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 «PAC-MAN»**

|  |  |  |
| --- | --- | --- |
| Студент |  | И.С. Андросов |
| Руководитель |  | Г.В. Данилова |

Минск 2019

СОДЕРЖАНИЕ

[Введение](#_Toc513665878) 5

[1 Анализ предметной области](#_Toc513665879) 7

[1.1 Обзор аналогов](#_Toc513665880) 7

[1.2 Постановка задачи](#_Toc513665881) 9

[2 Разработка программного средства](#_Toc513665882) 10

[2.1 Структура программы](#_Toc513665883) 10

[2.2 Интерфейс программного средства](#_Toc513665884) 10

[2.3 Звуковые и графические ресурсы](#_Toc513665885) 11

[2.3.1 Работа со звуком](#_Toc513665886) 11

[2.3.2 Работа с gif-изображениями](#_Toc513665887) 12

[2.4 Игровая логика](#_Toc513665888) 13

[2.5 Работа с таблицей рекордов](#_Toc513665889) 16

[3 Тестирование программного средства](#_Toc513665890) 19

[4 Руководство пользователя](#_Toc513665891) 21

[4.1 Правила игры](#_Toc513665892) 21

[4.2 Интерфейс программы](#_Toc513665893) 22

[Заключение](#_Toc513665894) 24

[Список использованных источников](#_Toc513665895) 25

[Приложение А. Исходный код программы](#_Toc513665896) 26

ВВЕДЕНИЕ

21-й век – век информационных и телекоммуникационных технологий. Всё больше отраслей специализируются в компьютерной сфере, так как это наиболее актуальное и современное направление. Новейшие научные достижения безудержно преображают мир вокруг нас. Большим изменениям подверглась и сфера развлечений, огромной частью которой стали видеоигры.

Так, английским ученым А. Дугласом была представлена в 1952-м году кандидатская диссертация на тему взаимодействия человека и компьютера. В качестве примера он создал программу игры «крестики-нолики», что уже явилось настоящей компьютерной игрой, где графика, хоть и неподвижная, выводилась на экран. В октябре 1958-го года появился еще один претендент в первые видеоигры – «Теннис для двоих», созданный физиком из Нью-Йорка В. Хигинботамом. К сожалению, все эти и ряд подобных проектов не получили широкого применения.

Автором идеи видеоигр считают американского изобретателя Ральфа Бэра, родившегося в 1922-м году в Германии. Седьмого мая 1967-го года Бэр со своим подчиненным Харрисоном продемонстрировал совету директоров Sanders первую видеоигру. В мае 1972-го года игровое устройство Бэра под названием «Magnavox Odyssey» появилось в продаже. В 1961-м году была создана «SpaceWar», в которой два корабля перемещались по всему экрану и обстреливали друг друга снарядами. Таким образом, первым в мире жанром стала аркада.

Концепция игры «Pac-Man» была разработана сотрудником японской компании [Namco](https://ru.wikipedia.org/wiki/Namco) [Тору Иватани](https://ru.wikipedia.org/wiki/%D0%98%D0%B2%D0%B0%D1%82%D0%B0%D0%BD%D0%B8,_%D0%A2%D0%BE%D1%80%D1%83). Он окончил колледж в 1977-м году, но при этом был самоучкой без формального образования в областях информатики, изобразительного искусства и дизайна.

В том же году [Taito](https://ru.wikipedia.org/wiki/Taito) выпустила «[Space Invaders](https://ru.wikipedia.org/wiki/Space_Invaders)», которая пользовалась успехом и изменила рынок видеоигр в сторону аркады. Тогда Тору, наблюдая за рынком видеоигр, пришёл к выводу, что тот ориентирован в первую очередь на мужчин. В большинстве своём игры представляли собой [шутеры](https://ru.wikipedia.org/wiki/%D0%A8%D1%83%D1%82%D0%B5%D1%80) на военную или космическую тематику, что делало их непопулярными среди женщин. Когда Тору заметил, что девушки играют в аркады только за компанию с друзьями или со своим молодым человеком, он решил разработать, с его слов, «милую» игру, способную понравиться всем. В итоге Иватани, считая, что тема еды привлекательна среди женщин, обратился к ней как к основе для создания «Pac-Man».

Первоначальной идеей Тору было разместить Пакмана в открытом пространстве, заполненном едой, но, придя к выводу, что игроку будет сложно понять, что делать, разработчик принял решение поместить персонажа в лабиринт, чтобы дать игроку направление. Имя героя образовано от японского [звукоподражательного выражения](https://ru.wikipedia.org/wiki/%D0%9E%D0%BD%D0%BE%D0%BC%D0%B0%D1%82%D0%BE%D0%BF%D0%B5%D1%8F) «паку-паку», обозначающего открывание и закрывание рта во время еды.

Иватани не умел программировать, поэтому, начиная с разработки «Gee Bee», он работал в паре с другим разработчиком – программистом, что было нетипично для того времени, когда обычно один человек и работал над дизайном игры и программировал её. Над «Pac-Man» с Тору работала команда из девяти человек, написавшая игру на языке ассемблера.

Первый аркадный автомат был установлен 22-го мая 1980-го года в кинотеатре района [Сибуя](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B1%D1%83%D1%8F_(%D1%81%D0%BF%D0%B5%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%80%D0%B0%D0%B9%D0%BE%D0%BD)), [Токио](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BA%D0%B8%D0%BE). «Pac-Man» привлёк к себе внимание: в первый же год было продано более 100 тысяч аркадных автоматов. Всего в мире продажи оцениваются в 400 тысяч.

Целью данного курсового проекта является разработка игрового приложения «Pac-Man».

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ
   1. Обзор аналогов

На данный момент существует множество вариаций игры «Pac-Man»: в бесконечном режиме, c закрытыми секциями лабиринта, с персонажем женского пола в главной роли.

Самой успешной и популярной является оригинальная версия «Pac-Man». Свою популярность игра получила за динамичный игровой процесс, персонажей с уникальными именами и отсутствие насильственного мотива. Внешний вид данного игрового средства представлен на рисунке 1.1.

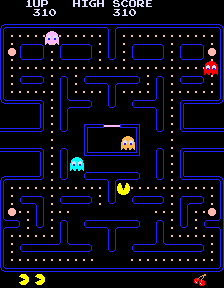


Рисунок 1.1 – «Pac-Man» в аркадной версии игры

Данная версия также знаменита своей ошибкой, именуемая «256-м уровнем». На 256-м уровне левая половина лабиринта отображается корректно, а правая представляет из себя беспорядочный набор разноцветных цифр, букв и других символов. Помимо этого, в правой части представлена иная архитектура лабиринта, которая невидима для игрока. Причиной этой ошибки является [процедура](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), выполняющая подсчёт бонусов под лабиринтом. При достижении 256-го уровня она загружает в [регистр центрального процессора](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0) значение 255 и увеличивает его на единицу. 255 является максимальным значением, которое способно уложиться в регистре процессора [Zilog Z80](https://ru.wikipedia.org/wiki/Zilog_Z80), поэтому при увеличении на единицу происходит [целочисленное переполнение](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D0%B5%D1%80%D0%B5%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5) и в регистр записывается ноль.

По замыслу разработчиков, у «Pac-Man» не должно было быть конца, так как возрастающая сложность игры не позволила бы продвинуться далеко. Однако игроки обнаружили, что, начиная с 21-го уровня и далее, поведение призраков становилось одинаковым. Это привело к идее создания шаблонов движения для Пакмана, которые позволяли проходить уровни по заученному маршруту. Так, некоторые игроки смогли пройти 255 уровней игры, что занимало несколько часов, и на 256-м уровне обнаружить проявление программной ошибки в виде «[экрана смерти](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%80%D0%B0%D0%BD_%D1%81%D0%BC%D0%B5%D1%80%D1%82%D0%B8)», о существовании которого не подозревали разработчики игры.

Ниже представлены некоторые аналоги:

– на рисунке 1.2 «Ms. Pac-Man», c женским персонажем в главной роли;

– на рисунке 1.3 «Super Pac-Man», с закрытыми секциями лабиринта.

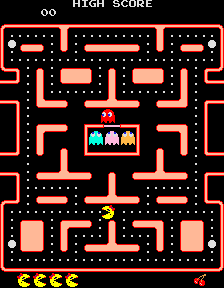


Рисунок 1.2 – «Ms. Pac-Man», аналог «Pac-Man» c женским персонажем

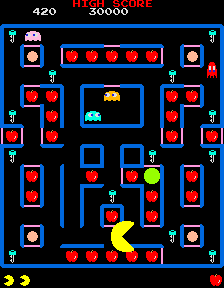


Рисунок 1.3 – «Pac-Man» с заблокированными секциями лабиринта

* 1. Постановка задачи

В рамках данного курсового проекта планируется разработка игрового программного средства «Pас-Man».

Будут разработаны алгоритмы заполнения и прорисовки игрового поля и взаимодействия с пользователем.

В игровом средстве планируется реализовать следующие функции:

* постепенное увеличение сложности;
* визуализация таблицы рекордов;
* звуковое сопровождение.

Логическая составляющая приложения будет включать в себя:

* различные состояния клеток поля;
* размещение объектов на поле;
* алгоритмы поиска и поимки персонажа;
* взаимодействие структурных блоков приложения.

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 семь структурных блоков:

– MainMenuUnit – модуль, отображающий главное окно программы;

– RecordsUnit – модуль, отображающий окно таблицы рекордов;

– AboutUnit – модуль, отображающий окно информации о приложении;

– GameUnit – основной логический модуль;

– ConfirmUnit – модуль, управляющий результатами игры.

* 1. Интерфейс программного средства

Внешний вид и удобность в использовании являются одними из главных критериев качества программного средства. Поэтому взаимодействие приложения с пользователем необходимо организовать максимально интуитивно и просто.

В качестве навигации был выбран очень простой и понятный большинству пользователей компонент TButton. Для взаимодействия с пользователем установлены следующие опции:

– «New game»;

– «Records»;

– «About»;

– «Exit».

На рисунке 2.1 представлено отображение главного окна программы.

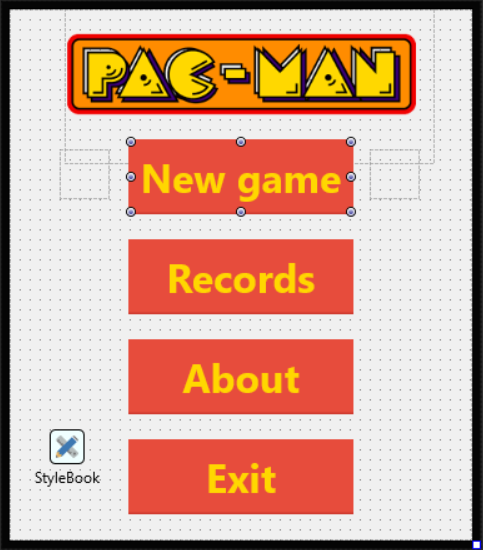


Рисунок 2.1 – Главное окно программы

Для лучшего визуального восприятия интерфейса был разработан специальный стиль для компонента TButton при помощи TStyleBook.

Стандартное оформление компонента TButton не удовлетворяло цветовой гамме игрового средства, в связи с этим было принято решение использовать красные оттенки.

В результате реализации были отрисованы три состояния кнопки: «NormalColor», «PressedColor» и «FocusedColor». На рисунке 2.2 представлены соответствующие отображения кнопок.



Рисунок 2.2 – Отображения состояний кнопок

Для разборчивости текста был выбран шрифт Segoe UI, являющийся стандартным в среде разработки Embarcadero Delphi 10.2 Tokyo, 32-го размера и золотого цвета.

Все эти настройки позволили реализовать приятный глазу интерфейс программного средства, который отлично вписывается в стилистику игры и не загромождает экран лишней информацией.

* 1. Звуковые и графические ресурсы

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

1. Работа со звуком

Управление звуком реализовано при помощи стандартной библиотеки Mmsystem и предоставлено функцией PlaySound. Различные случаи использования данной функции приведены ниже.

**procedure** TStartMenu.FormCreate(Sender: TObject);

**begin**

PlaySound('Audio\Menu.wav', 0, SND\_ASYNC **or** SND\_LOOP);

**end**;

В качестве параметров передается файл в виде строки, который будет воспроизводиться, параметр использования ресурса, а также специальные флаги SND\_ASYNC и SND\_LOOP, определяющие асинхронный и циклический характеры воспроизведения.

PlaySound(0, 0, SND\_PURGE);

Данный пример использования функции PlaySound позволяет прервать воспроизведение звука при наличии флага SND\_PURGE.

1. Работа с gif-изображениями

Визуальные эффекты включают в себя gif-изображения, сопровождающие весь игровой процесс.

На основе анализа имеющихся способов реализации работы с gif-изображениями был выбран метод, заключающийся в использовании компонента TImage, который взаимодействует с объектом класса TBitmapAnimation.

Использование данного компонента удобно в реализации данного игрового программного средства, благодаря объёмному количеству свойств компонента TBitmapAnimation.

Количество кадров в свойствах AnimationBitmap и AnimationCount напрямую влияют на качество анимации, но также и на производительность проекта.

На рисунке 2.3 представлены настройки свойств компонента TBitmapAnimation заднего плана главного окна программы.

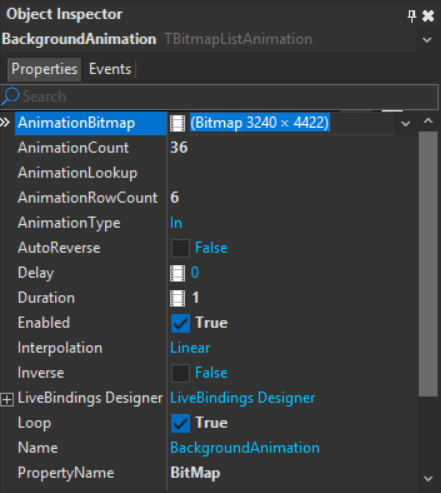


Рисунок 2.3 – Настройки свойств компонента TBitmapAnimation

Важным является установка свойств Loop и Enabled в значение True, так это обеспечивает непрерывное воспроизведение gif-анимации.

* 1. Игровая логика

Работу данной части программы обеспечивает модуль GameUnit.

В качестве структуры для хранения информации об игровом поле был выбран статический массив, содержащий элементы типа TTile, где TTile – запись, содержащая поля Wall, Eaten и Dot.

TTile = **record**

Wall: Boolean;

Eaten: Boolean;

Dot: TCircle;

**end**;

При создании игрового поля происходит инициализация массива процедуре TField.FormCreate, что позволяет в дальнейшем избежать ошибки некорректного обращения к свойствам игрового поля и выхода за границы массива.

В начале каждой новой игры всем персонажам задаётся первоначальное положение. Свойства элементов массива Tiles приобретают значения True или False в зависимости от значений статического двумерного массива TilesRepresentation, состоящего из нулей и единиц.

Помимо установки необходимого числа точек, представленных компонентом TCircle, происходит выставление текущего игрового рекорда, обнуление очков и восстановление количества жизней.



Рисунок 2.4 – Блок-схема инициализации изображений



Рисунок 2.5 – Блок-схема инициализации новой игры

Логика передвижения Пакмана по игровому полю реализована в процедуре TField.CheckMove:

**if** (NewY > PacMan.Y) **then**

Pac.RotationAngle := 90;

**if** (NewY < PacMan.Y) **then**

Pac.RotationAngle := -90;

**if** (NewX > PacMan.X) **then**

Pac.RotationAngle := 0;

**if** (NewX < PacMan.X) **then**

Pac.RotationAngle := 180;

Данный участок процедуры отвечает за поворот модели персонажа в сторону передвижения. После этого осуществляется проверка нахождения на краю игрового поля, а затем само перемещение.

**if** (NewX = -1) **or** (NewX = 28) **then** //проверка координат Пакмана

**if** (NewX = -1) **then**

**begin** //перемещение на правый край игрового поля

PacMan.X := 28;

Pac.Position.X := 28 \* 20;

Move(NewY, 27);

**if not**(Tiles[NewY, 27].Eaten) **then**

**begin**

PlaySound('Audio\Eating.wav', 0, SND\_ASYNC **or**

SND\_NOSTOP);

ScoreValueLbl.Text :=

IntToStr(StrToInt(ScoreValueLbl.Text) + 5);

Tiles[NewY][27].Dot.Visible := false;

Tiles[NewY, 27].Eaten := true;

**end**;

**end**

**else**

**begin** //перемещение на левый край игрового поля

PacMan.X := -1;

Pac.Position.X := -1 \* 20;

Move(NewY, 0);

**if not**(Tiles[NewY, 0].Eaten) **then**

**begin**

PlaySound('Audio\Eating.wav', 0, SND\_ASYNC **or**

SND\_NOSTOP);

ScoreValueLbl.Text :=

IntToStr(StrToInt(ScoreValueLbl.Text) + 5);

Tiles[NewY][0].Dot.Visible := false;

Tiles[NewY, 0].Eaten := true;

**end**;

**end**

**else**

**if not**(Tiles[NewY, NewX].Wall) **then**

**if not**(Tiles[NewY, NewX].Eaten) **then**

**begin** //поедание точки

Move(NewY, NewX);

PlaySound('Audio\Eating.wav', 0, SND\_ASYNC **or**

SND\_NOSTOP);

ScoreValueLbl.Text :=

IntToStr(StrToInt(ScoreValueLbl.Text) + 5);

Tiles[NewY][NewX].Dot.Visible := false;

Tiles[NewY, NewX].Eaten := true;

**end**

**else** //перемещение на пустую клетку

Move(NewY, NewX);

* 1. Работа с таблицей рекордов

Работа с таблицей рекордов реализована в модуле RecordsUnit.

Для операций с рекордами используются две структуры данных.

Первая – TPlayer – запись c полями Name, содержащее имя пользователя, и Score, хранящее численное значение рекорда. Данная структура используется в качестве базовой, так как ее элементы могут быть записаны в файл, а также загружены из него.

Вторая структура – одномерный массив, элементами которого являются записи типа TPlayer. Данный массив хранит информацию о таблице рекордов на протяжении работы программы.

При запуске программы происходит проверка наличия типизированного файла, в котором хранится информация о игроках и их рекордах. Если такой файл не обнаружен, то происходит создание нового файла с рекордами.

Описание данной процедуры указано ниже.

**procedure** TRecordsForm.FormCreate(Sender: TObject);

**var**

i: Byte;

Temp: TPlayer;

**begin**

AssignFile(RecordsFile, 'DontTouch.pcm');

**try** //проверка на наличие файла

Reset(RecordsFile);

**except**

Rewrite(RecordsFile);

**end**;

Temp.Name := '...........';

Temp.Score := '0000';

i := 0;

**for** i := 0 **to** 4 **do** //заполнение массива данными

**if not**(Eof(RecordsFile)) **then**

Read(RecordsFile, RecordsDataBase[i])

**else**

RecordsDataBase[i] := Temp;

CloseFile(RecordsFile);

**end**;

Работа с массивом записей начинается при очередной смене лучшего рекорда. Максимальное количество отображаемых игроков, добившихся наилучшего результата к моменту окончания игры, равняется пяти, поэтому при перезаписи массива информация о игроке на последнем месте удаляется. Данная логика реализована в процедуре TConfirmForm.ConfirmBtnClick, участок кода которой представлен ниже.

**if** ConfirmBtn.Text = 'Confirm' **then**

**With** RecordsForm **do**

**begin**

**for** i := 3 **Downto** 0 **do** //обновление данных массива

RecordsDataBase[i + 1] := RecordsDataBase[i];

RecordsDataBase[0].Name := NameEdit.Text;

RecordsDataBase[0].Score := Field.ScoreValueLbl.Text;

Rewrite(RecordsFile);

**for** i := 0 **to** 4 **do** //сохранение данных в файл

**if not**(RecordsDataBase[i].Name = '...........') **then**

Write(RecordsFile, RecordsDataBase[i]);

CloseFile(RecordsFile);

**end**;

Close;

Если же игрок не смог побить лучший рекорд, то предложения ввести имя пользователя не поступает. Сообщение и имеет другой характер, направленный на повторную попытку побить рекорд. Внешний вид данной ситуации формы ConfirmForm модуля RecordsUnit, представлен на рисунке 2.6.

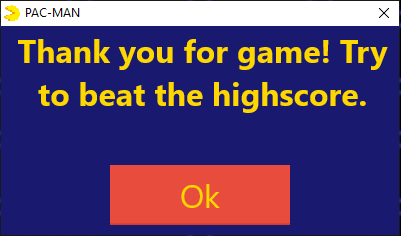


Рисунок 2.6 – Внешний вид формы ConfirmForm при неудаче

Вторая ситуация – победа пользователя. При этом пользователю показывается диалоговое окно с просьбой ввода имени. Внешний вид данной ситуации формы ConfirmForm модуля RecordsUnit, представлен на рисунке 2.7.

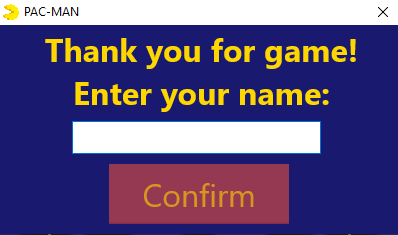


Рисунок 2.7 – Внешний вид формы ConfirmForm при победе

Для просмотра таблицы рекордов нужно воспользоваться опцией «Records» на главном окне программ, внешний вид которой представлен на рисунке 2.8. Процедура TRecordsForm.FormShow, используя данные массива записей отображает поставленные ранее рекорды. Блок-схема данной процедуры представлена на рисунке 2.9.



Рисунок 2.8 – Окно формы RecordsForm



Рисунок 2.9 – Блок-схема процедуры TRecordsForm.FormShow

1. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В ходе тестирования приложения были выявлены некоторые недостатки интерфейса программного средства.

Была обнаружена проблема при нажатии клавиш (UP), (DOWN), (LEFT) и (RIGHT) для перемещения персонажа по игровому полю. Если во время не завершившейся анимации перемещения персонажа попытаться переместиться на другую позицию, то это приводило к перемещению на клетку со стенкой и неправильной логике поведения призраков. Изменение было связано с проверкой на активную анимацию перемещения персонажа. Ниже приведен изменённый вариант кода данной процедуры.

**if not**(PacXAnimation.Enabled **or** PacYAnimation.Enabled) **then**

**case** key **of**

VkUp: CheckMove(PacMan.Y - 1, PacMan.X);

VkDown: CheckMove(PacMan.Y + 1, PacMan.X);

VkLeft: CheckMove(PacMan.Y, PacMan.X - 1);

VkRight: CheckMove(PacMan.Y, PacMan.X + 1);

**end**;

Второй сложностью стало поведение призрака, имеющего тип TBlinky. Его поведение заключается в периодическом появлении около главного персонажа. Это приводило к ошибкам, указанным выше. Корректная версия данного блока представлена ниже.

**if** (NewX < 27) **and** (NewX > 0) **and** (NewY < 30) **and** (NewY > 0) **and**

**not**(Tiles[NewY][NewX].Wall) **then**

**begin**

Blinky.X := NewX;

Blinky.Y := NewY;

BlinkyImg.Position.Y := NewY \* 20 + 50;

BlinkyImg.Position.X := NewX \* 20;

**end**;

Ещё одним недостатком стала некорректная логика перемещения призрака типа TClyde, использующего положение Пакмана. При изначальном сравнивании координат призрак полностью игнорировал клетки со стенками, быстро добираясь до персонажа. Если же удавалось оказаться выше призрака, то он полностью останавливался, перекрывая путь для дальнейшего перемещения по игровому полю. Данная ошибка была связана с неправильной проверкой координат и расчётом длительности перемещения до следующего поворота. Исправленный код данного участка представлен ниже.

**if** (NewY <> Clyde.Y) **then** //проверка координат

**begin**

**if** (NewY > Clyde.Y) **then** //перемещение вниз

**...**

**else** //перемещение вверх

...

**end**

**else**

**begin**

**if** (NewX > Clyde.X) **then** //перемещение вправо

**...**

**else** //перемещение влево

**...**

**end**;

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

1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
   1. Правила игры

Правила полностью соответствуют классической концепции игры «Pac-Man». Пользователю необходимо съесть как можно больше точек на игровом поле, играя за Пакмана, и избегать призраков. Визуальное отображение Пакмана изображено на рисунке 4.1.

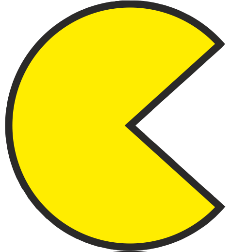


Рисунок 4.1 – Визуальное отображение Пакмана

Передвижение по игровому полю осуществляется при помощи клавиш (UP), (DOWN), (LEFT) и (RIGHT).

Поедание точки производится перемещением на клетку с точкой. Также можно перемещаться с одного края игрового поля на другой.

При третьей поимке призраками игра оканчивается.

Все призраки имеют различные имена, характерные цвета, модели поведения и появляются при различных значениях очков. На рисунке 4.2 представлено визуальное отображение призраков.

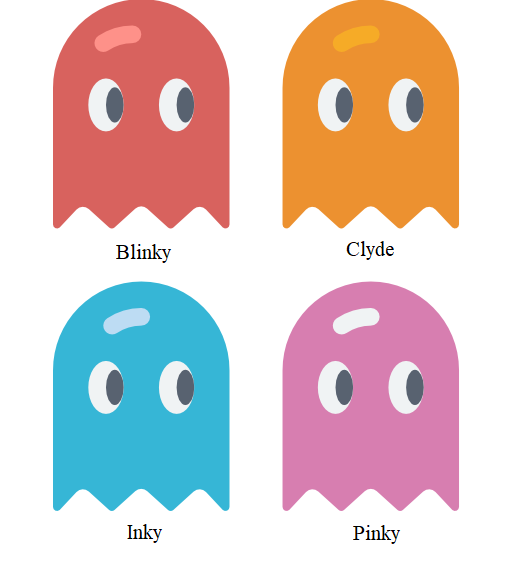


Рисунок 4.2 – Визуальное отображение призраков

Когда все точки съедены, победа пользователю не засчитывается до тех пор, пока он не будет пойман.

В случае победы таблица рекордов будет автоматически обновлена с учётом последнего результата.

* 1. Интерфейс программы

Главное окно программы, изображенное на рисунке 4.4, содержит следующие элементы:

– «New game»;

– «Records»;

– «About»;

– «Exit».

Для начала игры необходимо выбрать любой из пунктов меню «New game». Игра запускается автоматически после нажатия любой клавиши. Позиции точек задаются в начале игры и заканчиваются по мере игрового процесса. Игровое поле изображено на рисунке 4.5.

В программе присутствует звуковое и визуальное сопровождение. Проверка и загрузка ресурсов происходит в момент запуска приложения. При этом поиск файлов осуществляется непосредственно в директории программы.

В случае некорректной загрузки какого-либо из ресурсов будет выведено соответствующее сообщение, которое представлено на рисунке 4.3, и некоторые эффекты станут недоступны.

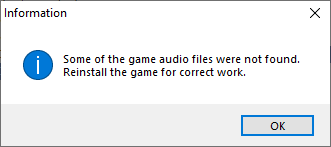


Рисунок 4.3 – Сообщение при некорректной загрузки ресурса

Данная проблема может быть исправлена путем добавления или замены группы файлов, соответствующих каждому элементу сопровождения.

Для корректной работы программы также необходим файл с таблицей рекордов. В случае отсутствия файл будет создан автоматически. При этом значения в таблице будут установлены по умолчанию.



Рисунок 4.4 – Внешний вид главного окна программы

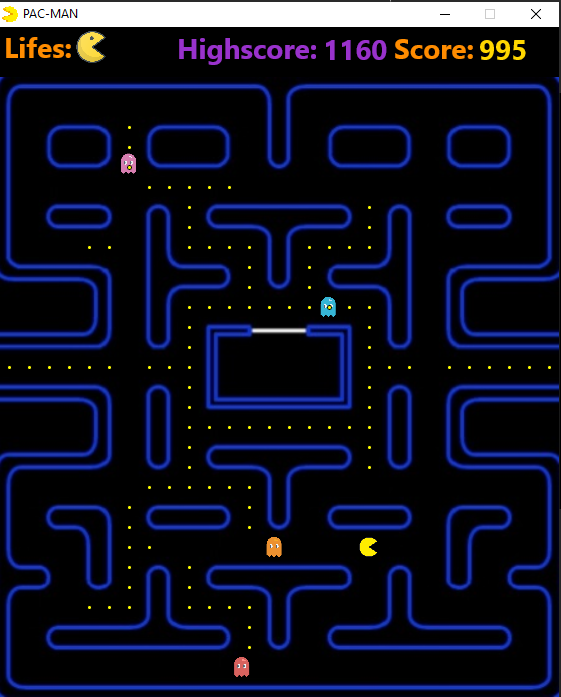


Рисунок 4.5 – Внешний вид игрового поля программы

ЗАКЛЮЧЕНИЕ

Популярность видеоигр не прекращает расти. Немалая часть из них не несет никакой смысловой нагрузки, однако жанр аркады не относится к таковым. Такие игры как «Limbo», «Super Mario» и подобные способствуют развитию внимания, реакции и логического мышления, если не проводить у экрана излишне много времени.

В рамках данного курсового проекта было разработано игровое программное средство «Pac-Man», которое обеспечит веселое и увлекательное времяпрепровождение. Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

* постепенное увеличение сложности;
* визуализация таблицы рекордов;
* звуковое сопровождение.

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

Существует много возможностей для дальнейшего улучшения приложения. Одним из самых простых направлений является разработка новых игровых алгоритмов и введение других игровых уровней. Также возможна переработка визуальной концепции игры, как например, переход к 3D-изображениям. Ещё одним вариантом развития является адаптация проекта для запуска на устройствах с низкой разрешающей способностью экрана и переход к кроссплатформенной разработке.

Использование данного приложения позволит не только провести время с удовольствием, но также и развить свою реакцию, логическое мышление и способность просчитывать действия на несколько шагов вперед.

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. Фараонов, В. В. Delphi 6. Учебный курс: / В. В. Фараонов. – СПб: изд. С. В. Малгачёва, 2001. – 231 с.
3. Шупрута, В. В. Delphi 2005. Учимся программировать: / В. В. Шупрута. – Москва: изд. «НТ Пресс», 2001. – 140 с.
4. RAD Studio Product Documentation – Embarcadero Technologies [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://docs.embarcadero.com/products/rad_studio>.
5. Программирование на языке Delphi / Д. А. Сурков [и др.]. – учеб. пособие. – Режим доступа: http://www.rsdn.ru/?summary/3165.xml, – 2005.
6. Орлов, С. А. Технологии разработки программного обеспечения: учеб. Пособие. – СПб, 2003.
7. Уилсон, С. Принципы проектирования и разработки программного обеспечения, yчебн. курс. – СПб, 2003.

ПРИЛОЖЕНИЕ А

Исходный код программы

**MainMenuUnit.pas**

**unit** MainMenuUnit;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants, FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics,

FMX.Dialogs, FMX.StdCtrls, FMX.Controls.Presentation, FMX.Objects,

FMX.Ani, Vcl.Dialogs, RecordsUnit, Mmsystem, WinAPI.Windows;

**type**

TStartMenu = **class**(TForm)

RecordsBtn: TButton;

AboutBtn: TButton;

ExitBtn: TButton;

PacMoveAnimation: TFloatAnimation;

Background: TImage;

BackgroundAnimation: TBitmapListAnimation;

Logo: TImage;

Pac: TImage;

Gost: TImage;

NewGameBtn: TButton;

StyleBook1: TStyleBook;

GostAnimation: TBitmapListAnimation;

GostMoveAnimation: TFloatAnimation;

PacAnimation: TBitmapListAnimation;

**procedure** ExitBtnClick(Sender: TObject);

**procedure** FormCloseQuery(Sender: TObject; **var** CanClose: Boolean);

**procedure** AboutBtnMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Single);

**procedure** RecordsBtnMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Single);

**procedure** ExitBtnMouseMove(Sender: TObject; Shift: TShiftState; X, Y:

Single);

**procedure** NewGameBtnMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Single);

**procedure** FormCreate(Sender: TObject);

**procedure** NewGameBtnClick(Sender: TObject);

**procedure** RecordsBtnClick(Sender: TObject);

**procedure** AboutBtnClick(Sender: TObject);

**end**;

**var**

StartMenu: TStartMenu;

**implementation**

{$R \*.fmx}

**uses** GameUnit, AboutUnit;

**procedure** TStartMenu.NewGameBtnClick(Sender: TObject);

**var**

i, j: ShortInt;

**begin**

Field.HighscoreValueLbl.Text := RecordsForm.RecordsDataBase[0].Score;

Pacman.Lifes := 3;

Field.Life\_3.Visible := true;

Field.Life\_2.Visible := true;

Field.ScoreValueLbl.Text := '0000';

**for** i := 0 **to** 30 **do**

**for** j := 0 **to** 27 **do**

**With** Tiles[i][j] **do**

**if** Eaten **then**

**begin**

Eaten := false;

Dot.Visible := true;

**end**;

Field.Setup;

PlaySound(0, 0, SND\_PURGE);

PlaySound('Audio\Intro.wav', 0, SND\_ASYNC);

StartMenu.Hide;

Field.Showmodal;

**end**;

**procedure** TStartMenu.NewGameBtnMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Single);

**begin**

PacMoveAnimation.StopAtCurrent;

PacMoveAnimation.Enabled := false;

PacMoveAnimation.StartValue := Pac.Position.Y;

PacMoveAnimation.StopValue := 112;

PacMoveAnimation.Enabled := true;

GostMoveAnimation.StopAtCurrent;

GostMoveAnimation.Enabled := false;

GostMoveAnimation.StartValue := Pac.Position.Y;

GostMoveAnimation.StopValue := 112;

GostMoveAnimation.Enabled := true;

**end**;

**procedure** TStartMenu.RecordsBtnClick(Sender: TObject);

**begin**

StartMenu.Hide;

RecordsForm.Showmodal;

**end**;

**procedure** TStartMenu.RecordsBtnMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Single);

**begin**

PacMoveAnimation.StopAtCurrent;

PacMoveAnimation.Enabled := false;

PacMoveAnimation.StartValue := Pac.Position.Y;

PacMoveAnimation.StopValue := 192;

PacMoveAnimation.Enabled := true;

GostMoveAnimation.StopAtCurrent;

GostMoveAnimation.Enabled := false;

GostMoveAnimation.StartValue := Pac.Position.Y;

GostMoveAnimation.StopValue := 192;

GostMoveAnimation.Enabled := true;

**end**;

**procedure** TStartMenu.AboutBtnClick(Sender: TObject);

**begin**

StartMenu.Hide;

AboutForm.Showmodal;

**end**;

**procedure** TStartMenu.AboutBtnMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Single);

**begin**

PacMoveAnimation.StopAtCurrent;

PacMoveAnimation.Enabled := false;

PacMoveAnimation.StartValue := Pac.Position.Y;

PacMoveAnimation.StopValue := 272;

PacMoveAnimation.Enabled := true;

GostMoveAnimation.StopAtCurrent;

GostMoveAnimation.Enabled := false;

GostMoveAnimation.StartValue := Pac.Position.Y;

GostMoveAnimation.StopValue := 272;

GostMoveAnimation.Enabled := true;

**end**;

**procedure** TStartMenu.ExitBtnMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Single);

**begin**

PacMoveAnimation.StopAtCurrent;

PacMoveAnimation.Enabled := false;

PacMoveAnimation.StartValue := Pac.Position.Y;

PacMoveAnimation.StopValue := 352;

PacMoveAnimation.Enabled := true;

GostMoveAnimation.StopAtCurrent;

GostMoveAnimation.Enabled := false;

GostMoveAnimation.StartValue := Pac.Position.Y;

GostMoveAnimation.StopValue := 352;

GostMoveAnimation.Enabled := true;

**end**;

**procedure** TStartMenu.ExitBtnClick(Sender: TObject);

**begin**

Close;

**end**;

**procedure** TStartMenu.FormCloseQuery(Sender: TObject; **var** CanClose: Boolean);

**var**

ButtonSelected: Byte;

**begin**

ButtonSelected := MessageDlg('Are you sure you want to exit?', mtConfirmation, [mbYes,mbNo], 0);

**if** ButtonSelected <> mrYes **then**

CanClose := False;

**end**;

**procedure** TStartMenu.FormCreate(Sender: TObject);

**var**

CheckFile: **File**;

**begin**

**if not**(FileExists('Audio\Menu.wav'))

**or not**(FileExists('Audio\Intro.wav'))

**or not**(FileExists('Audio\Win.wav'))

**or not**(FileExists('Audio\Eating.wav'))

**or not**(FileExists('Audio\Death.wav')) **then**

MessageDlg('Some of the game audio files were not found.' + #13#10 +

'Reinstall the game for correct work.', mtInformation,

[mbOk], 0);

PlaySound('Audio\Menu.wav', 0, SND\_ASYNC **or** SND\_LOOP);

**end**;

**end**.

**GameUnit.pas**

**unit** GameUnit;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants, FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics,

FMX.Dialogs, FMX.Ani, FMX.Objects, Mmsystem, FMX.Controls.Presentation,

FMX.StdCtrls, MainMenuUnit, Vcl.Dialogs, ConfirmUnit;

**type**

TField = **class**(TForm)

FieldImg: TImage;

Pac: TImage;

PacYAnimation: TFloatAnimation;

PacAnimation: TBitmapListAnimation;

Life\_1: TImage;

Life\_2: TImage;

Life\_3: TImage;

LifesLbl: TLabel;

ScoreLbl: TLabel;

ScoreValueLbl: TLabel;

Background: TImage;

PacXAnimation: TFloatAnimation;

PinkyImg: TImage;

GameTimer: TTimer;

PinkyAnimation: TBitmapListAnimation;

PinkyXAnimation: TFloatAnimation;

PinkyYAnimation: TFloatAnimation;

ClydeImg: TImage;

ClydeAnimation: TBitmapListAnimation;

ClydeXAnimation: TFloatAnimation;

ClydeYAnimation: TFloatAnimation;

InkyImg: TImage;

InkyAnimation: TBitmapListAnimation;

InkyXAnimation: TFloatAnimation;

InkyYAnimation: TFloatAnimation;

BlinkyImg: TImage;

BlinkyAnimation: TBitmapListAnimation;

BlinkyXAnimation: TFloatAnimation;

BlinkyYAnimation: TFloatAnimation;

BlinkyColdown: TTimer;

HighscoreValueLbl: TLabel;

HighscoreLbl: TLabel;

**procedure** SetImages;

**procedure** GetClydeMove;

**procedure** GetInkyMove;

**procedure** GetBlinkyMove;

**procedure** InitialiseResults;

**procedure** FormCloseQuery(Sender: TObject; **var** CanClose: Boolean);

**procedure** FormKeyDown(Sender: TObject; **var** Key: Word; **var** KeyChar:

Char; Shift: TShiftState);

**procedure** Move(NewY, NewX: ShortInt);

**procedure** CheckMove(NewY, NewX: ShortInt);

**procedure** PacYAnimationFinish(Sender: TObject);

**procedure** PacXAnimationFinish(Sender: TObject);

**procedure** GameTimerTimer(Sender: TObject);

**procedure** GetPinkyMove;

**procedure** PinkyMove(NewY, NewX: ShortInt);

**procedure** CheckPinckyMove(NewY, NewX: ShortInt);

**procedure** PinkyXAnimaionFinish(Sender: TObject);

**procedure** PinkyYAnmationFinish(Sender: TObject);

**procedure** Setup;

**procedure** FormCreate(Sender: TObject);

**procedure** PinkyXAnimationProcess(Sender: TObject);

**procedure** PinkyYAnimationProcess(Sender: TObject);

**procedure** CheckClydeMove(NewY, NewX: ShortInt);

**procedure** ClydeMove(NewY, NewX: ShortInt);

**procedure** ClydeXAnimationFinish(Sender: TObject);

**procedure** ClydeXAnimationProcess(Sender: TObject);

**procedure** ClydeYAnimationFinish(Sender: TObject);

**procedure** ClydeYAnimationProcess(Sender: TObject);

**procedure** CheckInkyMove(NewY, NewX: ShortInt);

**procedure** InkyMove(NewY, NewX: ShortInt);

**procedure** InkyXAnimationFinish(Sender: TObject);

**procedure** InkyXAnimationProcess(Sender: TObject);

**procedure** InkyYAnimationFinish(Sender: TObject);

**procedure** InkyYAnimationProcess(Sender: TObject);

**procedure** CheckBlinkyMove(NewY, NewX: ShortInt);

**procedure** BlinkyMove(NewY, NewX: ShortInt);

**procedure** BlinkyXAnimationFinish(Sender: TObject);

**procedure** BlinkyXAnimationProcess(Sender: TObject);

**procedure** BlinkyYAnimationFinish(Sender: TObject);

**procedure** BlinkyYAnimationProcess(Sender: TObject);

**procedure** BlinkyColdownTimer(Sender: TObject);

**end**;

TPacMan = **record**

X: ShortInt;

Y: ShortInt;

Lifes: ShortInt;

**end**;

TPinky = **record**

X: ShortInt;

Y: ShortInt;

LastMove: Byte;

**end**;

TClyde= **record**

X: ShortInt;

Y: ShortInt;

InGame: Boolean;

**end**;

TInky = **record**

X: ShortInt;

Y: ShortInt;

InGame: Boolean;

**end**;

TBlinky = **record**

X: ShortInt;

Y: ShortInt;

InGame: Boolean;

LastMove: Byte;

**end**;

TTile = **record**

Wall: Boolean;

Eaten: Boolean;

Dot: TCircle;

**end**;

**const**

PinkySpeed = 0.17;

InkySpeed = 0.15;

ClydeSpeed = 0.2;

BlinkySpeed = 0.25;

TilesRepresentation: **array** [0..30] **of array** [0..27] **of** Byte =

((1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1),

(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1),

(1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,

1, 1, 1, 0, 1),

(1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,

1, 1, 1, 0, 1),

(1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,

1, 1, 1, 0, 1),

(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1),

(1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 1, 1, 0, 1),

(1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 1, 1, 0, 1),

(1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,

0, 0, 0, 0, 1),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 1, 1, 1, 1),

(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1),

(1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,

1, 1, 1, 0, 1),

(1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,

1, 1, 1, 0, 1),

(1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,

1, 0, 0, 0, 1),

(1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 0, 1, 1, 1),

(1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,

1, 0, 1, 1, 1),

(1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,

0, 0, 0, 0, 1),

(1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 0, 1),

(1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 0, 1),

(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1),

(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1));

**var**

Field: TField;

Tiles: **array** [0..30] **of array** [0..27] **of** TTile;

PacMan: TPacMan;

Pinky: TPinky;

Clyde: TClyde;

Inky: TInky;

Blinky: TBlinky;

**implementation**

{$R \*.fmx}

**uses** RecordsUnit;

**procedure** TField.GetPinkyMove;

**begin**

**if not**(PinkyXAnimation.Enabled **or** PinkyYAnimation.Enabled) **then**

**case** random(4) **of**

0: **if not**(Pinky.LastMove = 1) **then**

CheckPinckyMove(Pinky.Y - 1, Pinky.X);

1: **if not**(Pinky.LastMove = 0) **then**

CheckPinckyMove(Pinky.Y + 1, Pinky.X);

2: **if not**(Pinky.LastMove = 3) **then**

CheckPinckyMove(Pinky.Y, Pinky.X - 1);

3: **if not**(Pinky.LastMove = 2) **then**

CheckPinckyMove(Pinky.Y, Pinky.X + 1);

**end**;

**end**;

**procedure** TField.GetClydeMove;

**begin**

**if** Clyde.InGame **and not**(ClydeXAnimation.Enabled **or**

ClydeYAnimation.Enabled) **then**

**begin**

**if** (Clyde.Y > PacMan.Y) **and not**(ClydeXAnimation.Enabled **or**

ClydeYAnimation.Enabled) **then**

CheckClydeMove(Clyde.Y - 1, Clyde.X)

**else**

**if** (Clyde.Y < PacMan.Y) **and not**(ClydeXAnimation.Enabled **or**

ClydeYAnimation.Enabled) **then**

CheckClydeMove(Clyde.Y + 1, Clyde.X);

**if** (Clyde.X > PacMan.X) **and not**(ClydeXAnimation.Enabled **or**

ClydeYAnimation.Enabled) **then**

CheckClydeMove(Clyde.Y, Clyde.X - 1)

**else**

**if** (Clyde.X < PacMan.X) **and not**(ClydeXAnimation.Enabled **or**

ClydeYAnimation.Enabled) **then**

CheckClydeMove(Clyde.Y, Clyde.X + 1)

**end**;

**end**;

**procedure** TField.GetInkyMove;

**begin**

**if** Inky.InGame **and not**(InkyXAnimation.Enabled **or** InkyYAnimation.Enabled) **then**

**begin**

**if** (Inky.X > PacMan.X) **and not**(InkyXAnimation.Enabled **or**

InkyYAnimation.Enabled) **then**

CheckInkyMove(Inky.Y, Inky.X - 1)

**else**

**if** (Inky.X < PacMan.X) **and not**(InkyXAnimation.Enabled **or**

InkyYAnimation.Enabled) **then**

CheckInkyMove(Inky.Y, Inky.X + 1)

**else**

**begin**

**if not**(InkyXAnimation.Enabled **or** InkyYAnimation.Enabled) **then**

CheckInkyMove(Inky.Y - 1, Inky.X);

**if not**(InkyXAnimation.Enabled **or** InkyYAnimation.Enabled) **then**

CheckInkyMove(Inky.Y + 1, Inky.X);

**end**;

**if** (Inky.Y > PacMan.Y) **and not**(InkyXAnimation.Enabled **or**

InkyYAnimation.Enabled) **then**

CheckInkyMove(Inky.Y - 1, Inky.X)

**else**

**if** (Inky.Y < PacMan.Y) **and not**(InkyXAnimation.Enabled **or**

InkyYAnimation.Enabled) **then**

CheckInkyMove(Inky.Y + 1, Inky.X)

**else**

**begin**

**if not**(InkyXAnimation.Enabled **or** InkyYAnimation.Enabled) **then**

CheckInkyMove(Inky.Y, Inky.X - 1);

**if not**(InkyXAnimation.Enabled **or** InkyYAnimation.Enabled) **then**

CheckInkyMove(Inky.Y, Inky.X + 1);

**end**;

**end**;

**end**;

**procedure** TField.GetBlinkyMove;

**begin**

**if** Blinky.InGame **and not**(BlinkyXAnimation.Enabled **or**

BlinkyYAnimation.Enabled) **then**

**case** random(4) **of**

0: **if not**(Blinky.LastMove = 1) **then**

CheckBlinkyMove(Blinky.Y - 1, Blinky.X);

1: **if not**(Blinky.LastMove = 0) **then**

CheckBlinkyMove(Blinky.Y + 1, Blinky.X);

2: **if not**(Blinky.LastMove = 3) **then**

CheckBlinkyMove(Blinky.Y, Blinky.X - 1);

3: **if not**(Blinky.LastMove = 2) **then**

CheckBlinkyMove(Blinky.Y, Blinky.X + 1);

**end**;

**end**;

**procedure** TField.InitialiseResults;

**begin**

GameTimer.Enabled := false;

BlinkyColdown.Enabled := false;

**if** StrToInt(ScoreValueLbl.Text) >= StrToInt(HighscoreValueLbl.Text) **then**

**begin**

PlaySound('Audio\Win.wav', 0, SND\_ASYNC);

ConfirmForm.NameEdit.Visible := true;

ConfirmForm.NameEdit.Text := '';

ConfirmForm.ConfirmBtn.Enabled := false;

ConfirmForm.ConfirmBtn.Text := 'Confirm';

ConfirmForm.ConfirmLbl.Text := 'Thank you for game! Enter your

name:';

**end**

**else**

**begin**

PlaySound('Audio\Death.wav', 0, SND\_ASYNC);

ConfirmForm.NameEdit.Visible := false;

ConfirmForm.ConfirmBtn.Enabled := true;

ConfirmForm.ConfirmBtn.Text := 'Ok';

ConfirmForm.ConfirmLbl.Text := 'Thank you for game! Try to beat the

highscore.';

**end**;

ConfirmForm.Showmodal;

Close;

**end**;

**procedure** TField.GameTimerTimer(Sender: TObject);

**begin**

Randomize;

GetPinkyMove;

GetClydeMove;

GetInkyMove;

GetBlinkyMove;

**if** ((PacMan.X = Inky.X) **and** (PacMan.Y = Inky.Y)) **or**

((PacMan.X = Pinky.X) **and** (PacMan.Y = Pinky.Y)) **or**

((PacMan.X = Clyde.X) **and** (PacMan.Y = Clyde.Y)) **or**

((PacMan.X = Blinky.X) **and** (PacMan.Y = Blinky.Y)) **then**

**begin**

Dec(PacMan.Lifes);

**case** PacMan.Lifes **of**

2:

**begin**

Life\_3.Visible := false;

Setup;

**end**;

1:

**begin**

Life\_2.Visible := false;

Setup;

**end**;

0: InitialiseResults;

**end**;

**end**;

**end**;

**procedure** TField.FormKeyDown(Sender: TObject; **var** Key: Word; **var** KeyChar:

Char; Shift: TShiftState);

**begin**

**if not**(PacXAnimation.Enabled **or** PacYAnimation.Enabled) **then**

**case** key **of**

VkUp: CheckMove(PacMan.Y - 1, PacMan.X);

VkDown: CheckMove(PacMan.Y + 1, PacMan.X);

VkLeft: CheckMove(PacMan.Y, PacMan.X - 1);

VkRight: CheckMove(PacMan.Y, PacMan.X + 1);

**end**;

**if not**(GameTimer.Enabled) **then**

GameTimer.Enabled := true;

**if not**(Clyde.InGame) **and** (StrToInt(ScoreValueLbl.Text) = 200) **then**

**begin**

Clyde.InGame := true;

Clyde.Y := 11;

Clyde.X := 14;

ClydeImg.Position.Y := 11 \* 20 + 50;

ClydeImg.Position.X := 14 \* 20;

**end**;

**if not**(Inky.InGame) **and** (StrToInt(ScoreValueLbl.Text) > 400) **then**

**begin**

Inky.InGame := true;

Inky.Y := 11;

Inky.X := 12;

InkyImg.Position.Y := 11 \* 20 + 50;

InkyImg.Position.X := 12 \* 20;

**end**;

**if not**(Blinky.InGame) **and** (StrToInt(ScoreValueLbl.Text) > 600) **then**

**begin**

BlinkyColdown.Enabled := true;

Blinky.InGame := true;

Blinky.Y := 11;

Blinky.X := 15;

BlinkyImg.Position.Y := 11 \* 20 + 50;

BlinkyImg.Position.X := 15 \* 20;

**end**;

**end**;

**procedure** TField.FormCreate(Sender: TObject);

**var**

i, j: ShortInt;

TempC: TCircle;

**begin**

**for** i := 0 **to** 30 **do**

**for** j := 0 **to** 27 **do**

**case** TilesRepresentation[i][j] **of**

0: **With** Tiles[i][j] **do**

**begin**

Wall := false;

Eaten := false;

Dot := TCircle.Create(Self);

Dot.Parent := Self;

Dot.Fill.color := $FFFFFF00;

Dot.Position.X := j \* 20 + 8;

Dot.Position.Y := i \* 20 + 58;

Dot.Width := 5;

Dot.Height := 5;

**end**;

1: Tiles[i][j].Wall := true;

**end**;

**end**;

**procedure** TField.FormCloseQuery(Sender: TObject; **var** CanClose: Boolean);

**begin**

PlaySound(NIL, 0, SND\_PURGE);

StartMenu.Show;

PlaySound('Audio\Menu.wav', 0, SND\_ASYNC **or** SND\_LOOP);

**end**;

**procedure** TField.SetImages;

**begin**

GameTimer.Enabled := false;

PinkyXAnimation.Enabled := false;

PinkyYAnimation.Enabled := false;

PacXAnimation.Enabled := false;

PacYAnimation.Enabled := false;

ClydeXAnimation.Enabled := false;

ClydeYAnimation.Enabled := false;

InkyXAnimation.Enabled := false;

InkyYAnimation.Enabled := false;

BlinkyXAnimation.Enabled := false;

BlinkyYAnimation.Enabled := false;

PacMan.Y := 23;

PacMan.X := 13;

Pac.Position.X := 260;

Pac.Position.Y := 510;

Pinky.Y := 11;

Pinky.X := 13;

PinkyImg.Position.X := 260;

PinkyImg.Position.Y := 270;

**end**;

**procedure** TField.Setup;

**var**

i, j: ShortInt;

TempC: TCircle;

**begin**

SetImages;

**if** StrToInt(ScoreValueLbl.Text) > 195 **then**

**begin**

Clyde.InGame := true;

Clyde.Y := 11;

Clyde.X := 14;

ClydeImg.Position.X := 14 \* 20;

ClydeImg.Position.Y := 11 \* 20 + 50;

**end**

**else**

**begin**

Clyde.InGame := false;

Clyde.Y := 0;

Clyde.X := 0;

ClydeImg.Position.X := 232;

ClydeImg.Position.Y := 332;

**end**;

**if** StrToInt(ScoreValueLbl.Text) > 395 **then**

**begin**

Inky.InGame := true;

Inky.Y := 11;

Inky.X := 12;

InkyImg.Position.X := 12 \* 20;

InkyImg.Position.Y := 11 \* 20 + 50;

**end**

**else**

**begin**

Inky.InGame := false;

Inky.Y := 0;

Inky.X := 0;

InkyImg.Position.X := 270;

InkyImg.Position.Y := 332;

**end**;

**if** StrToInt(ScoreValueLbl.Text) > 595 **then**

**begin**

Blinky.InGame := true;

Blinky.Y := 11;

Blinky.X := 15;

BlinkyImg.Position.X := 15 \* 20;

BlinkyImg.Position.Y := 11 \* 20 + 50;

BlinkyColdown.Enabled := true;

**end**

**else**

**begin**

Blinky.InGame := false;

Blinky.Y := 0;

Blinky.X := 0;

BlinkyImg.Position.X := 308;

BlinkyImg.Position.Y := 332;

BlinkyColdown.Enabled := false;

**end**;

**end**;

**procedure** TField.Move(NewY, NewX: ShortInt);

**begin**

**if** (NewY <> PacMan.Y) **then**

**begin**

PacYAnimation.StartValue := PacMan.Y \* 20 + 50;

PacYAnimation.StopValue := NewY \* 20 + 50;

PacYAnimation.Enabled := true;

**end**

**else**

**begin**

PacXAnimation.StartValue := PacMan.X \* 20;

PacXAnimation.StopValue := NewX \* 20;

PacXAnimation.Enabled := true;

**end**;

**end**;

**procedure** TField.PacXAnimationFinish(Sender: TObject);

**begin**

PacXAnimation.Enabled := false;

Pac.Position.X := PacXAnimation.StopValue;

PacMan.X := Trunc(PacXAnimation.StopValue / 20);

**end**;

**procedure** TField.PacYAnimationFinish(Sender: TObject);

**begin**

PacYAnimation.Enabled := false;

Pac.Position.Y := PacYAnimation.StopValue;

PacMan.Y := Trunc((PacYAnimation.StopValue - 50) / 20);

**end**;

**procedure** TField.CheckMove(NewY, NewX: ShortInt);

**begin**

**if** (NewY > PacMan.Y) **then**

Pac.RotationAngle := 90;

**if** (NewY < PacMan.Y) **then**

Pac.RotationAngle := -90;

**if** (NewX > PacMan.X) **then**

Pac.RotationAngle := 0;

**if** (NewX < PacMan.X) **then**

Pac.RotationAngle := 180;

**if** (NewX = -1) **or** (NewX = 28) **then**

**if** (NewX = -1) **then**

**begin**

PacMan.X := 28;

Pac.Position.X := 28 \* 20;

Move(NewY, 27);

**if not**(Tiles[NewY, 27].Eaten) **then**

**begin**

PlaySound('Audio\Eating.wav', 0, SND\_ASYNC **or** SND\_NOSTOP);

ScoreValueLbl.Text := IntToStr(StrToInt(ScoreValueLbl.Text) +

5);

Tiles[NewY][27].Dot.Visible := false;

Tiles[NewY, 27].Eaten := true;

**end**;

**end**

**else**

**begin**

PacMan.X := -1;

Pac.Position.X := -1 \* 20;

Move(NewY, 0);

**if not**(Tiles[NewY, 0].Eaten) **then**

**begin**

PlaySound('Audio\Eating.wav', 0, SND\_ASYNC **or** SND\_NOSTOP);

ScoreValueLbl.Text := IntToStr(StrToInt(ScoreValueLbl.Text) +

5);

Tiles[NewY][0].Dot.Visible := false;

Tiles[NewY, 0].Eaten := true;

**end**;

**end**

**else**

**if not**(Tiles[NewY, NewX].Wall) **then**

**if not**(Tiles[NewY, NewX].Eaten) **then**

**begin**

Move(NewY, NewX);

PlaySound('Audio\Eating.wav', 0, SND\_ASYNC **or** SND\_NOSTOP);

ScoreValueLbl.Text := IntToStr(StrToInt(ScoreValueLbl.Text) +

5);

Tiles[NewY][NewX].Dot.Visible := false;

Tiles[NewY, NewX].Eaten := true;

**end**

**else**

Move(NewY, NewX);

**end**;

**procedure** TField.CheckPinckyMove(NewY, NewX: ShortInt);

**begin**

**if** ((NewX = 5) **or** (NewX = 22)) **and** (NewY = 14) **then**

**if** (NewX = 5) **then**

**begin**

PinkyXAnimation.Duration := PinkySpeed \* 6;

PinkyXAnimation.StartValue := Pinky.X \* 20;

PinkyXAnimation.StopValue := 0;

PinkyXAnimation.Enabled := true;

**end**

**else**

**begin**

PinkyXAnimation.Duration := PinkySpeed \* 6;

PinkyXAnimation.StartValue := Pinky.X \* 20;

PinkyXAnimation.StopValue := 27 \* 20;

PinkyXAnimation.Enabled := true;

**end**

**else**

**if not**(Tiles[NewY, NewX].Wall) **then**

PinkyMove(NewY, NewX);

**end**;

**procedure** TField.PinkyMove(NewY, NewX: ShortInt);

**begin**

**if** (NewY <> Pinky.Y) **then**

**begin**

**if** (NewY > Pinky.Y) **then**

**while** (Tiles[NewY][NewX + 1].Wall **and** Tiles[NewY][NewX - 1].Wall) **do**

**begin**

Inc(NewY);

PinkyYAnimation.Duration := PinkyYAnimation.Duration +

PinkySpeed

**end**

**else**

**while** (Tiles[NewY][NewX + 1].Wall **and** Tiles[NewY][NewX - 1].Wall) **do**

**begin**

Dec(NewY);

PinkyYAnimation.Duration := PinkyYAnimation.Duration +

PinkySpeed

**end**;

PinkyYAnimation.StartValue := Pinky.Y \* 20 + 50;

PinkyYAnimation.StopValue := NewY \* 20 + 50;

PinkyYAnimation.Enabled := true;

**end**

**else**

**begin**

**if** (NewX > Pinky.X) **then**

**while** (Tiles[NewY + 1][NewX].Wall **and** Tiles[NewY - 1][NewX].Wall) **do**

**begin**

Inc(NewX);

PinkyXAnimation.Duration := PinkyXAnimation.Duration +

PinkySpeed;

**end**

**else**

**while** (Tiles[NewY + 1][NewX].Wall **and** Tiles[NewY - 1][NewX].Wall) **do**

**begin**

Dec(NewX);

PinkyXAnimation.Duration := PinkyXAnimation.Duration +

PinkySpeed;

**end**;

PinkyXAnimation.StartValue := Pinky.X \* 20;

PinkyXAnimation.StopValue := NewX \* 20;

PinkyXAnimation.Enabled := true;

**end**;

**end**;

**procedure** TField.PinkyXAnimaionFinish(Sender: TObject);

**begin**

PinkyXAnimation.Enabled := false;

PinkyXAnimation.Duration := PinkySpeed;

**if** PinkyXAnimation.StopValue = 0 **then**

**begin**

Pinky.X := 27;

PinkyImg.Position.X := 27 \* 20;

PinkyMove(14, 26);

**end**;

**if** PinkyXAnimation.StopValue = 27 \* 20 **then**

**begin**

Pinky.X := 0;

PinkyImg.Position.X := 0;

PinkyMove(14, 1);

**end**;

**if** PinkyXAnimation.StopValue > PinkyXAnimation.StartValue **then**

Pinky.LastMove := 3

**else**

Pinky.LastMove := 2;

PinkyImg.Position.X := PinkyXAnimation.StopValue;

Pinky.X := Trunc(PinkyXAnimation.StopValue / 20);

**end**;

**procedure** TField.PinkyXAnimationProcess(Sender: TObject);

**begin**

Pinky.X := Trunc(PinkyImg.Position.X / 20);

**end**;

**procedure** TField.PinkyYAnmationFinish(Sender: TObject);

**begin**

PinkyYAnimation.Enabled := false;

**if** PinkyYAnimation.StopValue > PinkyYAnimation.StartValue **then**

Pinky.LastMove := 1

**else**

Pinky.LastMove := 0;

PinkyImg.Position.Y := PinkyYAnimation.StopValue;

Pinky.Y := Trunc((PinkyYAnimation.StopValue - 50) / 20);

PinkyYAnimation.Duration := PinkySpeed;

**end**;

**procedure** TField.PinkyYAnimationProcess(Sender: TObject);

**begin**

Pinky.Y := Trunc((PinkyImg.Position.Y - 50) / 20);

**end**;

**procedure** TField.CheckClydeMove(NewY, NewX: ShortInt);

**begin**

**if not**(Tiles[NewY, NewX].Wall) **then**

ClydeMove(NewY, NewX);

**end**;

**procedure** TField.ClydeMove(NewY, NewX: ShortInt);

**begin**

**if** (NewY <> Clyde.Y) **then**

**begin**

**if** (NewY > Clyde.Y) **then**

**while** (Tiles[NewY][NewX + 1].Wall **and** Tiles[NewY][NewX - 1].Wall) **do**

**begin**

Inc(NewY);

ClydeYAnimation.Duration := ClydeYAnimation.Duration +

ClydeSpeed;

**end**

**else**

**while** (Tiles[NewY][NewX + 1].Wall **and** Tiles[NewY][NewX - 1].Wall) **do**

**begin**

Dec(NewY);

ClydeYAnimation.Duration := ClydeYAnimation.Duration +

ClydeSpeed;

**end**;

ClydeYAnimation.StartValue := Clyde.Y \* 20 + 50;

ClydeYAnimation.StopValue := NewY \* 20 + 50;

ClydeYAnimation.Enabled := true;

**end**

**else**

**begin**

**if** (NewX > Clyde.X) **then**

**while** (Tiles[NewY + 1][NewX].Wall **and** Tiles[NewY - 1][NewX].Wall)

**do**

**begin**

Inc(NewX);

ClydeXAnimation.Duration := ClydeXAnimation.Duration +

ClydeSpeed;

**end**

**else**

**while** (Tiles[NewY + 1][NewX].Wall **and** Tiles[NewY - 1][NewX].Wall)

**do**

**begin**

Dec(NewX);

ClydeXAnimation.Duration := ClydeXAnimation.Duration +

ClydeSpeed;

**end**;

ClydeXAnimation.StartValue := Clyde.X \* 20;

ClydeXAnimation.StopValue := NewX \* 20;

ClydeXAnimation.Enabled := true;

**end**;

**end**;

**procedure** TField.ClydeXAnimationFinish(Sender: TObject);

**begin**

ClydeXAnimation.Enabled := false;

ClydeXAnimation.Duration := ClydeSpeed;

ClydeImg.Position.X := ClydeXAnimation.StopValue;

Clyde.X := Trunc(ClydeXAnimation.StopValue / 20);

**end**;

**procedure** TField.ClydeXAnimationProcess(Sender: TObject);

**begin**

Clyde.X := Trunc(ClydeImg.Position.X / 20);

**end**;

**procedure** TField.ClydeYAnimationFinish(Sender: TObject);

**begin**

ClydeYAnimation.Enabled := false;

ClydeYAnimation.Duration := ClydeSpeed;

ClydeImg.Position.Y := ClydeYAnimation.StopValue;

Clyde.Y := Trunc((ClydeYAnimation.StopValue - 50) / 20);

**end**;

**procedure** TField.ClydeYAnimationProcess(Sender: TObject);

**begin**

Clyde.Y := Trunc((ClydeImg.Position.Y - 50) / 20);

**end**;

**procedure** TField.CheckInkyMove(NewY, NewX: ShortInt);

**begin**

**if not**(Tiles[NewY, NewX].Wall) **then**

InkyMove(NewY, NewX);

**end**;

**procedure** TField.InkyMove(NewY, NewX: ShortInt);

**begin**

**if** (NewY <> Inky.Y) **then**

**begin**

**if** (NewY > Inky.Y) **then**

**while** (Tiles[NewY][NewX + 1].Wall **and** Tiles[NewY][NewX - 1].Wall)

**do**

**begin**

Inc(NewY);

InkyYAnimation.Duration := InkyYAnimation.Duration +

InkySpeed;

**end**

**else**

**while** (Tiles[NewY][NewX + 1].Wall **and** Tiles[NewY][NewX - 1].Wall)

**do**

**begin**

Dec(NewY);

InkyYAnimation.Duration := InkyYAnimation.Duration +

InkySpeed;

**end**;

InkyYAnimation.StartValue := Inky.Y \* 20 + 50;

InkyYAnimation.StopValue := NewY \* 20 + 50;

InkyYAnimation.Enabled := true;

**end**

**else**

**begin**

**if** (NewX > Inky.X) **then**

**while** (Tiles[NewY + 1][NewX].Wall **and** Tiles[NewY - 1][NewX].Wall)

**do**

**begin**

Inc(NewX);

InkyXAnimation.Duration := InkyXAnimation.Duration +

InkySpeed;

**end**

**else**

**while** (Tiles[NewY + 1][NewX].Wall **and** Tiles[NewY - 1][NewX].Wall)

**do**

**begin**

Dec(NewX);

InkyXAnimation.Duration := InkyXAnimation.Duration +

InkySpeed;

**end**;

InkyXAnimation.StartValue := Inky.X \* 20;

InkyXAnimation.StopValue := NewX \* 20;

InkyXAnimation.Enabled := true;

**end**;

**end**;

**procedure** TField.InkyXAnimationFinish(Sender: TObject);

**begin**

InkyXAnimation.Enabled := false;

InkyXAnimation.Duration := InkySpeed;

InkyImg.Position.X := InkyXAnimation.StopValue;

Inky.X := Trunc(InkyXAnimation.StopValue / 20);

**end**;

**procedure** TField.InkyXAnimationProcess(Sender: TObject);

**begin**

Inky.X := Trunc(InkyImg.Position.X / 20);

**end**;

**procedure** TField.InkyYAnimationFinish(Sender: TObject);

**begin**

InkyYAnimation.Enabled := false;

InkyYAnimation.Duration := InkySpeed;

InkyImg.Position.Y := InkyYAnimation.StopValue;

Inky.Y := Trunc((InkyYAnimation.StopValue - 50) / 20);

**end**;

**procedure** TField.InkyYAnimationProcess(Sender: TObject);

**begin**

Inky.Y := Trunc((InkyImg.Position.Y - 50) / 20);

**end**;

**procedure** TField.CheckBlinkyMove(NewY, NewX: ShortInt);

**begin**

**if** ((NewX = 5) **or** (NewX = 22)) **and** (NewY = 14) **then**

**if** (NewX = 5) **then**

**begin**

BlinkyXAnimation.Duration := BlinkySpeed \* 6;

BlinkyXAnimation.StartValue := Blinky.X \* 20;

BlinkyXAnimation.StopValue := 0;

BlinkyXAnimation.Enabled := true;

**end**

**else**

**begin**

BlinkyXAnimation.Duration := BlinkySpeed \* 6;

BlinkyXAnimation.StartValue := Blinky.X \* 20;

BlinkyXAnimation.StopValue := 27 \* 20;

BlinkyXAnimation.Enabled := true;

**end**

**else**

**if not**(Tiles[NewY, NewX].Wall) **then**

BlinkyMove(NewY, NewX);

**end**;

**procedure** TField.BlinkyColdownTimer(Sender: TObject);

**var**

NewX, NewY: ShortInt;

**begin**

BlinkyXAnimation.StopAtCurrent;

BlinkyYAnimation.StopAtCurrent;

**case** random(4) **of**

0:

**begin**

NewY := PacMan.Y - 3;

NewX := PacMan.X;

**end**;

1:

**begin**

NewY := PacMan.Y + 3;

NewX := PacMan.X;

**end**;

2:

**begin**

NewY := PacMan.Y;

NewX := PacMan.X - 3;

**end**;

3:

**begin**

NewY := PacMan.Y;

NewX := PacMan.X + 3;

**end**;

**end**;

**if** (NewX < 27) **and** (NewX > 0) **and** (NewY < 30) **and** (NewY > 0) **and**

**not**(Tiles[NewY][NewX].Wall) **then**

**begin**

Blinky.X := NewX;

Blinky.Y := NewY;

BlinkyImg.Position.Y := NewY \* 20 + 50;

BlinkyImg.Position.X := NewX \* 20;

**end**;

**end**;

**procedure** TField.BlinkyMove(NewY, NewX: ShortInt);

**begin**

**if** (NewY <> Blinky.Y) **then**

**begin**

**if** (NewY > Blinky.Y) **then**

**while** (Tiles[NewY][NewX + 1].Wall **and** Tiles[NewY][NewX - 1].Wall)

**do**

**begin**

Inc(NewY);

BlinkyYAnimation.Duration := BlinkyYAnimation.Duration +

BlinkySpeed;

**end**

**else**

**while** (Tiles[NewY][NewX + 1].Wall **and** Tiles[NewY][NewX - 1].Wall)

**do**

**begin**

Dec(NewY);

BlinkyYAnimation.Duration := BlinkyYAnimation.Duration +

BlinkySpeed;

**end**;

BlinkyYAnimation.StartValue := Blinky.Y \* 20 + 50;

BlinkyYAnimation.StopValue := NewY \* 20 + 50;

BlinkyYAnimation.Enabled := true;

**end**

**else**

**begin**

**if** (NewX > Blinky.X) **then**

**while** (Tiles[NewY + 1][NewX].Wall **and** Tiles[NewY - 1][NewX].Wall)

**do**

**begin**

Inc(NewX);

BlinkyXAnimation.Duration := BlinkyXAnimation.Duration +

BlinkySpeed;

**end**

**else**

**while** (Tiles[NewY + 1][NewX].Wall **and** Tiles[NewY - 1][NewX].Wall)

**do**

**begin**

Dec(NewX);

BlinkyXAnimation.Duration := BlinkyXAnimation.Duration +

BlinkySpeed;

**end**;

BlinkyXAnimation.StartValue := Blinky.X \* 20;

BlinkyXAnimation.StopValue := NewX \* 20;

BlinkyXAnimation.Enabled := true;

**end**;

**end**;

**procedure** TField.BlinkyXAnimationFinish(Sender: TObject);

**begin**

BlinkyXAnimation.Enabled := false;

BlinkyXAnimation.Duration := BlinkySpeed;

**if** BlinkyXAnimation.StopValue = 0 **then**

**begin**

Blinky.X := 27;

BlinkyImg.Position.X := 27 \* 20;

BlinkyMove(14, 26);

**end**;

**if** BlinkyXAnimation.StopValue = 27 \* 20 **then**

**begin**

Blinky.X := 0;

BlinkyImg.Position.X := 0;

BlinkyMove(14, 1);

**end**;

**if** BlinkyXAnimation.StopValue > BlinkyXAnimation.StartValue **then**

Blinky.LastMove := 3

**else**

Blinky.LastMove := 2;

BlinkyImg.Position.X := BlinkyXAnimation.StopValue;

Blinky.X := Trunc(BlinkyXAnimation.StopValue / 20);

**end**;

**procedure** TField.BlinkyXAnimationProcess(Sender: TObject);

**begin**

Blinky.X := Trunc(BlinkyImg.Position.X / 20);

**end**;

**procedure** TField.BlinkyYAnimationFinish(Sender: TObject);

**begin**

BlinkyYAnimation.Enabled := false;

**if** BlinkyYAnimation.StopValue > BlinkyYAnimation.StartValue **then**

Blinky.LastMove := 1

**else**

Blinky.LastMove := 0;

BlinkyImg.Position.Y := BlinkyYAnimation.StopValue;

Blinky.Y := Trunc((BlinkyYAnimation.StopValue - 50) / 20);

BlinkyYAnimation.Duration := BlinkySpeed;

**end**;

**procedure** TField.BlinkyYAnimationProcess(Sender: TObject);

**begin**

Blinky.Y := Trunc((BlinkyImg.Position.Y - 50) / 20);

**end**;

**end**.

**ConfirmUnit.pas**

**unit** ConfirmUnit;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants, FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics,

FMX.StdCtrls, FMX.Controls.Presentation, FMX.Edit, RecordsUnit;

**type**

TConfirmForm = **class**(TForm)

NameEdit: TEdit;

ConfirmLbl: TLabel;

ConfirmBtn: TButton;

**procedure** ConfirmBtnClick(Sender: TObject);

**procedure** NameEditKeyUp(Sender: TObject; **var** Key: Word; **var** KeyChar:

Char; Shift: TShiftState);

**procedure** NameEditKeyDown(Sender: TObject; **var** Key: Word; **var**

KeyChar: Char; Shift: TShiftState);

**end**;

**var**

ConfirmForm: TConfirmForm;

**implementation**

{$R \*.fmx}

**uses** GameUnit;

**procedure** TConfirmForm.ConfirmBtnClick(Sender: TObject);

**var**

i: Byte;

**begin**

**if** ConfirmBtn.Text = 'Confirm' **then**

**With** RecordsForm **do**

**begin**

**for** i := 3 **Downto** 0 **do**

RecordsDataBase[i + 1] := RecordsDataBase[i];

RecordsDataBase[0].Name := NameEdit.Text;

RecordsDataBase[0].Score := Field.ScoreValueLbl.Text;

Rewrite(RecordsFile);

**for** i := 0 **to** 4 **do**

**if not**(RecordsDataBase[i].Name = '...........') **then**

Write(RecordsFile, RecordsDataBase[i]);

CloseFile(RecordsFile);

**end**;

Close;

**end**;

**procedure** TConfirmForm.NameEditKeyDown(Sender: TObject; **var** Key: Word;

**var** KeyChar: Char; Shift: TShiftState);

**begin**

**if** Length(NameEdit.Text) > 11 **then**

KeyChar := #0;

**end**;

**procedure** TConfirmForm.NameEditKeyUp(Sender: TObject; **var** Key: Word;

**var** KeyChar: Char; Shift: TShiftState);

**begin**

**if** Length(NameEdit.Text) > 0 **then**

ConfirmBtn.Enabled := true

**else**

ConfirmBtn.Enabled := false;

**end**;

**end**.

**RecordsUnit.pas**

**unit** RecordsUnit;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants, FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics,

FMX.Dialogs, FMX.StdCtrls, FMX.Controls.Presentation, System.Rtti,

FMX.Grid.Style, FMX.ScrollBox, Mmsystem, FMX.Grid, LinkedList;

**type**

TPlayer = **record**

Name: String[11];

Score: String[4];

**end**;

TRecordsForm = **class**(TForm)

BackBtn: TButton;

Label6: TLabel;

PaceLbl: TLabel;

ScoreValueLbl: TLabel;

PlayerNameLbl: TLabel;

**procedure** BackBtnClick(Sender: TObject);

**procedure** FormCloseQuery(Sender: TObject; **var** CanClose: Boolean);

**procedure** FormCreate(Sender: TObject);

**procedure** FormShow(Sender: TObject);

**public**

RecordsFile: **File of** TPlayer;

RecordsDataBase: **array** [0..4] **of** TPlayer;

**end**;

**var**

RecordsForm: TRecordsForm;

**implementation**

{$R \*.fmx}

**uses** MainMenuUnit;

**procedure** TRecordsForm.BackBtnClick(Sender: TObject);

**begin**

RecordsForm.Close;

**end**;

**procedure** TRecordsForm.FormCloseQuery(Sender: TObject; **var** CanClose:

Boolean);

**begin**

StartMenu.Show;

**end**;

**procedure** TRecordsForm.FormCreate(Sender: TObject);

**var**

i: Byte;

Temp: TPlayer;

**begin**

AssignFile(RecordsFile, 'DontTouch.pcm');

**try**

Reset(RecordsFile);

**except**

Rewrite(RecordsFile);

**end**;

Temp.Name := '...........';

Temp.Score := '0000';

i := 0;

**for** i := 0 **to** 4 **do**

**if not**(Eof(RecordsFile)) **then**

Read(RecordsFile, RecordsDataBase[i])

**else**

RecordsDataBase[i] := Temp;

CloseFile(RecordsFile);

**end**;

**procedure** TRecordsForm.FormShow(Sender: TObject);

**var**

i: ShortInt;

Temp: TPlayer;

**begin**

PlayerNameLbl.Text := '';

ScoreValueLbl.Text := '';

**for** i := 0 **to** 4 **do**

**begin**

PlayerNameLbl.Text := PlayerNameLbl.Text + RecordsDataBase[i].Name +

#13#10;

ScoreValueLbl.Text := ScoreValueLbl.Text + RecordsDataBase[i].Score +

#13#10;

**end**;

**end**;

**end**.

**AboutUnit.pas**

**unit** AboutUnit;

**interface**

**uses**

System.SysUtils, System.Types, System.UITypes, System.Classes,

System.Variants, FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics,

FMX.Dialogs, FMX.Controls.Presentation, FMX.StdCtrls, FMX.Ani,

FMX.Objects;

**type**

TAboutForm = **class**(TForm)

RulesLbl: TLabel;

BackBtn: TButton;

PinkyLbl: TLabel;

InkyLbl: TLabel;

BlinkyLbl: TLabel;

ClydeImg: TImage;

ClydeAnimation: TBitmapListAnimation;

PinkyImg: TImage;

PinkyAnimation: TBitmapListAnimation;

InkyImg: TImage;

InkyAnimation: TBitmapListAnimation;

BlinkyImg: TImage;

BlinkyAnimation: TBitmapListAnimation;

Pac: TImage;

PacAnimation: TBitmapListAnimation;

ClydeLbl: TLabel;

DeveloperLbl: TLabel;

RulesLogoLbl: TLabel;

**procedure** BackBtnClick(Sender: TObject);

**procedure** FormCloseQuery(Sender: TObject; **var** CanClose: Boolean);

**end**;

**var**

AboutForm: TAboutForm;

**implementation**

{$R \*.fmx}

**uses** MainMenuUnit;

**procedure** TAboutForm.BackBtnClick(Sender: TObject);

**begin**

Close;

**end**;

**procedure** TAboutForm.FormCloseQuery(Sender: TObject; **var** CanClose: Boolean);

**begin**

StartMenu.Show;

**end**;

**end**.