Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**Программное игровое средство «Brain train»**

|  |  |  |
| --- | --- | --- |
| Студент |  | А.В. Бобко |
| Руководитель |  | Г.В. Данилова |

Минск 2018

СОДЕРЖАНИЕ

[Введение 5](#_Toc513665878)

[1 Анализ предметной области 6](#_Toc513665879)

[1.1 Обзор аналогов 6](#_Toc513665880)

[1.2 Постановка задачи 8](#_Toc513665881)

[2 Разработка программного средства 9](#_Toc513665882)

[2.1 Структура программы 9](#_Toc513665883)

[2.2 Интерфейс программного средства 9](#_Toc513665884)

[2.3 Графические ресурсы 11](#_Toc513665885)

[2.4 Игровая логика 13](#_Toc513665888)

[2.4.1 Игра «Colors» 13](#_Toc513665886)

[2.4.2 Игра «Fruits» 15](#_Toc513665886)

[2.5 Работа с таблицей рекордов 17](#_Toc513665889)

[2.6 Диалоговое окно и окно ввода имени 19](#_Toc513665889)

[2.6.1 Диалоговое окно 19](#_Toc513665886)

[2.6.2 Окно ввода имени 20](#_Toc513665886)

[3 Тестирование программного средства 21](#_Toc513665890)

[4 Руководство пользователя 22](#_Toc513665891)

[4.1 Правила игры 22](#_Toc513665892)

[4.1.1 Игра «Colors» 22](#_Toc513665886)

[4.1.2 Игра «Fruits» 22](#_Toc513665886)

[4.2 Интерфейс программы 22](#_Toc513665893)

[Заключение 25](#_Toc513665894)

[Список использованных источников 26](#_Toc513665895)

[Приложение А. Исходный код программы 27](#_Toc513665896)

ВВЕДЕНИЕ

В повседневной жизни человек постоянно сталкивается с различной информацией. Каждый день люди пытаются держать в голове огромные списки дел, важные даты, мелкие бытовые вопросы, рецепты, инструкции. Также очень часто бывает необходимо запоминать большие объемы информации, различного рода формулы, речи для выступлений, определенные правила и законы. Кроме того, постоянно приходится сталкиваться с рекламой, которая встречается повсеместно: по радио, телевизору, в интернете, на улице – которая также остается у нас в голове и забивает нашу память.

У многих людей бывают такие моменты, когда они просто «зависают» посреди фразы, потому что забывают, что хотели сказать. Или же не могут вспомнить, куда положили ключи, кошелек или телефон. Часто люди ощущают, что их мыслительные способности с возрастом угасают. Время начинает работать против них. Однако ученые сообщают, что, если беречь здоровье и поддерживать умственную активность, то человеческих мозг будет легко справляться со своими задачами до глубокой старости. С возрастом наш мозг меняется. Коммуникация в нем осуществляется за счет прохождения электрических сигналов вдоль нервных волокон, изолированных жировой тканью, называемой миелином. При старении количество миелина снижается, и сигналы начинают проходить медленнее.

Некоторые люди с возрастом начинают страдать от провалов в памяти и различных нейродегенеративных нарушений, таких как болезнь Альцгеймера. Но учеными было доказано, что мозг продолжает адаптироваться и обновляться даже в зрелые годы. Например, когда люди становятся старше, мозг начинает использовать более обширные площади, чтобы поддержать свое функционирование. Исследования показывают, что поддержание мозга в активном состоянии противодействует естественным изменениям старения настолько эффективно, что многие люди вообще их не замечают. Люди могут поддерживать жизнедеятельность мозга и помогать себе сохранить здоровье, когда принимаемся за новую и трудную деятельность.

В интернете каждый может легко найти большое количество книг по улучшению памяти, в которых есть отличные подборки головоломок и упражнений. Но время – это крайне ценный ресурс и не у каждого человека получается выделить минуту на чтение такой литературы. Научно доказано, что человеку намного проще обучаться в процессе игры или соревнования.

Целью данного курсового проекта является анализ и разработка игрового приложения, содержащего проверенные временем игры, которые помогут развивать память и внимание и которые будут понятны, доступны и интересны как детям, так и взрослым.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ
   1. Обзор аналогов

В интернете можно встретить колоссальное количество игр, тренажеров, задачек и сайтов для тренировки и развития мозга. Далее будут приведены наиболее интересные примеры.

Сайт *Brainapps.ru* – один из популярных сайтов с множеством мини-игр и упражнений для мозга. На нем собрано огромное обилие материала для совершенствования памяти, внимания, стратегического мышления, а также детальная статистика, составленная на основе в аших результатов, полезные советы, курсы и тесты. Внешний вид сайта отображен на рисунке 1.1.

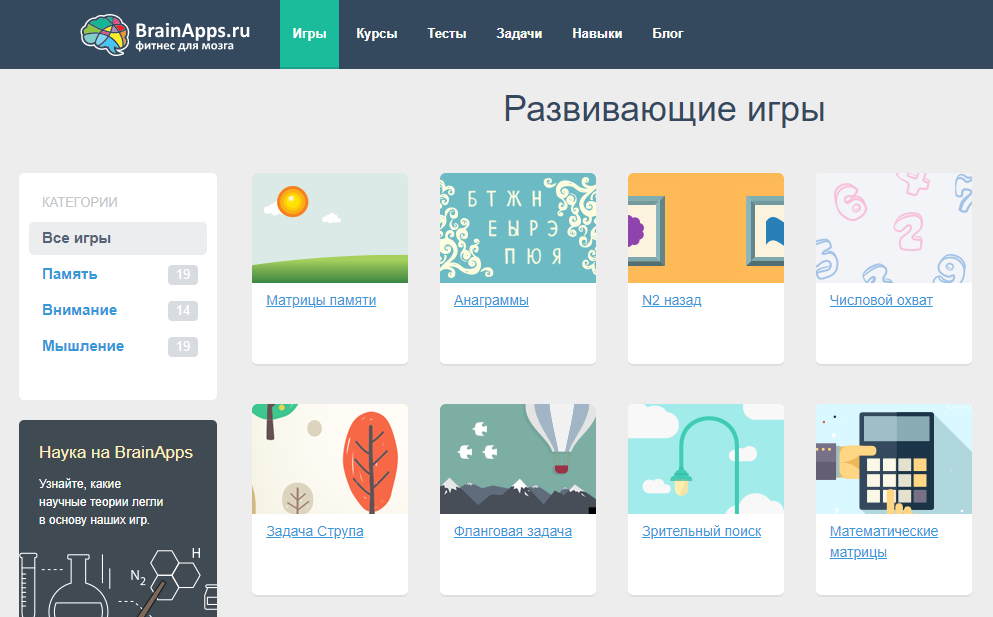


Рисунок 1.1 – Интерфейс сайта brainapps.ru

Из недостатков сайта можно выделить то, что для использования всех его функций необходимо приобретать ежемесячную или ежегодную премиум-подписку, а в стандартном режиме возможности сайта сильно урезаны. Из плюсов можно добавить отсутствие рекламы (кроме рекламы внутрисайтового контента).

Следующим аналогом является мобильное приложение «Cimon Says». Дизайн игры очень прост, как и ее механика, поэтому она не потребляет много ресурсов, а с правилами может разобраться как взрослый, так и каждый ребенок. Несмотря на свою простоту, данное приложение является эффективным инструментом для тренировки, использование которого поможет вам быстро запоминать обширные объёмы информации.

В данной игре присутствует таблица рекордов, что дает ее пользователям дух соперничества. Внешний вид окна данного приложения представлен на рисунке 1.2.

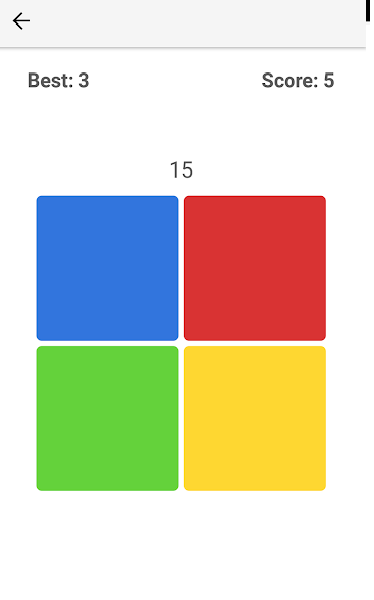


Рисунок 1.2 – Интерфейс игры «Cimon Says»

Еще одним интересным аналогом является мобильное приложение «Найди пару». Внешний вид приложения представлен на рисунке 1.3.

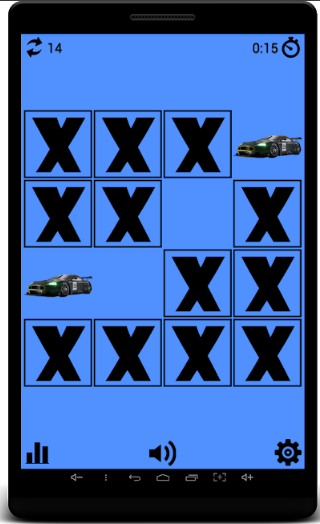


Рисунок 1.3 – Интерфейс игры «Найди пару»

Как и с предыдущим примером, правила игры «Найди пару» очень просты для понимания. Из плюсов данного приложения можно выделить следующие характеристики: удобный пользовательский интерфейс, несколько режимов сложности, колоритная коллекция изображений и приятные звуковые эффекты. Самый заметный минус приложения – обилие рекламы, которое сильно бросается в глаза. Но разработчики пообещали исправить этот недостаток в ближайших обновлениях игры.

* 1. Постановка задачи

В рамках данного курсового проекта планируется разработка игрового программного средства «Brain Train», адаптированного под запуск на современных операционных системах, в частности, версиях ОС Windows 8 и старше.

В процессе реализации будут разработаны алгоритмы генерации ходов компьютера и прорисовки игрового поля.

В игровом средстве планируется реализовать следующие функции:

* выбор игры из доступных;
* визуализация таблицы рекордов;
* повышение уровня сложности в процессе игры.

Для разработки программного средства будет использоваться язык программирования Delphi и среда разработки Embarcadero Delphi 10.2 Tokyo.

1. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА
2. 1. Структура программы

В данном приложении необходимо использовать девять структурных блоков:

– GameScreen – модуль, отображающий окно программы с главным меню;

– ColorGame – модуль, отображающий окно мини-игры «Colors»;

– FruitGame – модуль, отображающий окно мини-игры «Fruits»;

– TopPlayers – модуль, отображающий окно с таблицей рекордов;

– Question – модуль, отображающий диалоговое окно;

– EnterName – модуль, отображающий окно для ввода имени;

– GameOver – модуль, отображающий окно конца игры;

– Help – модуль, отображающий окно помощи;

– GameLib – вспомогательный модуль с объявлением типов.

* 1. Интерфейс программного средства

Так как пользователь обращает внимание в первую очередь на программный интерфейс, то необходимо сделать его понятным и простым. Ввиду того, что мы имеем дело с игровым средством, он не должен быть однообразным и скучным, чтобы удерживать внимание пользователя (обычно ребенка). При этом нужно подбирать цвета, которые не будут сильно выделяться, мешать процессу игры и при этом сочетаться с общей картинкой.

Для удобства использования программы было разработано интерактивное меню с красочными и понятными картинками. Для пользователя будет доступно три кнопки посреди экрана:

– игра «Colors»;

– игра «Fruits»;

– таблицы рекордов.

А также две кнопки в углах: «Справка» и «Выход». Наброски окна главного меню представлены на рисунке 2.1.

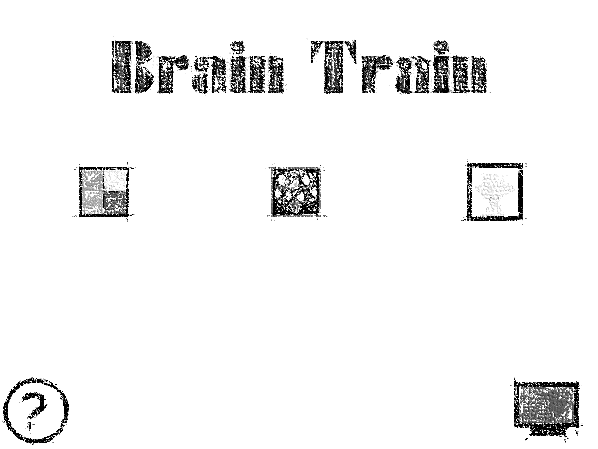


Рисунок 2.1 – Наброски главного меню приложения

При нажатии на кнопку «Выход» (экран в правом нижнем углу) появляется диалоговое окно, чтобы пользователь случайно не вышел из приложения. Наброски диалогового окна показано на рисунке 2.2.

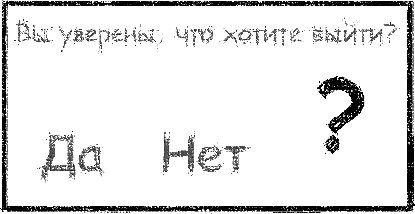


Рисунок 2.2 – Наброски диалогового окна

При нажатии пользователя на кнопку «Да», программа закроется. На «Нет» пользователь вернется в главное меню игры.

Данное диалоговое окно появляется и при попытке выхода из мини-игры в главное меню.

Нажатие на кнопку «Справка» вызовется окно с подробным описанием всех функций приложения.

При выборе кнопки «Colors» или «Fruits» откроется соответствующее окно с мини-игрой.

* 1. Графические ресурсы

Программное средство взаимодействует с определенным набором ресурсов, обеспечивающих визуальные эффекты.

В главном меню игры будет реализовано движение поезда, необходимого исключительно для добавления привлекательности приложению. Числовые константы и тип поезда показаны ниже.

Const

N = 3; //количество вагонов в поезде

Start\_Point = 90; //точка остановки поезда

Distance = 600; //расстояние между вагонами

TopOffsetOfIcon = 13;

//координата для описания положения верха иконки

LeftOffsetOfIcon = 25;

// координата для описания положения левой стороны иконки

type

TTrain = array[0..N] of TImage;

Движение и создание поезда происходит в определенных процедурах, принадлежащих классу TFMainMenu, которые показаны ниже.

procedure FormCreate(Sender: TObject);

//загрузка и прорисовка поезда при создании формы

procedure TmrTrainTimer(Sender: TObject);

//таймер движения поезда

При создании объекта данного класса происходит загрузка изображений в память. Процедура создания приведена далее.

procedure TFMainMenu.FormCreate(Sender: TObject);

var

i: Integer;

begin

for i := 0 to N do//цикл создания поезда по вагонам

begin

Train[i] := TImage.Create(FMainMenu);

//создание элемента в поезде

with Train[i] do

begin

Parent := FMainMenu;

if i = 0 then

Picture.LoadFromFile('Img\Train\Train.bmp')

//загрузка поезда

else

if i = N then

Picture.LoadFromFile('Img\Train\WagonEnd.bmp')

//загрузка средних вагонов

else

Picture.LoadFromFile('Img\Train\Wagon.bmp');

//загрузка последнего вагона

Stretch := True;

Top := 284;

//положение верха поезда

Left := Distance + i \* 100;

//прорисовка вагонов друг за другом

Transparent := True;

//установка прозрачного фона изобращения

end;

end;

TmrTrain.Enabled := True;

//запуск таймера движения поезда

end;

Движение поезда будет осуществляться путем изменения его координат по тику таймера TmrTrain с интервалом 10 миллисекунд. По завершении движения на экране будет появляться часть иконок меню. Работа таймера описана ниже.

procedure TFMainMenu.TmrTrainTimer(Sender: TObject);

var

i: Integer;

begin

for i := 0 to N do //перемещение каждого вагона влево

Train[i].Left := Train[i].Left - 5;

if Train[0].Left = Start\_Point then

//приход поезда в конечную точку

begin

ColorGameIcon.Top := Train[1].Top - TopOffsetOfIcon;

ColorGameIcon.Left := Train[1].Left + LeftOffsetOfIcon;

ColorGameIcon.Visible := True;

ColorGameIcon.BringToFront;

//размещение и появление иконки мини-игры «Colors»

FruitGameIcon.Top := Train[2].Top - TopOffsetOfIcon;

FruitGameIcon.Left := Train[2].Left + LeftOffsetOfIcon;

FruitGameIcon.Visible := True;

FruitGameIcon.BringToFront;

//размещение и появление иконки мини-игры «Fruits»

TopIcon.Top := Train[3].Top - TopOffsetOfIcon;

TopIcon.Left := Train[3].Left + LeftOffsetOfIcon;

TopIcon.Visible := True;

TopIcon.BringToFront;

//размещение и появление иконки «Рекорды»

TmrTrain.Enabled := False;

//завершение работы таймера

end;

end;

Если пользователь во время движения поезда нажмет на кнопку «Справка» или «Выход», то движение будет приостановлено до возвращения в главное меню.

* 1. Игровая логика

В приложении реализовано несколько мини-игр, у каждой из которых имеется своя игровая логика. В начале каждой игры запускается обратный отсчет, чтобы пользователь смог сориентироваться.

2.4.1 Игра «Colors»

Работу мини-игры обеспечивает модуль ColorGame.

Игровой процесс мини-игры состоит из двух фаз: хода компьютера и хода игрока. Компьютер нажимает на кнопки определенного цвета, после чего игрок должен повторить нажатия в том же порядке, в каком они были произведены. Игра заканчивается, как только игрок ошибается.

Для удобства каждому цвету было присвоено свое число:

– ноль (синий);

– один (оранжевый);

– два (зеленый);

– три (красный).

В качестве структуры для хранения информации об ходах компьютера был выбран динамический массив. В процессе игры компьютер генерирует случайные нажатия, чтобы каждая новая игра отличалась от предыдущей. Процедура, генерирующая следующий ход компьютера, представлена ниже.

procedure TFColorGame.ComputerTurn(Sender: TObject);

//генерация хода компьютера

begin

Turn := Computer; //обозначаем переход кода

BlueBtn.Enabled := False;

GreenBtn.Enabled := False;

OrangeBtn.Enabled := False;

RedBtn.Enabled := False; //блокируем кнопки

SetLength(ClrSeq, GameCount);

//увеличиваем количество нажатых компьютером клавиш

Randomize;

ClrSeq[GameCount - 1] := Random(4);

//генерируем цвет

TmrComputer.Enabled := True;

//ход компьютера

end;

В начале каждого нового хода компьютера случайным образом выполняется выбор цвета. Случайность достигается использованием комбинации процедуры Randomize и функции Random. Массив ClrSeq заполняется номерами цветов, на которые потом нажимает компьютер по таймеру, работа которого представлена далее.

procedure TFColorGame.TmrComputerTimer(Sender: TObject);

//ход компьютера

begin

if Count < GameCount then

begin

case ClrSeq[Count] of

//определение цвета и нажатие на кнопку

0:

BlueBtnClick(Sender);

1:

GreenBtnClick(Sender);

2:

OrangeBtnClick(Sender);

3:

RedBtnClick(Sender);

end;

Inc(Count);//увеличение счетчика для перехода на след шаг

end

else

begin

//конец хода компьютера

Count := 0; //обнуление счетчика

TmrComputer.Enabled := False;//завершение работы таймера

PlayerTurn(Sender); //передача хода игроку

end;

end;

Когда компьютер заканчивает свой цикл, ход передается игроку. Игрок нажимает по нужным цветам. После клика игрока по кнопке происходит проверка нажатия. В случае правильного клика увеличивается счетчик хода. Когда счетчик хода будет равен определенному значению, ход передастся обратно компьютеру. Если же нажатие было произведено не на ту кнопку, количество полученных игроком очков сравнивается с результатами в таблице рекордов: при высоких результатах игрок сможет сохранить свои очки вместе с именем в таблицу. В противном случае игрок получит уведомление об окончании игры. После чего пользователя перенаправят на главное меню приложения.

2.4.2 Игра «Fruits»

Работу мини-игры обеспечивает модуль FruitGame.

В данной игре игрок должен запомнить положение карточек с фруктами в таблице за небольшой промежуток времени, после чего они переворачиваются. Далее пользователь должен по памяти отыскать пары одинаковых фруктов. Конец игры наступает, когда игрок допускает определенное количество ошибок.

В мини-игре для простоты каждой карточке с фруктом в соответствие поставлено определенное число. Для удобства были введены определенные константные значения, показанные ниже.

const

FruitN = 17;//max кол-во фруктов начиная с 0

CardsStartX = 165;//начальное положение таблицы

CardsStartY = 15;//начальное положение таблицы

EmptyCell = 20;//код пустой карточки

CardSize = 70;//размер карточки

MarginForError = 30;//допустимое кол-во ошибок

EasyModeStart = 0;

EasyModeEnd = 1;//диапазон легкого режима

MediumModeStart = EasyModeEnd + 1;

MediumModeEnd = 6;//диапазон среднего режима

При запуске мини-игры (а также при наступлении следующего хода) происходит генерация таблицы, которая осуществляется с помощью определенных процедур, имеющих вид:

procedure TFFruitGame.GenerateTable;

var

i, j: Integer;

FruitNum: Integer;

begin

Randomize;

FruitNum := (TableSize \* TableSize) div 2;

//количество разных фруктов

SetLength(Pairs, FruitNum);

for i := 0 to High(Pairs) do

Pairs[i] := 0;

//кол-во сгенерированных элементов в паре

SetLength(FruitTable, TableSize, TableSize);

for i := 0 to High(FruitTable) do

for j := 0 to High(FruitTable) do

FruitTable[i, j] := GenNum(FruitNum);

//генерация таблицы

end;

function TFFruitGame.GenNum(FruitNum: Integer): Integer;

//генерация номера фрукта

var

GenFruit: Integer;

Loop: Boolean;

begin

Loop := True;

while Loop do

begin

GenFruit := Random(FruitNum);

//генерирование номера фрукта

if Pairs[GenFruit] < 2 then

//проверка кол-ва сгенерированных фруктов с этим номером

begin

Inc(Pairs[GenFruit]);

GenNum := GenFruit;

Loop := False;

//окончание цикла

end

end;

end;

После истечения времени на запоминание карточки с фруктами переворачиваются рубашками вверх.

В процессе игры при каждом нажатии на карточку происходит проверка содержимого и прорисовка согласно результатам.

Если игрок открыл две карточки с одинаковыми фруктами, то они убираются с игрового поля. Иначе карточки опять переворачиваются рубашками вверх, а счетчик ошибок увеличивается и сравнивается с максимальным числом ошибок. Процедуры закрытия и удаления карточек приведены ниже.

procedure TFFruitGame.DeletePair(i, j: Integer);

//удаление пары

begin

FruitTable[i, j] := EmptyCell;

FruitTable[FirstOpenCard.X, FirstOpenCard.Y] := EmptyCell;

//присваиваем элементам пары значение пустой карты

Cards.Canvas.Draw(i \* CardSize, j \* CardSize, DeleteCard);

Cards.Canvas.Draw(FirstOpenCard.X \* CardSize, FirstOpenCard.Y

\* CardSize, DeleteCard);

//закрашиваем карточки

end;

procedure TFFruitGame.ClosePair(i, j: Integer);

//закрытие пары

begin

Cards.Canvas.Draw(i \* CardSize, j \* CardSize, CloseCard);

Cards.Canvas.Draw(FirstOpenCard.X \* CardSize, FirstOpenCard.Y

\* CardSize, CloseCard);

//переворачиваем пару

Inc(Errors);

//увеличиваем счетчик ошибок

LErrCnt.Caption := IntToStr(Errors);

if Errors = MarginForError + 1 then

EndOfGame;

// смотрим на значение счетчика ошибок

//если макс начение, то заканчиваем игру

end;

Когда число ошибок достигает максимального доступного значения, программа сверяет количество очков игрока с очками из таблицы рекордов. Далее либо игроку будет предложено ввести свое имя (если игрок побил один из рекордов), либо выведется сообщение об окончании игры и пользователь будет перенаправлен в главное меню.

Во всех мини-играх, если вдруг пользователь побил один из рекордов, игрок может отказаться и не вводить своё имя. При этом его рекорд сохранен не будет. В случае ввода некорректного имени приложение либо сообщит пользователю об ошибке, либо исправит ее автоматически. Имя одного и того же пользователя вполне может встречаться в таблице рекордов несколько раз.

Если пользователь решил покинуть мини-игру, то он может в любое время нажать на кнопку «Назад». В этом случае середине экрана появится диалоговое окно, чтобы пользователь подтвердил свое решение. Если пользователь все-таки хочет продолжить игру, он может нажать на кнопку «Нет» и продолжить игру с того места, где остановился. Если же пользователь уверен в том, что хочет выйти (нажатие на кнопку «Да»), приложение сравнит очки пользователя с очками из таблицы рекордов. Если игрок побил один из рекордов, то ему будет предложено ввести имя и сохранить результат.

* 1. Работа с таблицей рекордов

В программе используется несколько таблиц рекордов (для каждой мини-игры), между которыми можно легко и удобно переключаться в любой момент времени. Работа с таблицами рекордов реализована в модуле TopPlayers.

В обеих таблицах будут отображены номер позиции игрока, его имя и количество очков. Всего в одной таблице будет записано десять участников.

Работа с данными осуществляется с помощью массивов записей, которые показаны далее.

TPlayer = record

Name: string[15];

Points: Integer;

end;

type

TTopArray = array[1..10] of TPlayer;

TopColor: TTopArray;//рейтинг игры «Colors»

TopFruit: TTopArray;//рейтинг игры «Fruits»

Таблицы заполняются данными из заранее сохраненных файлов. Если файл поврежден или содержит некорректные данные, то программа сообщит об этом пользователю. В случае, когда файл пуст, приложение заполнит его автоматические начальными значениями. Чтение файла отображено ниже.

procedure TFTop.ReadTop(Location: string; var Top: TTopArray);

// Location - путь к файлу

// Top - рейтинг игроков

var

InputFile: File of TPlayer;

isCorrect: Boolean;

i: Integer;

EmptyPlayer: TPlayer;

Begin

if FileExists(Location) then

repeat

AssignFile(InputFile, Location);

Reset(InputFile);

isCorrect := True;

if FileSize(InputFile) = 0 then

begin

EmptyPlayer.Name := '-----';

//пустое имя игрока

EmptyPlayer.Points := 0;

for i := 1 to 10 do

Write(InputFile, EmptyPlayer);

//заполнение файла пустыми значениями

end

else

begin

try

for i := 1 to 10 do

Read(InputFile, Top[i]);

//чтение данных из файла

except

ShowMessage('Некорректные данные в файле.');

isCorrect := False;

//ошибка при чтении

end;

CloseFile(InputFile);

end;

if not isCorrect then

begin

EmptyPlayer.Name := '-----';

EmptyPlayer.Points := 0;

for i := 1 to 10 do

Write(InputFile, EmptyPlayer);

end;

else

ShowMessage('Файл с таблицей рекордов не найден.');

end;

Сохранение результатов в файл осуществляется с помощью процедуры, приведенной ниже.

procedure TFTop.SaveResults(Top: TTopArray; Game: TGame);

//Top - рейтинг

//Game - мини-игра

var

OutputFile: File of TPlayer;

i: Integer;

begin

if Game = GColor then

AssignFile(OutputFile, 'ColorGameFile.dat')

else

AssignFile(OutputFile, 'FruitGameFile.dat');

ReWrite(OutputFile);

for i := 1 to 10 do

Write(OutputFile, Top[i]);

//сохранение в файл

CloseFile(OutputFile);

end;

Переключение между таблицами рекордов мини-игр происходит по нажатию на кнопку.

* 1. Диалоговое окно и окно ввода имени

В игре добавлены окна, помогающие пользователю сделать выбор и случайно не выйти из игры без сохранения своих результатов.

2.6.1 Диалоговое окно

При попытке пользователя вернуться назад в главное меню либо выйти из игры будет отображено диалоговое окно, в котором пользователь должен подтвердить свой выбор. Диалоговое окно было введено, чтобы пользователь не вышел во время игры из-за случайного нажатия. Работа с диалоговым окном осуществляется в модуле Question. Диалоговое окно отображено на рисунке 2.3.

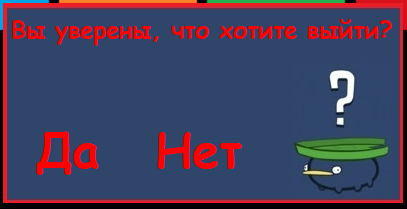


Рисунок 2.3 – Диалоговое окно приложения

Во время появления диалогового окна все процессы в игре приостанавливаются, чтобы игрок ничего не пропустил и смог вернуться назад в игру с того места, где остановился.

2.6.2 Окно ввода имени

Для того, чтобы пользователь смог отделять свой рекорд от остальных, было добавлено окно ввода имени. Оно всплывает только в тех ситуациях, когда пользователь побил один из имеющихся сохраненных рекордов. В противном случае вместо окна ввода имени возникнет окно конца игры. Работа с окном ввода имени реализована в модуле EnterName.

1. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В ходе тестирования приложения были выявлены некоторые недочеты и ошибки в работе программного средства.

Было обнаружено, что на главном меню доступна для нажатия только определенная часть каждой иконки. Проблема была решена путем перемещения иконок на передний план.

Другим недочетом оказалось то, что в мини-игре «Colors», при нажатии пользователем на кнопку «Назад», компьютер продолжал нажимать на кнопки с цветами, что было существенным недостатком, если пользователь решит вернуться в игру, так как окно диалоговое окно перекрывает часть кнопок. Недочет был исправлен с помощью остановки таймера хода компьютера и при необходимости последующим возобновлением его работы. Итоговый вариант приведен далее.

procedure TFColorGame.BtnExitClick(Sender: TObject);

//нажатие на кнопку выхода из игры

begin

if TmrComputer.Enabled then

//остановка хода компьютера

begin

TmrComputer.Enabled := False;

StopPoint := SPGameProcess;

end

else

//остановка обратного отсчета

begin

TmrCountdown.Enabled := False;

StopPoint := SPCountdown;

end;

FColorGame.Enabled := False;

FQuestion.NumbOfParentForm := 1;

//номер игры

FQuestion.Show;

//вызов диалогового окна

end;

Следующий недочет был связан с тем, что при нажатии на кнопки «Выход» - «Нет» в главном меню, если на заднем фоне были открыты другие окна, то приложение перемещалось на задний план. Недочет был устранен путем закрытия диалогового окна в другой части кода.

Остальные проблемы были связаны с недочётами на стадии проектирования и были устранены в процессе разработки приложения.

1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
   1. Правила игры

Правила мини-игр очень просты и понятны для людей практически любого возраста.

4.1.1 Игра «Fruits»

Правила игры «Fruits» полностью совпадают с ее классической версией, более известной как «Найди пару». Игрок за небольшой промежуток времени должен запомнить расположение карточек с фруктами на поле, после чего они закрываются. Игрок переворачивает по две карточки. Если изображения совпадают, карточки убираются с поля, а игроку начисляется один балл. В противном случае карточки снова закрываются. Игра продолжается до тех пор, пока игрок не допустит определенное количество ошибок.

**4.1.2** Игра «Colors»

В игре «Colors» игрок должен следить за нажатием кнопок четырех различных цветов и после этого стараться повторить последовательно все нажатия в том же порядке. С каждым ходом количество нажатий увеличивается на одно. Как только игрок ошибается и нажимает не на ту кнопку, игра заканчивается.

* 1. Интерфейс программы

Главное меню программы, изображенное на рисунке 4.1, содержит следующие элементы:

– картинку с названием игры;

– анимированный поезд;

– иконку мини-игры «Colors» (иконка с четырьмя квадратами);

– иконку мини-игры «Fruits» (иконка с изображением дерева);

– иконку «Рекорды» (иконка с изображением кубка);

– кнопку «Справка» (кнопка в виде вопросительного знака);

– кнопку «Выход» (кнопка в виде монитора).

Для начала одной из мини-игр необходимо кликнуть на соответствующую иконку. Игры запускаются автоматически после нажатия на кнопку.

Для работы приложения необходим файл с таблицей рекордов. В случае, если файл пуст, он будет заполнен автоматически. При этом значения в таблице рекордов будут заполнены пустыми игроками с нулевым количеством очков так, как показано на рисунке 4.2.



Рисунок 4.1 – Внешний вид главного меню программы

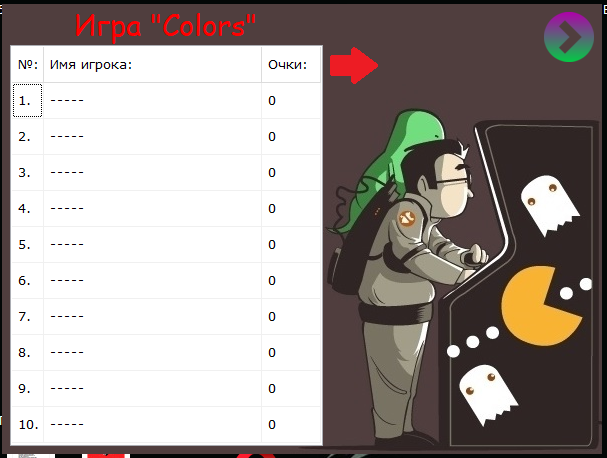


Рисунок 4.2 – Пример пустой таблицы рекордов

Также в приложении присутствуют мини-игры со своим оригинальным интерфейсом. Мини-игра «Colors» показана на рисунке 4.3, игра «Fruits» – на рисунке 4.4.

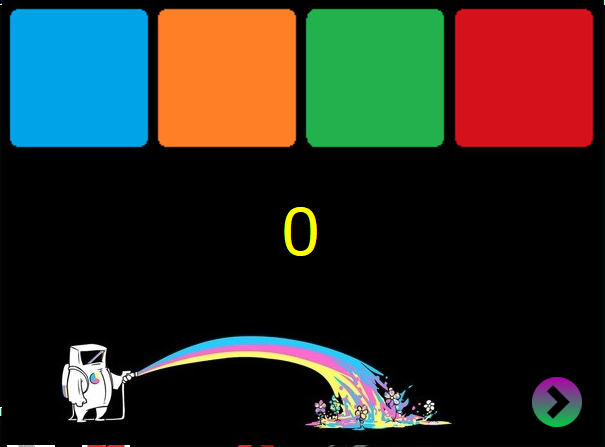


Рисунок 4.3 – Интерфейс мини-игры «Colors»



Рисунок 4.4 – Интерфейс мини-игры «Fruits»

В программе есть подробная цветная инструкция со всеми правилами и руководствами, чтобы пользователь смог разобраться во всех функциях приложения.

ЗАКЛЮЧЕНИЕ

В наши дни компьютерные игры давно перестали нести только развлекательный характер: их используют для обучения, соревнований, исследований. Многие передовые разработки выходят за рамки игровой индустрии и внедряются в медицину, военное дело, строительство и другие сферы жизни.

В рамках данного курсового проекта было разработано игровое программное средство «Brain Train», которое поможет пользователю развивать свои память и внимание в свободное время. Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

* выбор мини-игры из доступных;
* визуализация таблицы рекордов;
* повышение уровня сложности в процессе игры.

Для успешного выполнения всех поставленных целей потребовалось изучить объектно-ориентированные возможности языка Delphi, изучить основные принципы данной парадигмы, а также освоить создание собственных компонентов на основе уже существующих.

В дальнейшем можно по-разному улучшать приложение, например, добавляя новые мини-игры. Также можно добавить звуковое сопровождение и улучшить уже доступные визуальные эффекты. Еще возможно сделать окно настроек, чтобы пользователь смог менять характеристики приложения, такие как размер окна, сложность игры, отключение анимации для слабых компьютеров.

Данное приложение очень хорошо развивает умственные показатели человека: память, внимание и логическое мышление. Продолжительность игры обычно небольшая, что позволяет пользователю не тратить свое время на длительное прохождение уровней.

Однако, как бы игра не помогала вашему мозгу, следует помнить, что зрение также является важным чувством. Поэтому следует делать определенные перерывы и поддерживать не только умственную, но и физическую активность, так как здоровье – бесценный ресурс, который нужно беречь каждому.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Глухова, Л. А. Основы алгоритмизации и программирования: Лаб. практикум для студ. спец. I-40 01 01 «Программное обеспечение информационных технологий» дневной формы обуч. В 4 ч. / Л. А. Глухова, Е. Е. Фадеева, Е. П. Фадеева. – Минск: БГУИР, 2007. – Ч. 3. – 51 с.
2. Орлов, С. А. Технологии разработки программного обеспечения: учеб. Пособие. – СПб, изд. «Питер», 2003.
3. Программирование на языке Delphi / Д. А. Сурков [и др.]. – учеб. пособие. – Режим доступа: http://www.rsdn.ru/?summary/3165.xml, – 2005.
4. Уилсон, С. Принципы проектирования и разработки программного обеспечения, yчебн. курс. – СПб, изд. «Русская редакция», 2003.
5. Фараонов, В. В. Delphi 6. Учебный курс: / В. В. Фараонов. – СПб: изд. С. В. Малгачёва, 2001. – 231 с.
6. Шупрута, В. В. Delphi 2005. Учимся программировать: / В. В. Шупрута. – Москва: изд. «НТ Пресс», 2001. – 140 с.
7. RAD Studio Product Documentation – Embarcadero Technologies [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://docs.embarcadero.com/products/rad_studio>.

ПРИЛОЖЕНИЕ А

Исходный код программы

Форма с главным меню:

unit GameScreen;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.StdCtrls, Vcl.Imaging.jpeg,Vcl.ExtCtrls, GameLib, Vcl.Imaging.pngimage,

ColorGame, TopPlayers, Vcl.ComCtrls, FruitGame, Question, Help;

const

N = 3;

Start\_Point = 90;

Distance = 600;

TopOffsetOfIcon = 13;

LeftOffsetOfIcon = 25;

type

TTrain = array[0..N] of TImage;

TFMainMenu = class(TForm)

ImgBackground: TImage;

ImgMainText: TImage;

BtnClose: TImage;

ColorGameIcon: TImage;

FruitGameIcon: TImage;

TopIcon: TImage;

TmrTrain: TTimer;

BtnHelp: TImage;

procedure FormCreate(Sender: TObject);

procedure TmrTrainTimer(Sender: TObject);

procedure BtnCloseClick(Sender: TObject);

procedure ColorGameIconClick(Sender: TObject);

procedure FruitGameIconClick(Sender: TObject);

procedure TopIconClick(Sender: TObject);

procedure BtnHelpClick(Sender: TObject);

private

{ Private declarations }

public

Train: TTrain;

end;

var

FMainMenu: TFMainMenu;

implementation

{$R \*.dfm}

procedure TFMainMenu.BtnCloseClick(Sender: TObject);

begin

TmrTrain.Enabled := False;

FMainMenu.Enabled := False;

FQuestion.NumbOfParentForm := 0;

FQuestion.Show;

end;

procedure TFMainMenu.BtnHelpClick(Sender: TObject);

begin

TmrTrain.Enabled := False;

FMainMenu.Enabled := False;

FHelp.Show;

end;

procedure TFMainMenu.ColorGameIconClick(Sender: TObject);

begin

FColorGame.Show;

end;

procedure TFMainMenu.FormCreate(Sender: TObject);

var

i: Integer;

begin

for i := 0 to N do

begin

Train[i] := TImage.Create(FMainMenu);

with Train[i] do

begin

Parent := FMainMenu;

if i = 0 then

Picture.LoadFromFile('Img\Train\Train.bmp')

else

if i = N then

Picture.LoadFromFile('Img\Train\WagonEnd.bmp')

else

Picture.LoadFromFile('Img\Train\Wagon.bmp');

Stretch := True;

Top := 284;

Left := Distance + i \* 100;

Transparent := True;

end;

end;

TmrTrain.Enabled := True;

end;

procedure TFMainMenu.FruitGameIconClick(Sender: TObject);

begin

FFruitGame.Show;

end;

procedure TFMainMenu.TmrTrainTimer(Sender: TObject);

var

i: Integer;

begin

for i := 0 to N do

Train[i].Left := Train[i].Left - 5;

if Train[0].Left = Start\_Point then

begin

ColorGameIcon.Top := Train[1].Top - TopOffsetOfIcon;

ColorGameIcon.Left := Train[1].Left + LeftOffsetOfIcon;

FruitGameIcon.Top := Train[2].Top - TopOffsetOfIcon;

FruitGameIcon.Left := Train[2].Left + LeftOffsetOfIcon;

TopIcon.Top := Train[3].Top - TopOffsetOfIcon;

TopIcon.Left := Train[3].Left + LeftOffsetOfIcon;

ColorGameIcon.Visible := True;

FruitGameIcon.Visible := True;

TopIcon.Visible := True;

ColorGameIcon.BringToFront;

FruitGameIcon.BringToFront;

TopIcon.BringToFront;

TmrTrain.Enabled := False;

end;

end;

procedure TFMainMenu.TopIconClick(Sender: TObject);

begin

FTop.Show;

end;

end.

Модуль мини-игры «Colors»:

unit ColorGame;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.Imaging.pngimage,Vcl.Imaging.jpeg, Vcl.ExtCtrls, Vcl.StdCtrls, GameLib,

Question, TopPlayers,EnterName, GameOver;

type

TFColorGame = class(TForm)

ImgBackground: TImage;

BlueBtn: TImage;

OrangeBtn: TImage;

GreenBtn: TImage;

RedBtn: TImage;

BtnExit: TImage;

TmrBtnClick: TTimer;

TmrComputer: TTimer;

TmrCountdown: TTimer;

LPointCnt: TLabel;

LCountdown: TLabel;

procedure TmrBtnClickTimer(Sender: TObject);

procedure BlueBtnClick(Sender: TObject);

procedure OrangeBtnClick(Sender: TObject);

procedure GreenBtnClick(Sender: TObject);

procedure RedBtnClick(Sender: TObject);

procedure BtnClick(Clr: Integer; Btn: TImage; Sender: TObject);

procedure FormShow(Sender: TObject);

procedure TmrCountdownTimer(Sender: TObject);

procedure TmrComputerTimer(Sender: TObject);

procedure BtnExitClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

ClickTime: Integer;

Color: Integer;

Countdown: Integer;

Count: Integer;

Turn: TTurn;

ClrSeq: array of Integer;

procedure GameOver;

procedure StrokeToComputer(Sender: TObject);

procedure GameProcess(Sender: TObject);

procedure ComputerTurn(Sender: TObject);

procedure PlayerTurn(Sender: TObject);

public

GameCount: Integer;

StopPoint: TStopPoint;

end;

{ 0 - blue

1 - green

2 - orange

3 - red }

var

FColorGame: TFColorGame;

implementation

{$R \*.dfm}

procedure TFColorGame.GameOver;

begin

if GameCount - 1 > FTop.TopColor[10].Points then

begin

FEnterName.Points := GameCount - 1;

FEnterName.Game := GColor;

FColorGame.Enabled := False;

FEnterName.Show;

end

else

FGameOver.Show;

end;

procedure TFColorGame.PlayerTurn(Sender: TObject); //ход игрока

begin

Turn := Player;

BlueBtn.Enabled := True;

GreenBtn.Enabled := True;

OrangeBtn.Enabled := True;

RedBtn.Enabled := True;

end;

procedure TFColorGame.TmrComputerTimer(Sender: TObject);

//продолжительность хода

begin

if Count < GameCount then

begin

case ClrSeq[Count] of

0:

BlueBtnClick(Sender);

1:

GreenBtnClick(Sender);

2:

OrangeBtnClick(Sender);

3:

RedBtnClick(Sender);

end;

Inc(Count);

end

else

begin

Count := 0;

TmrComputer.Enabled := False;

PlayerTurn(Sender);

end;

end;

procedure TFColorGame.FormClose(Sender: TObject; var Action: TCloseAction);

begin

LPointCnt.Visible := False;

TmrComputer.Enabled := False;

end;

procedure TFColorGame.StrokeToComputer(Sender: TObject);

begin

Inc(GameCount);

Count := 0;

LPointCnt.Caption := IntToStr(StrToInt(LPointCnt.Caption) + 1);

ComputerTurn(Sender);

end;

procedure TFColorGame.BtnClick(Clr: Integer; Btn: TImage; Sender: TObject);

begin

Btn.Picture.LoadFromFile('Img\ColorGame\ClrButtonClick'

+IntToStr(Clr)+'.bmp');

ClickTime := 3;

TmrBtnClick.Enabled := True;

if Turn = Player then

if Count <= GameCount - 1 then

if Clr = ClrSeq[Count] then

begin

Inc(Count);

if Count = GameCount then

StrokeToComputer(Sender);

end

else

GameOver

else

StrokeToComputer(Sender);

end;

procedure TFColorGame.BtnExitClick(Sender: TObject);

begin

if TmrComputer.Enabled then

begin

TmrComputer.Enabled := False;

StopPoint := SPGameProcess;

end

else

begin

TmrCountdown.Enabled := False;

StopPoint := SPCountdown;

end;

FColorGame.Enabled := False;

FQuestion.NumbOfParentForm := 1;

FQuestion.Show;

end;

procedure TFColorGame.BlueBtnClick(Sender: TObject);

begin

Color := 0;

BtnClick(Color, BlueBtn, Sender);

end;

procedure TFColorGame.GreenBtnClick(Sender: TObject);

begin

Color := 1;

BtnClick(Color, GreenBtn, Sender);

end;

procedure TFColorGame.OrangeBtnClick(Sender: TObject);

begin

Color := 2;

BtnClick(Color, OrangeBtn, Sender);

end;

procedure TFColorGame.RedBtnClick(Sender: TObject);

begin

Color := 3;

BtnClick(Color, RedBtn, Sender);

end;

procedure TFColorGame.TmrBtnClickTimer(Sender: TObject);

begin

Dec(ClickTime);

if ClickTime = 0 then

begin

TmrBtnClick.Enabled := False;

case Color of

0:

BlueBtn.Picture.LoadFromFile('Img\ColorGame\ClrButton'

+IntToStr(Color)+'.bmp');

1:

GreenBtn.Picture.LoadFromFile('Img\ColorGame\ClrButton'

+IntToStr(Color)+'.bmp');

2:

OrangeBtn.Picture.LoadFromFile('Img\ColorGame\ClrButton'

+IntToStr(Color)+'.bmp');

3:

RedBtn.Picture.LoadFromFile('Img\ColorGame\ClrButton'

+IntToStr(Color)+'.bmp');

end;

end;

end;

procedure TFColorGame.ComputerTurn(Sender: TObject); //ход компьютера

begin

Turn := Computer;

BlueBtn.Enabled := False;

GreenBtn.Enabled := False;

OrangeBtn.Enabled := False;

RedBtn.Enabled := False;

SetLength(ClrSeq, GameCount);

Randomize;

ClrSeq[GameCount - 1] := Random(4);

TmrComputer.Enabled := True;

end;

procedure TFColorGame.GameProcess(Sender: TObject); //начало игры

begin

ComputerTurn(Sender);

end;

procedure TFColorGame.TmrCountdownTimer(Sender: TObject);

begin

Dec(Countdown);

LCountdown.Caption := IntToStr(Countdown);

if Countdown = 0 then

begin

TmrCountdown.Enabled := False;

LCountdown.Visible := False;

LPointCnt.Visible := True;

GameProcess(Sender);

end;

end;

procedure TFColorGame.FormShow(Sender: TObject);

begin

LPointCnt.Caption := '0';

Count := 0;

GameCount := 1;

LCountdown.Visible := True;

FColorGame.Enabled := True;

BlueBtn.Enabled := False;

GreenBtn.Enabled := False;

OrangeBtn.Enabled := False;

RedBtn.Enabled := False;

Countdown := 3;

LCountdown.Caption := IntToStr(Countdown);

TmrCountdown.Enabled := True;

end;

end.

Модуль мини-игры «Fruits»:

unit FruitGame;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.Imaging.jpeg, Vcl.ExtCtrls, Vcl.Grids, Vcl.StdCtrls,

Vcl.Samples.Gauges, Vcl.ComCtrls, Vcl.Imaging.pngimage, Question, GameLib,

GameOver;

const

FruitN = 17;

CardsStartX = 165;

CardsStartY = 15;

EmptyCell = 20;

CardSize = 70;

MarginForError = 30;

EasyModeStart = 0;

EasyModeEnd = 1;

MediumModeStart = EasyModeEnd + 1;

MediumModeEnd = 6;

type

TFirstOpenCard = record

X, Y: Integer;

FruitNum: Integer;

end;

TCellCoordinate = record

X, Y: Integer;

end;

TFFruitGame = class(TForm)

ImgFruitGameBackground: TImage;

Cards: TImage;

BtnExit: TImage;

LCountdown: TLabel;

LPointCnt: TLabel;

LTimeToRemember: TLabel;

LWellDone: TLabel;

LErrors: TLabel;

LErrCnt: TLabel;

TmrCountdown: TTimer;

TmrToRemember: TTimer;

TmrWaitCheck: TTimer;

procedure FormCreate(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure TmrCountdownTimer(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure TmrToRememberTimer(Sender: TObject);

procedure CardsMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure TmrWaitCheckTimer(Sender: TObject);

procedure BtnExitClick(Sender: TObject);

private

Fruit: array[0..FruitN] of TBitmap;

CloseCard: TBitmap;

DeleteCard: TBitmap;

TableSize: Integer;

FruitTable: TTable;

Pairs: array of Integer;

Countdown: Integer;

CountdownForMemory: Integer;

CardOpen: Integer;

FirstOpenCard: TFirstOpenCard;

CellCoordinate: TCellCoordinate;

Errors: Integer;

procedure DrawTable(OffsetX, OffsetY: Integer);

procedure GenerateTable;

procedure StartLevel(LengthOfSide, OffSetX, OffsetY: Integer);

procedure CloseTable;

procedure DeletePair(i, j: Integer);

procedure ClosePair(i, j: Integer);

procedure ClearTable;

procedure EndOfGame;

procedure NextLevel;

function GenNum(FruitNum: Integer): Integer;

function EmptyTable: Boolean;

function EqualFruits(i, j: Integer): Boolean;

public

Points: Integer;

StopPoint: TStopPoint;

end;

var

FFruitGame: TFFruitGame;

implementation

{$R \*.dfm}

uses

GameScreen, TopPlayers, EnterName;

procedure TFFruitGame.EndOfGame;

begin

FFruitGame.Enabled := False;

if Points > FTop.TopFruit[10].Points then

begin

FEnterName.Points := Points;

FEnterName.Game := GFruit;

FEnterName.Show;

end

else

FGameOver.Show;

end;

procedure TFFruitGame.ClearTable;

var

i, j: Integer;

X, Y: Integer;

begin

X := 0;

Y := 0;

for i := 0 to (TableSize - 1) do

begin

for j := 0 to (TableSize - 1) do

begin

Cards.Canvas.Draw(X, Y, DeleteCard);

X := X + 70;

end;

X := 0;

Y := Y + 70;

end;

end;

function TFFruitGame.EmptyTable: Boolean;

var

i, j: Integer;

begin

EmptyTable := True;

for i := 0 to (TableSize - 1) do

for j := 0 to (TableSize - 1) do

if FruitTable[i, j] <> 20 then

EmptyTable := False;

end;

procedure TFFruitGame.ClosePair(i, j: Integer);

begin

Cards.Canvas.Draw(i \* CardSize, j \* CardSize, CloseCard);

Cards.Canvas.Draw(FirstOpenCard.X \* CardSize, FirstOpenCard.Y

\* CardSize, CloseCard);

Inc(Errors);

LErrCnt.Caption := IntToStr(Errors);

if Errors = MarginForError + 1 then

EndOfGame;

end;

procedure TFFruitGame.DeletePair(i, j: Integer);

begin

FruitTable[i, j] := EmptyCell;

FruitTable[FirstOpenCard.X, FirstOpenCard.Y] := EmptyCell;

Cards.Canvas.Draw(i \* CardSize, j \* CardSize, DeleteCard);

Cards.Canvas.Draw(FirstOpenCard.X \* CardSize, FirstOpenCard.Y

\* CardSize, DeleteCard);

end;

function TFFruitGame.EqualFruits(i, j: Integer): Boolean;

begin

if FruitTable[i, j] = FirstOpenCard.FruitNum then

EqualFruits := True

else

EqualFruits := False;

end;

procedure TFFruitGame.TmrWaitCheckTimer(Sender: TObject);

begin

if (CardOpen = 2) and ((FirstOpenCard.X <> CellCoordinate.X)

or (FirstOpenCard.Y <> CellCoordinate.Y)) then

begin

if EqualFruits(CellCoordinate.X, CellCoordinate.Y) then

DeletePair(CellCoordinate.X, CellCoordinate.Y)

else

ClosePair(CellCoordinate.X, CellCoordinate.Y);

CardOpen := 0;

end

else

if CardOpen > 1 then

Dec(CardOpen);

TmrWaitCheck.Enabled:= False;

if EmptyTable then

begin

Inc(Points);

LPointCnt.Caption := IntToStr(Points);

LWellDone.Visible := True;

NextLevel;

end;

end;

procedure TFFruitGame.CardsMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if CardOpen < 2 then

begin

CellCoordinate.X := X div CardSize;

CellCoordinate.Y := Y div CardSize;

if (CellCoordinate.X < TableSize) and (CellCoordinate.Y < TableSize) then

if FruitTable[CellCoordinate.X, CellCoordinate.Y] <> EmptyCell then

begin

Inc(CardOpen);

Cards.Canvas.Draw(CellCoordinate.X \* CardSize,

CellCoordinate.Y \* CardSize,

Fruit[FruitTable[CellCoordinate.X, CellCoordinate.Y]]);

if CardOpen = 1 then

begin

FirstOpenCard.FruitNum := FruitTable[CellCoordinate.X,

CellCoordinate.Y];

FirstOpenCard.X := CellCoordinate.X;

FirstOpenCard.Y := CellCoordinate.Y;

end;

TmrWaitCheck.Enabled := True;

end;

end;

end;

procedure TFFruitGame.CloseTable;

var

i, j: Integer;

X, Y: Integer;

begin

X := 0;

Y := 0;

for i := 0 to (TableSize - 1) do

begin

for j := 0 to (TableSize - 1) do

begin

Cards.Canvas.Draw(X, Y, CloseCard);

X := X + 70;

end;

X := 0;

Y := Y + 70;

end;

end;

procedure TFFruitGame.BtnExitClick(Sender: TObject);

begin

if TmrToRemember.Enabled then

begin

TmrToRemember.Enabled := False;

StopPoint := SPGameProcess;

end

else

begin

TmrCountdown.Enabled := False;

StopPoint := SPCountdown;

end;

FFruitGame.Enabled := False;

FQuestion.NumbOfParentForm := 2;

FQuestion.Show;

end;

procedure TFFruitGame.FormCreate(Sender: TObject);

var

i: Integer;

begin

for i := 0 to FruitN do

begin

Fruit[i] := TBitmap.Create;

Fruit[i].LoadFromFile('Img\FruitGame\' + IntToStr(i) + '.bmp');

end;

CloseCard := TBitmap.Create;

CloseCard.LoadFromFile('Img\FruitGame\CloseCard.bmp');

DeleteCard := TBitmap.Create;

DeleteCard.LoadFromFile('Img\FruitGame\DeleteCard.bmp');

end;

procedure TFFruitGame.DrawTable(OffsetX, OffsetY: Integer);

var

i, j: Integer;

X, Y: Integer;

begin

X := 0;

Y := 0;

Cards.Top := CardsStartY + OffSetY;

Cards.Left := CardsStartX + OffSetX;

for i := 0 to (TableSize - 1) do

begin

for j := 0 to (TableSize - 1) do

begin

Cards.Canvas.Draw(X, Y, Fruit[FruitTable[i, j]]);

Y := Y + 70;

end;

Y := 0;

X := X + 70;

end;

end;

function TFFruitGame.GenNum(FruitNum: Integer): Integer;

var

GenFruit: Integer;

Loop: Boolean;

begin

Loop := True;

while Loop do

begin

GenFruit := Random(FruitNum);

if Pairs[GenFruit] < 2 then

begin

Inc(Pairs[GenFruit]);

GenNum := GenFruit;

Loop := False;

end

end;

end;

procedure TFFruitGame.TmrCountdownTimer(Sender: TObject);

begin

Dec(Countdown);

LCountdown.Caption := IntToStr(Countdown);

if Countdown = 0 then

begin

TmrCountdown.Enabled := False;

LCountdown.Visible := False;

LPointCnt.Visible := True;

LErrCnt.Visible := True;

LErrors.Visible := True;

LTimeToRemember.Visible := True;

TmrToRemember.Enabled := True;

end;

end;

procedure TFFruitGame.TmrToRememberTimer(Sender: TObject);

begin

LWellDone.Visible := False;

Dec(CountdownForMemory);

LTimeToRemember.Caption := IntToStr(CountdownForMemory);

if CountdownForMemory = 0 then

begin

TmrToRemember.Enabled := False;

CountdownForMemory := 3;

CloseTable;

LTimeToRemember.Visible := False;

Cards.Enabled := True;

end;

end;

procedure TFFruitGame.GenerateTable;

var

i, j: Integer;

FruitNum: Integer;

begin

Randomize;

FruitNum := (TableSize \* TableSize) div 2;

SetLength(Pairs, FruitNum);

for i := 0 to High(Pairs) do

Pairs[i] := 0;

SetLength(FruitTable, TableSize, TableSize);

for i := 0 to High(FruitTable) do

for j := 0 to High(FruitTable) do

FruitTable[i, j] := GenNum(FruitNum);

end;

procedure TFFruitGame.StartLevel(LengthOfSide, OffSetX, OffsetY: Integer);

begin

TableSize := LengthOfSide;

GenerateTable;

DrawTable(OffSetX, OffsetY);

end;

procedure TFFruitGame.NextLevel;

begin

case Points of

EasyModeStart..EasyModeEnd:

begin

StartLevel(2, 140, 140);

CountdownForMemory := 3;

end;

MediumModeStart..MediumModeEnd:

begin

StartLevel(4, 70, 70);

CountdownForMemory := 5;

end

else

begin

StartLevel(6, 0, 0);

CountdownForMemory := 15;

end;

end;

CardOpen := 0;

LTimeToRemember.Caption := IntToStr(CountdownForMemory);

Cards.Enabled := False;

LTimeToRemember.Visible := True;

TmrToRemember.Enabled := True;

end;

procedure TFFruitGame.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Cards.Top := CardsStartY;

Cards.Left := CardsStartX;

LPointCnt.Visible := False;

LErrors.Visible := False;

LErrCnt.Visible := False;

LTimeToRemember.Visible := False;

TmrToRemember.Enabled := False;

end;

procedure TFFruitGame.FormShow(Sender: TObject);

begin

TableSize := 6;

ClearTable;

StartLevel(2, 140, 140);

Points := 0;

Errors := 0;

CountdownForMemory := 3;

CardOpen := 0;

LWellDone.Visible := False;

LPointCnt.Caption := '0';

LErrCnt.Caption := '0';

LTimeToRemember.Caption := '3';

FFruitGame.Enabled := True;

LCountdown.Visible := True;

Cards.Enabled := False;

Countdown := 3;

LCountdown.Caption := IntToStr(Countdown);

TmrCountdown.Enabled := True;

end;

end.

Модуль работы с таблицами рекордов:

unit TopPlayers;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.Imaging.jpeg, Vcl.ExtCtrls, Vcl.StdCtrls, Vcl.Grids, Vcl.Imaging.pngimage, GameLib;

type

TTopArray = array[1..10] of TPlayer;

TFTop = class(TForm)

SGTop: TStringGrid;

ImgBackground: TImage;

BtnBack: TImage;

BtnNext: TImage;

LTitle: TLabel;

procedure BtnBackClick(Sender: TObject);

procedure BtnNextClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormShow(Sender: TObject);

private

GameShow: TGame;

public

TopColor: TTopArray;

TopFruit: TTopArray;

procedure FindPositionInTop(Points: Integer; Game: TGame);

procedure ReadTop(Location: string; var Top: TTopArray);

procedure FillTable(Top: TTopArray);

procedure Find(Top: TTopArray; Points: Integer; Game: TGame);

procedure SaveResults(Top: TTopArray; Game: TGame);

end;

var

FTop: TFTop;

implementation

{$R \*.dfm}

uses

EnterName;

procedure TFTop.SaveResults(Top: TTopArray; Game: TGame);

var

OutputFile: File of TPlayer;

i: Integer;

begin

if Game = GColor then

AssignFile(OutputFile, 'ColorGameFile.dat')

else

AssignFile(OutputFile, 'FruitGameFile.dat');

ReWrite(OutputFile);

for i := 1 to 10 do

Write(OutputFile, Top[i]);

CloseFile(OutputFile);

end;

procedure TFTop.Find(Top: TTopArray; Points: Integer; Game: TGame);

var

i, j: Integer;

More: Boolean;

begin

i := 9;

More := True;

while More do

begin

if (Points > Top[i].Points) and (i <> 0) then

Dec(i)

else

begin

for j := 10 downto (i + 2) do

Top[j] := Top[j - 1];

Top[i + 1].Points := Points;

Top[i + 1].Name := FEnterName.EdtName.Text;

SaveResults(Top, Game);

More := False;

end;

end;

end;

procedure TFTop.FindPositionInTop(Points: Integer; Game: TGame);

begin

if Game = GColor then

Find(TopColor, Points, Game)

else

Find(TopFruit, Points, Game);

end;

procedure TFTop.BtnBackClick(Sender: TObject);

begin

FTop.Close;

end;

procedure TFTop.BtnNextClick(Sender: TObject);

begin

if GameShow = GColor then

begin

GameShow := GFruit;

FillTable(TopFruit);

LTitle.Caption := 'Игра "Fruits"';

end

else

begin

GameShow := GColor;

FillTable(TopColor);

LTitle.Caption := 'Игра "Colors"';

end;

end;

procedure TFTop.ReadTop(Location: string; var Top: TTopArray);

var

InputFile: File of TPlayer;

isCorrect: Boolean;

i: Integer;

EmptyPlayer: TPlayer;

begin

if FileExists(Location) then

repeat

AssignFile(InputFile, Location);

Reset(InputFile);

isCorrect := True;

if FileSize(InputFile) = 0 then

begin

EmptyPlayer.Name := '-----';

EmptyPlayer.Points := 0;

for i := 1 to 10 do

Write(InputFile, EmptyPlayer);

end

else

begin

try

for i := 1 to 10 do

Read(InputFile, Top[i]);

except

ShowMessage('Некорректные данные в файле.');

isCorrect := False;

end;

CloseFile(InputFile);

end;

if not isCorrect then

begin

EmptyPlayer.Name := '-----';

EmptyPlayer.Points := 0;

for i := 1 to 10 do

Write(InputFile, EmptyPlayer);

end;

until isCorrect

else

ShowMessage('Файл с таблицей рекордов не найден.');

end;

procedure TFTop.FillTable(Top: TTopArray);

var

i: Integer;

begin

for i := 1 to 10 do

begin

SGTop.Cells[1, i] := Top[i].Name;

SGTop.Cells[2, i] := IntToStr(Top[i].Points);

end;

end;

procedure TFTop.FormCreate(Sender: TObject);

var

i: Integer;

begin

SGTop.Cells[0, 0] := '№:';

SGTop.Cells[1, 0] := 'Имя игрока:';

SGTop.Cells[2, 0] := 'Очки:';

for i := 1 to 10 do

SGTop.Cells[0, i] := IntToStr(i) + '.';

ReadTop('FruitGameFile.dat', TopFruit);

ReadTop('ColorGameFile.dat', TopColor);

end;

procedure TFTop.FormShow(Sender: TObject);

begin

LTitle.Caption := 'Игра "Colors"';

ReadTop('FruitGameFile.dat', TopFruit);

ReadTop('ColorGameFile.dat', TopColor);

GameShow := GColor;

FillTable(TopColor);

end;

end.

Модуль диалогового окна:

unit Question;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.StdCtrls, Vcl.Imaging.jpeg, Vcl.ExtCtrls, GameLib;

type

TFQuestion = class(TForm)

ImgBackground: TImage;

LQuestion: TLabel;

LYes: TLabel;

LNo: TLabel;

procedure LNoClick(Sender: TObject);

procedure LYesClick(Sender: TObject);

private

{ Private declarations }

public

NumbOfParentForm: Integer;

end;

var

FQuestion: TFQuestion;

implementation

uses

GameScreen, ColorGame, FruitGame, TopPlayers, EnterName;

{$R \*.dfm}

procedure TFQuestion.LNoClick(Sender: TObject);

begin

FQuestion.Close;

case NumbOfParentForm of

0:

begin

FMainMenu.Enabled := True;

if FMainMenu.Train[0].Left <> Start\_Point then

FMainMenu.TmrTrain.Enabled := True;

end;

1:

begin

FColorGame.Enabled := True;

if FColorGame.StopPoint = SPGameProcess then

FColorGame.TmrComputer.Enabled := True

else

FColorGame.TmrCountdown.Enabled := True;

end;

2:

begin

FFruitGame.Enabled := True;

if FFruitGame.StopPoint = SPGameProcess then

FFruitGame.TmrToRemember.Enabled := True

else

FFruitGame.TmrCountdown.Enabled := True;

end;

end;

end;

procedure TFQuestion.LYesClick(Sender: TObject);

begin

FQuestion.Close;

case NumbOfParentForm of

0:

FMainMenu.Close;

1:

begin

if FColorGame.GameCount - 1 > FTop.TopColor[10].Points then

begin

FEnterName.Points := FColorGame.GameCount - 1;

FEnterName.Game := GColor;

FColorGame.Enabled := False;

FEnterName.Show;

end

else

FColorGame.Close;

end;

2:

begin

if FFruitGame.Points > FTop.TopFruit[10].Points then

begin

FEnterName.Points := FFruitGame.Points;

FEnterName.Game := GFruit;

FEnterName.Show;

end

else

FFruitGame.Close;

end;

end;

end;

end.

Модуль окна ввода имени:

unit EnterName;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.StdCtrls, Vcl.Imaging.jpeg, Vcl.ExtCtrls, TopPlayers, GameLib,

Vcl.Menus;

type

TFEnterName = class(TForm)

ImgBackground: TImage;

EdtName: TEdit;

NoPopupMenu: TPopupMenu;

LPoints: TLabel;

BtnOk: TLabel;

BtnNo: TLabel;

LTip: TLabel;

procedure BtnOkClick(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure EdtNameKeyPress(Sender: TObject; var Key: Char);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure BtnNoClick(Sender: TObject);

procedure EdtNameKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

private

function RightName: Boolean;

public

Points: Integer;

Game: TGame;

end;

var

FEnterName: TFEnterName;

implementation

{$R \*.dfm}

uses

ColorGame, FruitGame;

procedure TFEnterName.EdtNameKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

TEdit(Sender).ReadOnly := (Shift = [ssShift]) or (Shift = [ssCtrl]);

end;

procedure TFEnterName.EdtNameKeyPress(Sender: TObject; var Key: Char);

begin

LTip.Visible := False;

if (Length(EdtName.Text) > 15) and not (Ord(Key) in [127, 8]) then

Key := #0;

if not ((Key in ['A'..'Z', 'a'..'z', ' ', '0'..'9'])

or (Ord(Key) in [127, 8])

or ((Ord(Key) > $40F) and (Ord(Key) < $45F)) or (Ord(Key) = $451)) then

Key := #0;

end;

function TFEnterName.RightName: Boolean;

var

i: Integer;

Name: string;

begin

RightName := True;

Name := EdtName.Text;

if Length(Name) > 0 then

begin

i := Low(Name);

if Length(Name) <> 1 then

while (i <> (High(Name) - 1)) do

if (Name[i] = ' ') and (Name[i + 1] = ' ') then

Delete(Name, i, 1)

else

Inc(i);

while Name[1] = ' ' do

Delete(Name, 1, 1);

end;

EdtName.Text := Name;

if Length(Name) = 0 then

RightName := False;

end;

procedure TFEnterName.FormClose(Sender: TObject; var Action: TCloseAction);

begin

LPoints.Caption := '';

LTip.Visible := False;

EdtName.Clear;

end;

procedure TFEnterName.FormShow(Sender: TObject);

begin

LPoints.Caption := IntToStr(Points);

end;

procedure TFEnterName.BtnNoClick(Sender: TObject);

begin

FColorGame.Close;

FFruitGame.Close;

FEnterName.Close;

end;

procedure TFEnterName.BtnOkClick(Sender: TObject);

begin

if RightName then

begin

if Game = GColor then

begin

FTop.FindPositionInTop(Points, GColor);

FColorGame.Close;

end

else

begin

FTop.FindPositionInTop(Points, GFruit);

FFruitGame.Close;

end;

FEnterName.Close;

end;

LTip.Visible := True;

end;

end.

Модуль конца игры:

unit GameOver;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.StdCtrls, Vcl.Imaging.jpeg, Vcl.ExtCtrls;

type

TFGameOver = class(TForm)

ImgBackground: TImage;

BtnOk: TLabel;

procedure BtnOkClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FGameOver: TFGameOver;

implementation

{$R \*.dfm}

uses

FruitGame, ColorGame;

procedure TFGameOver.BtnOkClick(Sender: TObject);

begin

FGameOver.Close;

FFruitGame.Close;

FColorGame.Close;

end;

end.

Модуль окна справки:

unit Help;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.Imaging.pngimage, Vcl.ExtCtrls;

const

FirstPic = 0;

LastPic = 11;

type

TFHelp = class(TForm)

ImgSlide: TImage;

BtnExit: TImage;

BtnNext: TImage;

BtnPrev: TImage;

procedure BtnExitClick(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure BtnNextClick(Sender: TObject);

procedure BtnPrevClick(Sender: TObject);

private

Pic: Integer;

public

{ Public declarations }

end;

var

FHelp: TFHelp;

implementation

{$R \*.dfm}

uses

GameScreen;

procedure TFHelp.BtnExitClick(Sender: TObject);

begin

FMainMenu.Enabled := True;

if FMainMenu.Train[0].Left <> Start\_Point then

FMainMenu.TmrTrain.Enabled := True;

FHelp.Close;

end;

procedure TFHelp.BtnNextClick(Sender: TObject);

begin

if Pic <> LastPic then

begin

Inc(Pic);

BtnPrev.Visible := True;

ImgSlide.Picture.LoadFromFile('Img\Help\' + IntToStr(Pic) + '.png');

end;

if Pic = LastPic then

BtnNext.Visible := False;

end;

procedure TFHelp.BtnPrevClick(Sender: TObject);

begin

if Pic <> FirstPic then

begin

Dec(Pic);

BtnNext.Visible := True;

ImgSlide.Picture.LoadFromFile('Img\Help\' + IntToStr(Pic) + '.png');

end;

if Pic = FirstPic then

BtnPrev.Visible:= False;

end;

procedure TFHelp.FormShow(Sender: TObject);

begin

Pic := 0;

ImgSlide.Picture.LoadFromFile('Img\Help\0.png');

BtnPrev.Visible := False;

BtnNext.Visible := True;

end;

end.

Вспомогательный модуль:

unit GameLib;

interface

uses

Vcl.ExtCtrls;

type

TPlayer = record

Name: string[15];

Points: Integer;

end;

TTable = array of array of Integer;

TTurn = (Player, Computer);

TGame = (GColor, GFruit);

TStopPoint = (SPCountdown, SPGameProcess);

implementation

end.