать более абстрактные программы и повысить коэффициент повторного использования кода. Общие свойства объектов объединяются в систему, которую могут называть по-разному — интерфейс, класс. Общность имеет внешнее и внутреннее выражение: внешняя общность проявляется как одинаковый набор методов с одинаковыми именами и сигнатурами (именем методов и типами аргументов и их количеством); внутренняя общность — одинаковая функциональность методов. Её можно описать интуитивно или выразить в виде строгих законов, правил, которым должны подчиняться методы. Возможность приписывать разную функциональность одному методу (функции, операции) называется перегрузкой метода (перегрузкой функций, перегрузкой операций).

## 1.5 Нефункциональные требования

В результате анализа существующих аналогов разработанной игры, а также предпочтений пользователей, были сформулированы следующие нефункциональные требования:

* приложение должно использоваться как развлечение;
* соответствие графики нынешним возможностям;
* интерфейс меню приложения должен быть интуитивно понятным;
* пользователь должен иметь возможность начать использование приложения без поиска дополнительной информации;
* программа должна выполнять только те действия, которые запрашивает пользователь;
* приложение должно быть портативными;
* приложение должно быть эффективным с точки зрения использования системных ресурсов;
* необходимо сопроводить программное средство подробной документацией, содержащей руководство по его применению;
* приложение должно вызывать желание играть снова и снова.

# 2 МОДЕЛЬ ДАННЫХ И ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

## 2.1 Функциональные требования к программному средству

В результате анализа существующих аналогов, а также мнений и отзывов пользователей о похожих приложениях, были сформулированы требования к функциональности разрабатываемого приложения.

Согласно определенным требованиям, приложение должно предоставлять пользователю следующие возможности:

возможность ознакомления с правилами игры;

* возможность выбора одиночной игры против компьютера или игры пользователя против пользователя;
* возможность лёгкого управления игровым объектом с помощью мыши;
* возможность выбора пунктов главного меню посредством компьютерной мыши;
* возможность моментального выхода в любой момент игры;
* возможность повторного запуска игры, не перезапуская приложение заново;
* возможность выбора действия после запуска приложения (играть, посмотреть правила, информацию или выйти);
* возможность испытывать азарт во время игры для победы над оппонентом.

## 2.2 Диаграмма вариантов использования

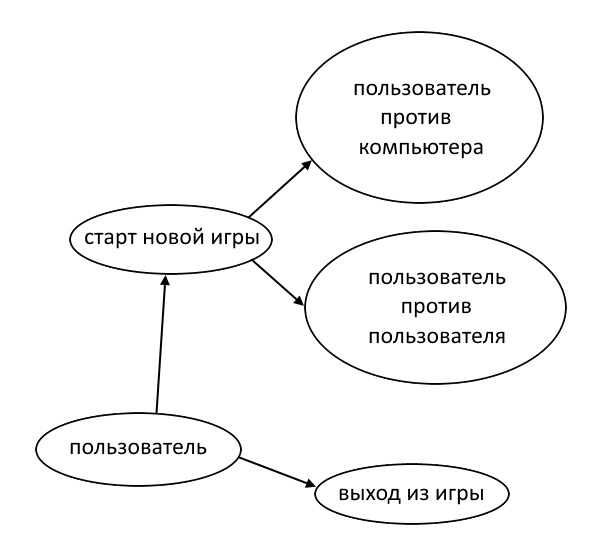


Рисунок 2.1 Use Case диаграмма

# 3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## 3.1 Разработка графического интерфейса пользователя

Для создания пользовательского интерфейса была применена технология Windows Forms.

## 3.2 Enum Opponents

enum Opponents

{

Bot,

Player

}

Используется для задания значения глобальной переменной, которая хранит выбор оппонента при запуске новой игры.

## 3.3 Enum CardOwner

enum CardOwner

{

First,

Second,

None

}

Используется для определения принадлежности карты игроку. Поле None означает что карта еще не была выбрана и находится в колоде.

## 3.4 Enum CardStatus

enum CardStatus

{

NotInGame,

Player,

Table,

OutOfGame

}

Используется для упрощения представления перемещения карты в ходе игрового процесса.

* NotInGame - карта еще не была задействована в игре и находится в колоде
* Player - карта находится у одного из игроков
* Table - карта была выбрана одним из игроков и ее положили на стол
* OutOfGame - карта после завершения хода была перемещена одному из игроков.

## 3.5 Class CardClass

Используется в качестве основного объекта взаимодействия. Содержит свойства для чтения:

**Value** - очки футболиста

**Nation** - национальность футболиста

Методы:

**MakeFrontImage** - загружает лицо футболиста, флаг, фон, очки и комбинирует полученное на один Image.

**LoadBackImage** - загружает «рубашку» карты и присваивает загруженный Image в соответствующий PictureBox.

**LoadFrontImage** - присваивает результат MakeFrontImage в соответствующий PictureBox.

**SetImagePosition** - устанавливает координаты PictureBox.

**SetVisible** - устанавливает видимость игровой стороны карты.

**CardClicked** - обработчик нажатия на карту.

## 3.6 Реализация основных функций

*3.6.1 ChangeTurn*

Данная функция принимает в качестве параметра переменную, которая хранит игрока, который сейчас ходил, и возвращает измененную переменную, в которой записан игрок, чей ход наступил. При этом, в зависимости от режим игры, выполняются некоторые дополнительные операции с картами для их корректного отображения.

Players ChangeTurn(Players item)

{

…

item = item == Players.First ? Players.Second : Players.First;

…

return item;

}

*3.6.2 MoveCurrentCard*

Осуществляет перемещение выбранной карты на стол, с проверкой возможности выбора, а также корректным размещением на столе.

void MoveCurrentCard(CardClass item)

{

…

++cardsChosen;

cardsOnTable[tableIndex] = item;

item.SetImagePosition(…);

item.ImageBox.BringToFront();

item.Status = CardStatus.Table;

turn = ChangeTurn(turn);

if (cardsChosen > 1)

CompareCards();

}

*3.6.3 GetNewCard*

При выборе карты и перенесении ее на стол происходит вызов данной функции, которая помещает верхнюю карту колоды на место выбранной.

void GetNewCard(CardClass item)

{

…

var index = item.PlayerOwner == CardOwner.First

? firstPlayerGameCards.IndexOf(item)

: secondPlayerGameCards.IndexOf(item);

if (item.PlayerOwner == CardOwner.Second)

{

cards[indexArray[counter]].SetImagePosition(…);

cards[indexArray[counter]].PlayerOwner = CardOwner.Second;

secondPlayerGameCards[index] = cards[indexArray[counter]];

cards[indexArray[counter]].SetVisible(opponent == Opponents.Player);

}

else

{

cards[indexArray[counter]].SetImagePosition(…);

cards[indexArray[counter]].PlayerOwner = CardOwner.First;

firstPlayerGameCards[index] = cards[indexArray[counter]];

cards[indexArray[counter]].SetVisible(true);

}

cards[indexArray[counter]].ImageBox.BringToFront();

cards[indexArray[counter]].Status = CardStatus.Player;

++counter;

}

*3.6.4 CompareCardsOnTable*

Функция сравнивает текущие карты игроков на столе и возвращает в качестве значения игрока, чья сумма больше, либо None, в случае, когда значения карт равны.

Players CompareCardsOnTable()

{

//player1, player2 - суммы первого и второго игроков

if (player1 > player2)

result = Players.First;

else

if (player2 > player1)

result = Players.Second;

else

result = Players.None;

return result

}

*3.6.5 RandomPlayerCards*

Раздает по 3 карты каждому игроку и по наибольшей сумме определяет победителя, который забирает все 8 карт.

void RandomCards()

{

MessageBox.Show("Penalties", "Football");

int firstPlayerSumma = 0;

int secondPlayerSumma = 0;

foreach (CardClass card in firstPlayerGameCards)

card.ImageBox.Visible = false;

foreach (CardClass card in secondPlayerGameCards)

card.ImageBox.Visible = false;

for (int i = 0; i < 4; i++)

if (cardsOnTable[i] != null)

cardsOnTable[i].ImageBox.Visible = false;

… // раздача и отображение карт

MessageBox.Show("First" + firstPlayerSumma.ToString() +"\nSecond"+secondPlayerSumma.ToString(), "Football");

if (firstPlayerSumma > secondPlayerSumma)

{

MessageBox.Show("First win", "Football");

…

}

else if (secondPlayerSumma > firstPlayerSumma)

{

MessageBox.Show("Second win", "Football");

…

}

else

{

MessageBox.Show("Round draw", "Football");

…

}

}

*3.6.6 CheckOtherPlayerCards*

Проверяет карты другого игрока на наличие возможности доложить карту с выбранной национальностью. False в случае отсутствия карты.

bool CheckOtherPlayerCards(string nation, Players turn)

{

bool result = false;

…

foreach (var card in PlayerGameCards)

{

if (nation == card.Nation)

result = true;

}

…

return result;

}

*3.6.7 CompareCards*

Основная функция сравнения карт, которая определяет победителя в раунде.

*3.6.8 StartNewRound*

Некоторые действия по очистке поля, размещению карт, наращиванию счетчиков.

void StartNewRound()

{

…

if (roundWinner != turn)

turn = ChangeTurn(turn);

++round;

…

if (round == NumberOfRounds)

{

MessageBox.Show("Final round", "Football");

FinalGame();

}

}

*3.6.9 FinalGame*

После 18 раундов происходит вызов функции FinalGame. Карты, которые игроки смогли заполучить, сортируются по убыванию, после чего 11 лучших выкладываются на стол. Победителем считается игрок, набравший наибольшую сумму очков игроков и количества карт.

void FinalGame()

{

firstPlayerOutCards.Sort(Compare);

secondPlayerOutCards.Sort(Compare);

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

{

firstSumma += firstPlayerOutCards[i].Value;

…

}

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

{

secondSumma += secondPlayerOutCards[i].Value;

…

}

firstSumma += firstPlayerOutCards.Count;

secondSumma += secondPlayerOutCards.Count;

…

if (firstSumma > secondSumma)

MessageBox.Show("First win game", "Football");

else if (secondSumma > firstSumma)

MessageBox.Show("Second win game", "Football");

else

{… //Определение наибольшего значения}

…

}

*3.6.10 Bot*

Данная функция позволяет имитировать человеческий выбор при игре с компьютером. Если компьютер ходит первый, то ход будет псевдослучайным, иначе компьютер будет оценивать ход человека, смотреть возможность выиграть раунд. При возможности выиграть компьютер сделает это с максимальной картой, чтобы увеличить сои шансы в финальном раунде. При невозможности выиграть он постарается запустить серию пенальти, иначе скинет самую маленькую карту.

# 4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

|  |  |  |
| --- | --- | --- |
| Команда | Действие пользователя | Результат действия пользователя |
| Запуск программы | Пользователь должен запустить приложение | На экране появляется рабочее окно с встроенным меню. |
| Запуск игры | Выбрать в меню пункт “Game” -> “New” | Появление диалогового окна с предложением выбрать режим игры. |
| Выбор оппонента | Выбрать подходящего соперника | В зависимости от выбора пользователя запускается либо одиночная игра против компьютера, либо игра пользователя против пользователя |
| Нажатие на карту | Нажатие мышкой на выбранную карту | Выбранная карта перемещается в центр стола и, если там уже есть карта, то происходит сравнение карт, иначе ходит оппонент. |
| Запуск серии пенальти | Выбор карты, которая совпадает по значению либо национальности с картой оппонента | Выдается предупреждение о старте серии пенальти, и происходит раздача трех карт каждому игроку. Победа присуждается игроку набравшему наибольшую сумму очков. |
| Удвоение карты | Наличие нескольких карт одной национальности и ход одной из них | Вслучае, если у пользователя имеется две и более карт одной национальности, он ходит первым, карта оппонента больше чем выбранная, то пользователь имеет возможность доложить карту такой же национальности либо отменить удвоение карты нажав активную кнопку “Cancel” и проиграть раунд. Данной правило справедливо и для оппонента. |
| Финальный раунд | Пользователь сыграл 18 раундов | После 18 раундов игры запускается сортировка выигранных карт по убыванию. Каждый игрок автоматически получает для сравнения 11 лучших карт (если он выиграл 11 и более карт). Далее общее число выигранных карт суммируется со значениями 11 лучших карт и сравниваются с суммой оппонента. Побеждает обладатель наибольшей суммы. |
| Предложение сыграть еще раз | Принятие решения об еще одной игре | Выдается диалоговое окно, предлагающее сыграть еще раз. В случае отрицательного ответа, выдается диалоговое окно с предложением закрыть игру. |
| Показ информации | Выбор в меню пункта “About” | Показ информации о продукте в диалоговом окне |
| Показ правил | Выбор в меню пункта “How to play” | Показ подробных правил игры “Карточный футбол” |

# 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для начала откроем приложение. После открытия первым делом появляется окно, которое изображено на рисунке 5.1.

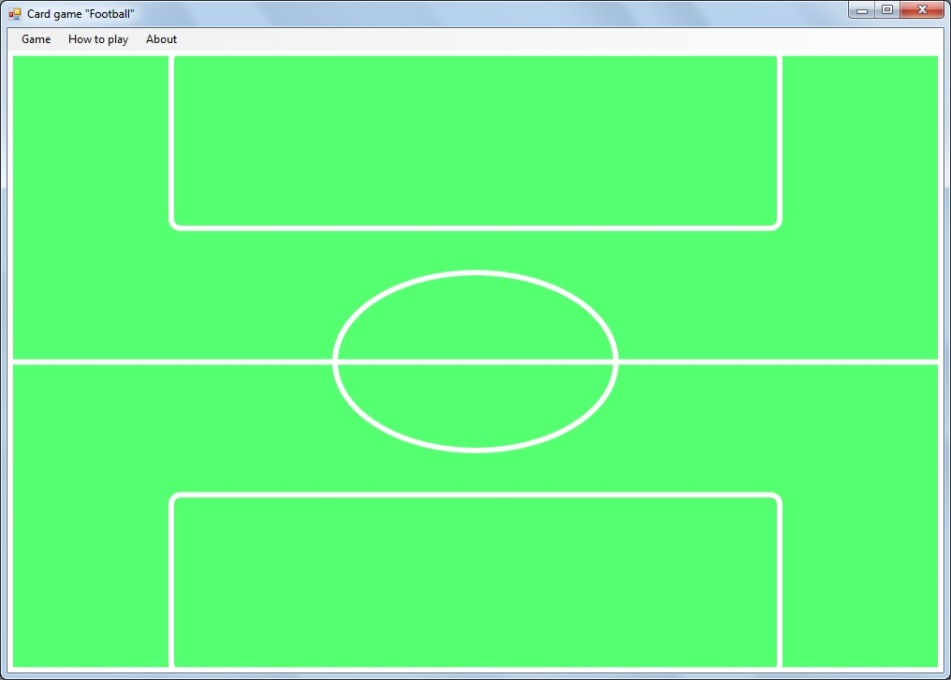


Рисунок 5.1 - Игровое поле

В нем есть 3 пункта меню “Game”, “How to play” и “About”.

Для получения сведений о продукте нужно нажать пункт меню “About”, после чего появиться диалоговое окно, которое можно видеть на рисунке 5.2. На нем краткая информация о продукте и создателе.

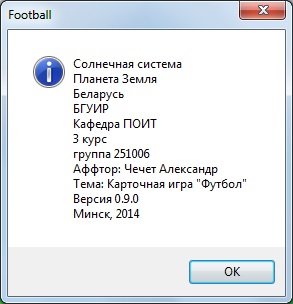


Рисунок 5.2 - Информация о продукте

Перед началом игры следует ознакомиться с правилами. Для получения информации по игровому процессу необходимо выбрать пункт меню “How to play”. На рисунке 5.3 показаны правила игры в диалоговом окне.

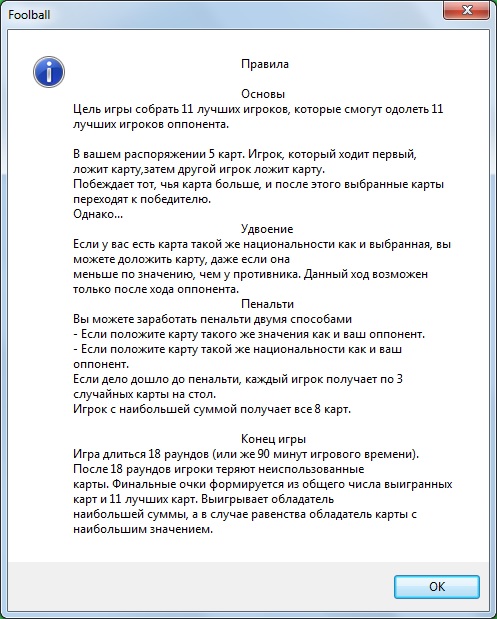


Рисунок 5.3 - Правила

Для запуска новой игры нужно выбрать пункт меню “Game” -> “New”. Появится диалоговое окно (Рисунок 5.4) с предложение выбрать оппонента. После выбора оппонента произойдет раздача карт и начнется первый раунд (Рисунок 5.5).

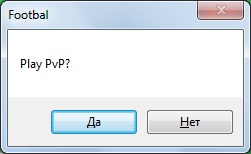


Рисунок 5.4 - Выбор оппонента

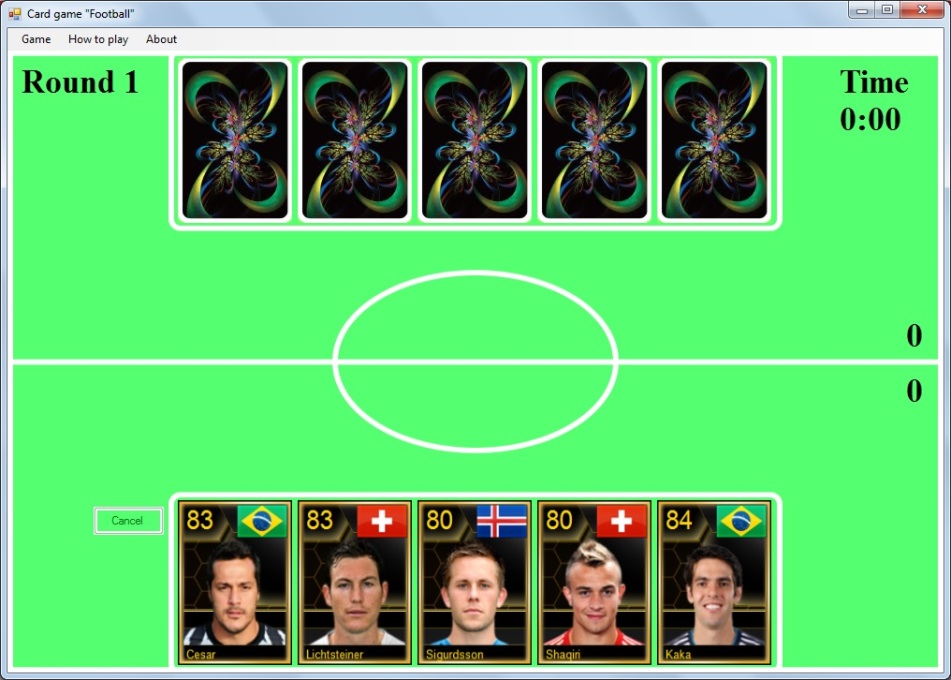


Рисунок 5.5 - Начало игры

Первый ход делает пользователь. В зависимости от выбранного режима, второй ход будет осуществлять либо компьютер, либо другой пользователь. Для совершения хода нажмите левой клавишей мыши на выбранную карту, она автоматически переместится на центр стола, а ее место займет карта из колоды. После хода (-ов) оппонента определяется победитель раунда и все карты переходят к нему(Рисунок 5.6).

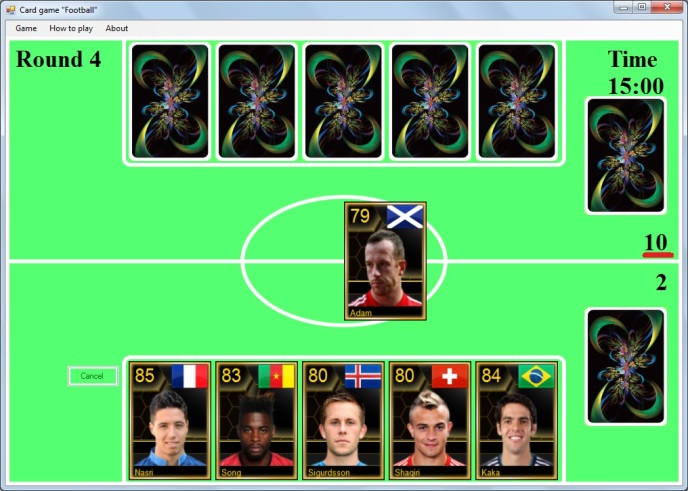
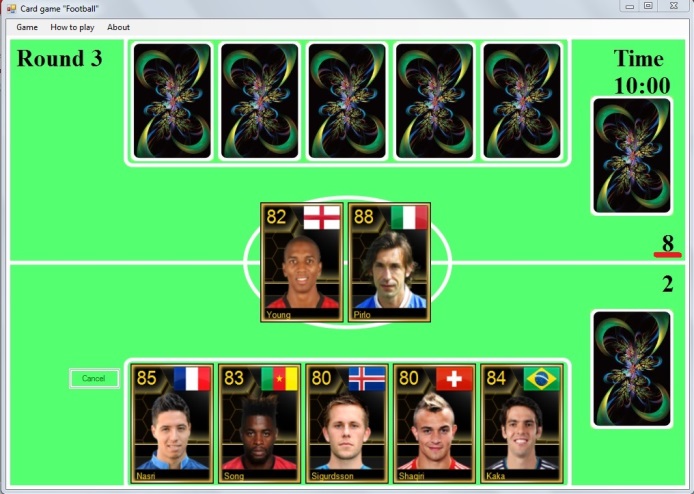




Рисунок 5.6 - Результат одного из раундов

Победитель игры определяется после 18 раундов по набранной финальной сумме (Рисунок 5.7).

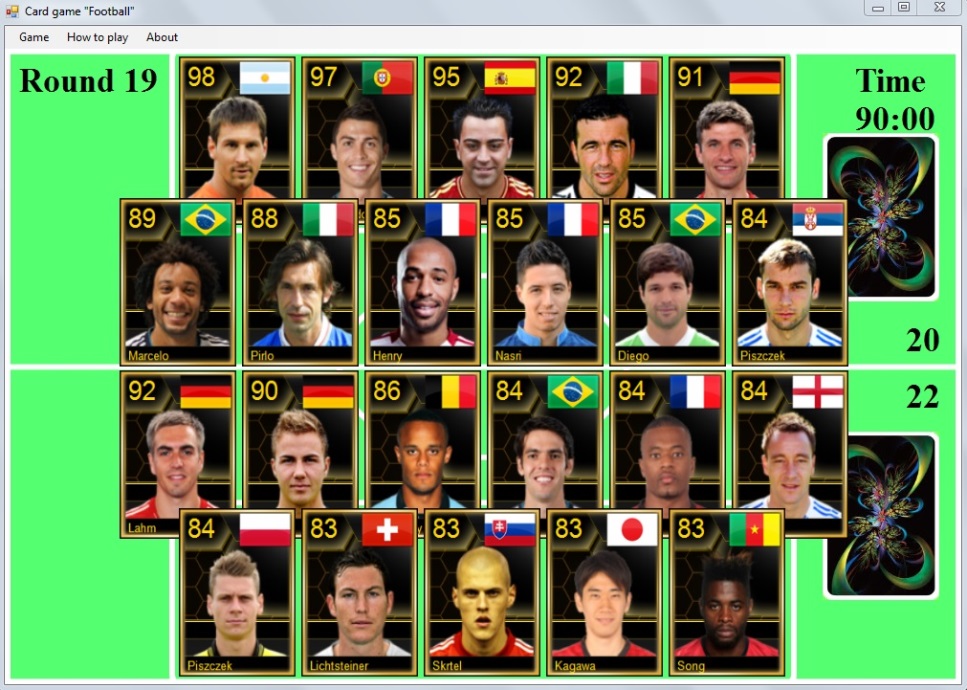




Рисунок 5.7 - Результат финального раунда

Пользователю предлагается сыграть еще раз и, в случае отрицательного ответа на первое предложение, выйти из игры (Рисунок 5.8).

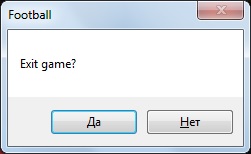
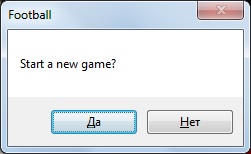


Рисунок 5.8 - Предложение начать новую игру и выйти из игры

# ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта было разработано игровое программное средство «Карточный футбол. Предполагает использование в развлекательных целях.

Данное программное средство поддерживает два режима игры.

1. Режим «Человек-человек». Этот режим предназначен для игры двух живых соперников.
2. Режим «Человек-компьютер». Этот режим предназначен для игры человека против искусственного интеллекта.

В дальнейшем в программное средство могут быть внесены указанные далее изменения.

1. Возможность игры на разных машинах с использование локальной или глобальной сети.
2. Сохранение игровой статистики в базе данных. Это позволит составить рейтинг игроков.
3. Сохранение текущего состояния игрового положения и загрузка его для продолжения игры.

В ходе выполнения курсовой работы были закреплены знания о языке программирования C#, интегрированной среде разработки Visual Studio 2013 и библиотеке графического интерфейса пользователя WinForms.

Приложение было протестировано на различных действиях, которые может совершить пользователь, в результате чего были получены результаты, соответствующие ожидаемым.

Полученное в результате работы приложение является полностью портативной, т.е. не требует установки и может быть запущено со съемного носителя.

В процессе работы над проектом было составлено подробное руководство пользователя, содержащее инструкции по использованию всех возможностей приложения, снабженное иллюстрациями и пояснениями.

На основании вышеперечисленного можно утверждать, что созданное приложение соответствует всем поставленным условиям и в процессе разработки были учтены все требования, сформулированные при постановке задачи.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рихтер. Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд.: Пер. с англ. / Д. Рихтер – СПб.: Питер, 2014 – 896 с.
2. Электронный учебно-методический комплекс дисциплины «Языки программирования» [Электронный ресурс]. – Электрон. дан. (6 Мб). – Минск: Бахтизин В.В., 2011.
3. Макконнелл, С. Совершенный код. Мастер-класс: Пер. с англ. / С. Макконнелл. – М.: «Русская редакция», 2010. – 896 с.
4. Кнут Д.Э. Искусство программирования: в 3 т. / Д. Э. Кнут. – 3-изд. – М.: Изд. Дом «Вильямс», 2006. – Т. 1: Основные алгоритмы. – 720 c.
5. Шилдт, Г. С# 4.0: полное руководство.: Пер. с англ. / Г. Шилтд. – М.: Изд. Дом «Вильямс», 2011. – 1056 с.
6. Троелсен Э., Язык программирования C# 5.0 и платформа .NET 4.5. Издание 6 - Пер. с англ. М.: Издательский дом «Вильямс», 2013. - 1321 с.
7. MSDN [Электронный ресурс] - Электронные данные - Режим доступа: <http://msdn.microsoft.com/en-us/library>

# ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД ПРОГРАММЫ

CardClass.cs

using System;

using System.Diagnostics;

using System.Drawing;

using System.Globalization;

using System.Windows.Forms;

namespace CardGameFoolball

{

class CardClass

{

public CardClass(string name, string nation, int numberToLoad, int value)

{

this.name = name;

Value = value;

Nation = nation;

this.numberToLoad = numberToLoad;

Status = CardStatus.NotInGame;

PlayerOwner = CardOwner.None;

ImageBox = new PictureBox();

ImageBox.Width = 122;

ImageBox.Height = 176;

ImageBox.SizeMode = PictureBoxSizeMode.StretchImage;

ImageBox.MouseClick += ImageMouseClick;

}

public void SetImagePosition(int x, int y)

{

ImageBox.Top = y;

ImageBox.Left = x;

}

public void SetVisible(bool b)

{

Visible = b;

if (Visible)

LoadFrontImage();

else

LoadBackImage();

}

public static void SetLocalPath(string path)

{

LocalPath = path + "\\Resources\\";

}

public delegate void MoveCard(CardClass item);

public event MoveCard OnCardClick;

public PictureBox ImageBox;

public CardStatus Status { get; set; }

public int Value { get; private set; }

public string Nation { get; private set; }

public CardOwner PlayerOwner { get; set; }

public static string LocalPath { get; private set; }

private Bitmap MakeFrontImage()

{

Bitmap bitmapPlayer = new Bitmap(LocalPath + "Players\\" + numberToLoad + ".png");

Bitmap bitmapFlag = new Bitmap(LocalPath + "Flags\\" + Nation + ".png");

Bitmap finalBitmap = new Bitmap(LocalPath + "CardFon.png");

Font font = new Font("Arial", 22);

Graphics graph = Graphics.FromImage(finalBitmap);

graph.DrawImage(bitmapPlayer, new Rectangle(7, 40, 120, 120));

graph.DrawImage(bitmapFlag, new Rectangle(70, 5, 60, 37));

graph.DrawString(Value.ToString(CultureInfo.InvariantCulture), font, Brushes.Gold, new Point(5, 5));

graph.DrawString(name, new Font("Arial", 10), Brushes.Gold, new Rectangle(8, 162, 100, 25));

return finalBitmap;

}

private void LoadFrontImage()

{

ImageBox.Image = MakeFrontImage();

}

private void LoadBackImage()

{

ImageBox.Load(LocalPath + "CardBack.png");

Color temp = ColorTranslator.FromOle(0x54FF70);

ImageBox.BackColor = Color.FromArgb(250, temp);

}

private bool Visible { get; set; }

private void ImageMouseClick(object sender, EventArgs e)

{

CardClicked();

}

public void CardClicked()

{

switch (Status)

{

case CardStatus.Player:

Debug.Assert(OnCardClick != null, "OnCardClick != null");

OnCardClick(this);

break;

}

}

private readonly int numberToLoad;

private readonly string name;

}

}

CardOwner.cs

namespace CardGameFoolball

{

enum CardOwner

{

First,

Second,

None

}

}

CardStatus.cs

namespace CardGameFoolball

{

enum CardStatus

{

NotInGame,

Player,

Table,

OutOfGame

}

}

MainForm.cs

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Globalization;

using System.IO;

using System.Windows.Forms;

namespace CardGameFoolball

{

public partial class MainForm : Form

{

private Opponents opponent;

private const int NumberOfRounds = 18;

private int cardsChosen;

private int counter;

private int round;

private readonly Random rand = new Random();

private Players roundWinner;

private bool firstPlayerBlocked;

private bool secondPlayerBlocked;

private Players turn = Players.First;

private short[] indexArray = new short[150];

private string gameRules;

private string about;

private readonly CardClass[] cards = new CardClass[150];

private readonly CardClass[] cardsOnTable = new CardClass[4];

private readonly List<CardClass> firstPlayerGameCards = new List<CardClass>();

private readonly List<CardClass> firstPlayerOutCards = new List<CardClass>();

private readonly List<CardClass> randomGameCards = new List<CardClass>();

private readonly List<CardClass> secondPlayerGameCards = new List<CardClass>();

private readonly List<CardClass> secondPlayerOutCards = new List<CardClass>();

public MainForm()

{

InitializeComponent();

CardClass.SetLocalPath(Path.GetDirectoryName(Application.ExecutablePath));

}

private void MainForm\_Load(object sender, EventArgs e)

{

gameRules = File.ReadAllText(CardClass.LocalPath + "GameRules.txt");

about = File.ReadAllText(CardClass.LocalPath + "About.txt");

var str = new StreamReader(CardClass.LocalPath + "Info.txt");

for (var i = 0; i < 150; i++)

{

var line = str.ReadLine();

Debug.Assert(line != null, "line != null");

var data = line.Split(';');

var value = 0;

var name = data[0];

try

{

value = int.Parse(data[1]);

}

catch

{

MessageBox.Show("Error parse");

}

var nation = data[2];

cards[i] = new CardClass(name, nation, i + 1, value);

cards[i].OnCardClick += GetNewCard;

cards[i].OnCardClick += MoveCurrentCard;

cards[i].OnCardClick += ActivateBot;

Controls.Add(cards[i].ImageBox);

}

skip1.Enabled = false;

skip2.Enabled = false;

skip1.Visible = false;

skip2.Visible = false;

RoundLabel.Visible = false;

TimeLabel.Visible = false;

NumberOutCards1Label.Visible = false;

NumberOutCards2Label.Visible = false;

}

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

{

opponent = MessageBox.Show("Play PvP?", "Footbal", MessageBoxButtons.YesNo) == DialogResult.Yes ? Opponents.Player : Opponents.Bot;

firstPlayerGameCards.Clear();

secondPlayerGameCards.Clear();

firstPlayerOutCards.Clear();

secondPlayerOutCards.Clear();

indexArray = MakeRandom();

counter = 0;

round = 0;

for (int i = 0; i < 150; i++)

cards[i].SetImagePosition(-200, -200);

for (int i = 0; i < 10; i++)

{

var cardItem = cards[indexArray[counter]];

if (i%2 == 0)

firstPlayerGameCards.Add(cardItem);

else

secondPlayerGameCards.Add(cardItem);

++counter;

}

var count = 0;

foreach (var item in firstPlayerGameCards)

{

item.SetImagePosition(182 + count\*128, 504);

item.SetVisible(false);

item.ImageBox.BringToFront();

item.Status = CardStatus.Player;

item.PlayerOwner = CardOwner.First;

++count;

}

count = 0;

foreach (var item in secondPlayerGameCards)

{

item.SetImagePosition(182 + count\*128, 32);

item.SetVisible(false);

item.ImageBox.BringToFront();

item.Status = CardStatus.Player;

item.PlayerOwner = CardOwner.Second;

++count;

}

skip1.Enabled = false;

skip2.Enabled = false;

skip2.Visible = false;

skip1.Visible = true;

turn = Players.Second;

turn = ChangeTurn(turn);

RoundLabel.Visible = true;

TimeLabel.Visible = true;

NumberOutCards1Label.Visible = true;

NumberOutCards2Label.Visible = true;

RoundLabel.Text = "Round " + (round + 1);

TimeLabel.Text = "Time\n" + (round\*5) + ":00";

NumberOutCards1Label.Text = firstPlayerOutCards.Count.ToString(CultureInfo.InvariantCulture);

NumberOutCards2Label.Text = secondPlayerOutCards.Count.ToString(CultureInfo.InvariantCulture);

if (opponent == Opponents.Bot)

{

foreach (var card in firstPlayerGameCards)

card.SetVisible(true);

secondPlayerBlocked = true;

}

}

private void skip1\_Click(object sender, EventArgs e)

{

turn = ChangeTurn(turn);

CompareCards();

}

private void skip2\_Click(object sender, EventArgs e)

{

turn = ChangeTurn(turn);

CompareCards();

}

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

{

if (MessageBox.Show("Exit game?", "Footbal", MessageBoxButtons.YesNo) == DialogResult.Yes)

Application.Exit();

}

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

{

MessageBox.Show(gameRules, "Foolball", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

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

{

MessageBox.Show(about, "Football", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

}

}

MainFunctions.cs

using System.Globalization;

using System.Linq;

using System.Windows.Forms;

namespace CardGameFoolball

{

public partial class MainForm

{

private Players ChangeTurn(Players item)

{

if (opponent == Opponents.Player)

{

foreach (var card in firstPlayerGameCards)

card.SetVisible(false);

foreach (var card in secondPlayerGameCards)

card.SetVisible(false);

skip1.Visible = false;

skip2.Visible = false;

if (item == Players.First)

{

MessageBox.Show("Player 2 turn", "Football");

foreach (var card in secondPlayerGameCards)

card.SetVisible(true);

skip2.Visible = true;

item = Players.Second;

}

else

{

MessageBox.Show("Player 1 turn", "Football");

foreach (var card in firstPlayerGameCards)

card.SetVisible(true);

skip1.Visible = true;

item = Players.First;

}

}

else

{

item = item == Players.First ? Players.Second : Players.First;

}

return item;

}

private void MoveCurrentCard(CardClass item)

{

if ((turn == Players.First && item.PlayerOwner == CardOwner.First) ||

(turn == Players.Second && item.PlayerOwner == CardOwner.Second))

{

if ((item.PlayerOwner == CardOwner.First && !firstPlayerBlocked) ||

(item.PlayerOwner == CardOwner.Second &&

((!secondPlayerBlocked && opponent == Opponents.Player) ||

(secondPlayerBlocked && opponent == Opponents.Bot))))

{

int tableIndex;

if (item.PlayerOwner == CardOwner.First)

if (cardsOnTable[1] == null)

tableIndex = 1;

else

{

tableIndex = 0;

firstPlayerBlocked = true;

}

else

if (cardsOnTable[2] == null)

tableIndex = 2;

else

{

tableIndex = 3;

secondPlayerBlocked = true;

}

++cardsChosen;

cardsOnTable[tableIndex] = item;

item.SetImagePosition(244 + tableIndex \* 128, 268);

item.ImageBox.BringToFront();

item.Status = CardStatus.Table;

turn = ChangeTurn(turn);

if (cardsChosen > 1)

CompareCards();

}

}

}

private void GetNewCard(CardClass item)

{

if ((turn == Players.First && item.PlayerOwner == CardOwner.First) ||

(turn == Players.Second && item.PlayerOwner == CardOwner.Second))

{

if ((item.PlayerOwner == CardOwner.First && !firstPlayerBlocked) ||

(item.PlayerOwner == CardOwner.Second &&

((!secondPlayerBlocked && opponent == Opponents.Player) ||

(secondPlayerBlocked && opponent == Opponents.Bot))))

{

var index = item.PlayerOwner == CardOwner.First

? firstPlayerGameCards.IndexOf(item)

: secondPlayerGameCards.IndexOf(item);

if (item.PlayerOwner == CardOwner.Second)

{

cards[indexArray[counter]].SetImagePosition(182 + index\*128, 32);

cards[indexArray[counter]].PlayerOwner = CardOwner.Second;

secondPlayerGameCards[index] = cards[indexArray[counter]];

cards[indexArray[counter]].SetVisible(opponent == Opponents.Player);

}

else

{

cards[indexArray[counter]].SetImagePosition(182 + index\*128, 504);

cards[indexArray[counter]].PlayerOwner = CardOwner.First;

firstPlayerGameCards[index] = cards[indexArray[counter]];

cards[indexArray[counter]].SetVisible(true);

}

cards[indexArray[counter]].ImageBox.BringToFront();

cards[indexArray[counter]].Status = CardStatus.Player;

++counter;

}

}

}

private void CompareCards()

{

switch (cardsChosen)

{

case 2:

{

if (cardsOnTable[1].Nation == cardsOnTable[2].Nation ||

cardsOnTable[1].Value == cardsOnTable[2].Value)

{

RandomCards();

StartNewRound();

}

else

if (cardsOnTable[1].Value > cardsOnTable[2].Value)

if (turn == Players.Second && CheckOtherPlayerCards(cardsOnTable[2].Nation, Players.Second))

skip2.Enabled = true;

else

{

MessageBox.Show("First win", "Football");

roundWinner = Players.First;

StartNewRound();

}

else

if (turn == Players.First && CheckOtherPlayerCards(cardsOnTable[1].Nation, Players.First))

skip1.Enabled = true;

else

{

MessageBox.Show("Second win", "Football");

roundWinner = Players.Second;

StartNewRound();

}

break;

}

case 3:

{

if (CompareCardsOnTable() == Players.First)

if (turn == Players.Second && skip2.Enabled == false &&

CheckOtherPlayerCards(cardsOnTable[2].Nation, turn))

skip2.Enabled = true;

else

{

MessageBox.Show("First win", "Football");

roundWinner = Players.First;

StartNewRound();

}

else

if (CompareCardsOnTable() == Players.Second)

if (turn == Players.First && skip1.Enabled == false &&

CheckOtherPlayerCards(cardsOnTable[1].Nation, turn))

skip1.Enabled = true;

else

{

MessageBox.Show("Second win", "Football");

roundWinner = Players.Second;

StartNewRound();

}

break;

}

case 4:

{

if (CompareCardsOnTable() == Players.First)

{

MessageBox.Show("First win", "Football");

roundWinner = Players.First;

StartNewRound();

}

else

if (CompareCardsOnTable() == Players.Second)

{

MessageBox.Show("Second win", "Football");

roundWinner = Players.Second;

StartNewRound();

}

else

{

RandomCards();

StartNewRound();

}

break;

}

}

}

private void RandomCards()

{

MessageBox.Show("Penalties", "Football");

int firstPlayerSumma = 0;

int secondPlayerSumma = 0;

foreach (CardClass card in firstPlayerGameCards)

card.ImageBox.Visible = false;

foreach (CardClass card in secondPlayerGameCards)

card.ImageBox.Visible = false;

for (int i = 0; i < 4; i++)

if (cardsOnTable[i] != null)

cardsOnTable[i].ImageBox.Visible = false;

firstPlayerBlocked = true;

secondPlayerBlocked = true;

skip1.Visible = false;

skip2.Visible = false;

for (int i = 0; i < 6; i++)

{

cards[indexArray[counter]].SetVisible(true);

cards[indexArray[counter]].ImageBox.BringToFront();

cards[indexArray[counter]].Status = CardStatus.Table;

if (i % 2 == 0)

{

cards[indexArray[counter]].SetImagePosition(240 + i \* 100, 420);

randomGameCards.Add(cards[indexArray[counter]]);

firstPlayerSumma += cards[indexArray[counter]].Value;

}

else

{

cards[indexArray[counter]].SetImagePosition(240 + (i - 1) \* 100, 120);

randomGameCards.Add(cards[indexArray[counter]]);

secondPlayerSumma += cards[indexArray[counter]].Value;

}

++counter;

if (ActiveForm != null) ActiveForm.Refresh();

System.Threading.Thread.Sleep(1000);

}

MessageBox.Show("First " + firstPlayerSumma.ToString(CultureInfo.InvariantCulture) +

"\nSecond " + secondPlayerSumma.ToString(CultureInfo.InvariantCulture), "Football");

if (firstPlayerSumma > secondPlayerSumma)

{

MessageBox.Show("First win", "Football");

foreach (CardClass card in randomGameCards)

{

card.SetVisible(false);

card.Status = CardStatus.OutOfGame;

card.SetImagePosition(855, 424);

firstPlayerOutCards.Add(card);

}

randomGameCards.Clear();

roundWinner = Players.First;

}

else if (secondPlayerSumma > firstPlayerSumma)

{

MessageBox.Show("Second win", "Football");

foreach (CardClass card in randomGameCards)

{

card.SetVisible(false);

card.Status = CardStatus.OutOfGame;

card.SetImagePosition(855, 112);

secondPlayerOutCards.Add(card);

}

randomGameCards.Clear();

roundWinner = Players.Second;

}

else

{

MessageBox.Show("Round draw", "Football");

foreach(CardClass card in randomGameCards)

{

card.SetVisible(false);

card.Status = CardStatus.OutOfGame;

card.SetImagePosition(-200, -200);

}

randomGameCards.Clear();

roundWinner = Players.None;

}

foreach (CardClass card in firstPlayerGameCards)

card.ImageBox.Visible = true;

foreach (CardClass card in secondPlayerGameCards)

card.ImageBox.Visible = true;

for (int i = 0; i < 4; i++)

if (cardsOnTable[i] != null)

cardsOnTable[i].ImageBox.Visible = true;

if (opponent == Opponents.Bot)

skip1.Visible = true;

}

private bool CheckOtherPlayerCards(string nation, Players turn)

{

bool result = false;

if (turn == Players.First)

foreach (var card in firstPlayerGameCards)

{

if (nation == card.Nation)

result = true;

}

else

foreach (var card in secondPlayerGameCards)

{

if (nation == card.Nation)

result = true;

}

return result;

}

private Players CompareCardsOnTable()

{

Players result;

int player1 = cardsOnTable[1].Value;

int player2 = cardsOnTable[2].Value;

if (cardsOnTable[0] != null)

player1 += cardsOnTable[0].Value;

if (cardsOnTable[3] != null)

player2 += cardsOnTable[3].Value;

if (player1 > player2)

result = Players.First;

else

if (player2 > player1)

result = Players.Second;

else

result = Players.None;

return result;

}

private void StartNewRound()

{

for (int i = 0; i < 4; i++)

if (cardsOnTable[i] != null)

{

cardsOnTable[i].SetVisible(false);

cardsOnTable[i].Status = CardStatus.OutOfGame;

if (roundWinner == Players.First)

{

firstPlayerOutCards.Add(cardsOnTable[i]);

cardsOnTable[i].SetImagePosition(855, 424);

}

else if (roundWinner == Players.Second)

{

secondPlayerOutCards.Add(cardsOnTable[i]);

cardsOnTable[i].SetImagePosition(855, 112);

}

else

{

cardsOnTable[i].SetImagePosition(-200, -200);

}

cardsOnTable[i] = null;

}

cardsChosen = 0;

firstPlayerBlocked = false;

if (opponent == Opponents.Player)

secondPlayerBlocked = false;

skip1.Enabled = false;

skip2.Enabled = false;

if (roundWinner != turn)

turn = ChangeTurn(turn);

++round;

RoundLabel.Text = string.Format("Round {0}", (round + 1));

TimeLabel.Text = string.Format("Time\n{0}:00", (round \* 5));

NumberOutCards1Label.Text = firstPlayerOutCards.Count.ToString(CultureInfo.InvariantCulture);

NumberOutCards2Label.Text = secondPlayerOutCards.Count.ToString(CultureInfo.InvariantCulture);

if (round == NumberOfRounds)

{

MessageBox.Show("Final round", "Football");

FinalGame();

}

else

ActivateBot(null);

}

private void FinalGame()

{

skip1.Visible = false;

skip2.Visible = false;

foreach(CardClass card in firstPlayerGameCards)

card.SetImagePosition(-200, -200);

foreach(CardClass card in secondPlayerGameCards)

card.SetImagePosition(-200, -200);

firstPlayerOutCards.Sort(Compare);

secondPlayerOutCards.Sort(Compare);

int firstIndex = 11;

if (firstPlayerOutCards.Count < 11)

firstIndex = firstPlayerOutCards.Count;

int secondIndex = 11;

if (secondPlayerOutCards.Count < 11)

secondIndex = secondPlayerOutCards.Count;

int firstSumma = 0;

int secondSumma = 0;

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

{

firstPlayerOutCards[i].SetVisible(true);

if (i < 6)

firstPlayerOutCards[i].SetImagePosition(120 + i \* 128, 360);

else

firstPlayerOutCards[i].SetImagePosition(182 + (i - 6) \* 128, 504);

firstSumma += firstPlayerOutCards[i].Value;

firstPlayerOutCards[i].ImageBox.BringToFront();

if (ActiveForm != null) ActiveForm.Refresh();

System.Threading.Thread.Sleep(1000);

}

if (ActiveForm != null) ActiveForm.Refresh();

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

{

secondPlayerOutCards[i].SetVisible(true);

if (i < 5)

secondPlayerOutCards[i].SetImagePosition(182 + i \* 128, 32);

else

secondPlayerOutCards[i].SetImagePosition(120 + (i - 5) \* 128, 180);

secondSumma += secondPlayerOutCards[i].Value;

secondPlayerOutCards[i].ImageBox.BringToFront();

if (ActiveForm != null) ActiveForm.Refresh();

System.Threading.Thread.Sleep(1000);

}

if (ActiveForm != null) ActiveForm.Refresh();

firstSumma += firstPlayerOutCards.Count;

secondSumma += secondPlayerOutCards.Count;

MessageBox.Show("First " + firstSumma + "\nSecond " + secondSumma, "Football");

if (firstSumma > secondSumma)

MessageBox.Show("First win game", "Football");

else if (secondSumma > firstSumma)

MessageBox.Show("Second win game", "Football");

else

{

var firstValue = firstPlayerOutCards.Max(x => x.Value);

var secondValue = secondPlayerOutCards.Max(x => x.Value);

if (firstValue > secondValue)

MessageBox.Show("First win game", "Football");

else if (firstValue < secondValue)

MessageBox.Show("Second win game", "Football");

else

MessageBox.Show("Draw game", "Foolball");

}

if (MessageBox.Show("Start a new game?", "Football", MessageBoxButtons.YesNo) == DialogResult.Yes)

newToolStripMenuItem\_Click(null, null);

else

{

if (MessageBox.Show("Exit game?", "Football", MessageBoxButtons.YesNo) == DialogResult.Yes)

Application.Exit();

}

}

private int Compare(CardClass x, CardClass y)

{

return y.Value.CompareTo(x.Value);

}

public short[] MakeRandom()

{

short[] s = new short[150];

const int number = 5;

int[] u = new int[number];

for (short i = 0; i < 150; i++)

s[i] = i;

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

u[i] = rand.Next();

int j = 0;

for (int i = 0; i < 150; i++)

{

j = (j + s[i] + u[i % number]) % 150;

short c = s[i];

s[i] = s[j];

s[j] = c;

}

int k = 0;

j = 0;

for (int i = 0; i < 150; i++)

{

k = (k + 1) % 150;

j = (j + s[k]) % 150;

short c = s[k];

s[k] = s[j];

s[j] = c;

}

return s;

}

private void Bot()

{

CardClass item = secondPlayerGameCards[0];

if (cardsChosen == 0)

item = secondPlayerGameCards[rand.Next(5)];

else if (cardsChosen == 1)

{

int value = 0;

int index = -1;

foreach (var card in secondPlayerGameCards)

if (value < card.Value)

{

value = card.Value;

index = secondPlayerGameCards.IndexOf(card);

}

if (index != -1)

item = secondPlayerGameCards[index];

else

{

value = 100;

index = -1;

foreach (var card in secondPlayerGameCards)

if (cardsOnTable[1].Nation == card.Nation)

if (card.Value < value)

index = secondPlayerGameCards.IndexOf(card);

if (index != -1)

item = secondPlayerGameCards[index];

else

{

value = 100;

index = -1;

foreach (var card in secondPlayerGameCards)

if (card.Value < value)

{

value = card.Value;

index = secondPlayerGameCards.IndexOf(card);

}

item = secondPlayerGameCards[index];

}

}

}

else if (cardsChosen == 2)

{

int value = 0;

int index = -1;

foreach (var card in secondPlayerGameCards)

{

if (card.Nation == cardsOnTable[2].Nation)

if (card.Value > value)

{

value = card.Value;

index = secondPlayerGameCards.IndexOf(card);

}

}

if (index != -1)

item = secondPlayerGameCards[index];

}

else if (cardsChosen == 3)

{

int value = 0;

int index = -1;

foreach (var card in secondPlayerGameCards)

{

if (card.Nation == cardsOnTable[2].Nation)

if (card.Value > value)

{

value = card.Value;

index = secondPlayerGameCards.IndexOf(card);

}

}

if (index != -1)

item = secondPlayerGameCards[index];

}

item.SetVisible(true);

item.CardClicked();

}

private void ActivateBot(CardClass cardClass)

{

if (turn == Players.Second && opponent == Opponents.Bot && round < NumberOfRounds)

Bot();

}

}

}

Opponents.cs

namespace CardGameFoolball

{

enum Opponents

{

Bot,

Player

}

}

Players.cs

namespace CardGameFoolball

{

public enum Players

{

First,

Second,

None

}

}