Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**ПРОГРАММНОЕ СРЕДСТВО**

**«Space Invaders»**

БГУИР КП 1-40 01 01 003 ПЗ

Студент гр. 351004 Бражалович А.И.

Руководитель Данилова Г.В.

Минск 2024

СОДЕРЖАНИЕ

[Введение 5](#_Toc167417859)

[1 Анализ предметной области 7](#_Toc167417860)

[1.1 Обзор аналогов 7](#_Toc167417861)

[1.2 Постановка задачи 9](#_Toc167417862)

[2 Проектирование программного средства 10](#_Toc167417863)

[2.1 Структура программы 10](#_Toc167417864)

[2.2 Проектирование интерфейса программного средства 10](#_Toc167417865)

[2.3 Проектирование функционала программного средства 14](#_Toc167417866)

[3 Разработка программного средства 18](#_Toc167417867)

[3.1 Прорисовка материалов для игры 18](#_Toc167417868)

[3.2 Игровой процесс 20](#_Toc167417869)

[3.3 Работа со списком игроков 22](#_Toc167417874)

[4 Тестирование программного средства 24](#_Toc167417875)

[5 Руководство пользователя 25](#_Toc167417876)

[5.1 Интерфейс программного средства 25](#_Toc167417877)

[5.2 Управление программным средством 28](#_Toc167417878)

[Заключение 30](#_Toc167417879)

[Список использованных источников 31](#_Toc167417880)

[Приложение А. Текст программы 32](#_Toc167417881)

# ВВЕДЕНИЕ

В последние десятилетия сфера информационных технологий претерпела множество изменений, и на сегодняшний день перед специалистами стоят задачи самого разного спектра: от разработки сложного технического обеспечения для крупных производственных организаций до создания простых аркадных мобильных приложений для развлечения среднестатистического пользователя. Именно в последней категории программных продуктов начала развиваться целая индустрия компьютерных игр. Сегодня рынок переполнен продуктами разного качества и жанров: платформеры, шутеры, приключения и головоломки.

Своему появлению компьютерные игры обязаны таким пионерам, как Ральф Бауэр (инженер, выдвинувший идею интерактивного телевидения в 1951 году), Александр Дуглас (разработчик «ОХО» – компьютерной реализации крестиков-ноликов как примера для диссертации на тему взаимодействия человека и компьютера в 1952 году) и Уильям Хигинботам (создатель первой многопользовательской игры «Tennis for Two» в 1958 году). Однако широкое распространение компьютерные игры получили только с выпуском первого компьютера серии PDP. Он получил название PDP-1, и только спустя два года, в 1962 году, для него была разработана первая компьютерная игра – «SpaceWar!».

Прообразами современных компьютерных игр всегда являлись вещи или события, уже существующие и перенесенные в цифровой формат. Так, жанр MMORPG стал продолжением настольной игры «Dungeons & Dragons», а аркадные шутеры, такие как «Space Invaders», были вдохновлены концепцией борьбы с нападающими волнами врагов, часто заимствованной из научно-фантастических фильмов и книг.

История создания и развития «Space Invaders» берет начало в 1978 году, когда японская компания Taito выпустила эту игру. Разработанная Томохиро Нисикадо, она быстро завоевала популярность благодаря своим простым, но увлекательным игровым механикам и незабываемому звуковому сопровождению. «Space Invaders» стала не просто игрой, а культурным феноменом, определившим направление развития аркадных игр. Основная идея заключалась в том, чтобы игрок отбивал волны инопланетных захватчиков, постепенно увеличивая сложность и напряжение.

Игра «Space Invaders» стала новаторской в своем роде. Она установила стандарт для аркадных шутеров, предложив игрокам постепенное увеличение сложности и требуя от них стратегического мышления и быстроты реакции. Простые, но затягивающие механики сделали игру популярной среди широкого круга игроков, от детей до взрослых, и привлекли внимание общественности к видеоиграм как к форме развлечения.

Почему сегодня «Space Invaders» остается популярной в этом современном, быстро развивающемся мире? Одна из возможных причин заключается в том, что эта игра обращается к базовому человеческому инстинкту соревновательности и стремления к совершенству. Есть что-то очень удовлетворяющее в уничтожении врагов и достижении новых уровней. Другая причина в том, что правила игры очень просты и понятны. Люди всех возрастов могут быстро научиться игре и начать наслаждаться захватывающим процессом. Глобальная популярность «Space Invaders» показывает, что игра легко понятна и привлекательна для людей по всему миру, так как основывается на простых и интуитивно понятных механиках.

«Space Invaders» также внесла значительный вклад в развитие игровой культуры и индустрии. Она стала вдохновением для множества последующих игр и способствовала развитию технологий и дизайнерских подходов в игровой индустрии. Многие современные шутеры и аркадные игры черпают вдохновение из оригинальных механик и концепций, заложенных в «Space Invaders».

Таким образом, целью данного проекта является создание игрового приложения, воссоздающего классический геймплей «Space Invaders» и адаптирующего его для современных пользователей. Этот проект направлен на оживление ностальгических воспоминаний о золотой эре аркадных игр, а также на демонстрацию того, как принципы и механики классики могут быть успешно адаптированы с использованием современных технологий и подходов в разработке игр.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Обзор аналогов

На данный момент существует большое множество вариаций игры «Space Invaders». В каждой вариации этой игры есть разные режимы и модификации.

Первой и самой популярной редакцией этой игры стала версия, выпущенная для игровых автоматов в 1978 году в Японии. Позже ее портировали на персональные компьютеры.

В портированных версиях сохранились дизайн и звуковое сопровождение. Также добавили кооперативный режим и новые механики атак.

Внешний вид данного приложения представлен на рисунке 1.1.

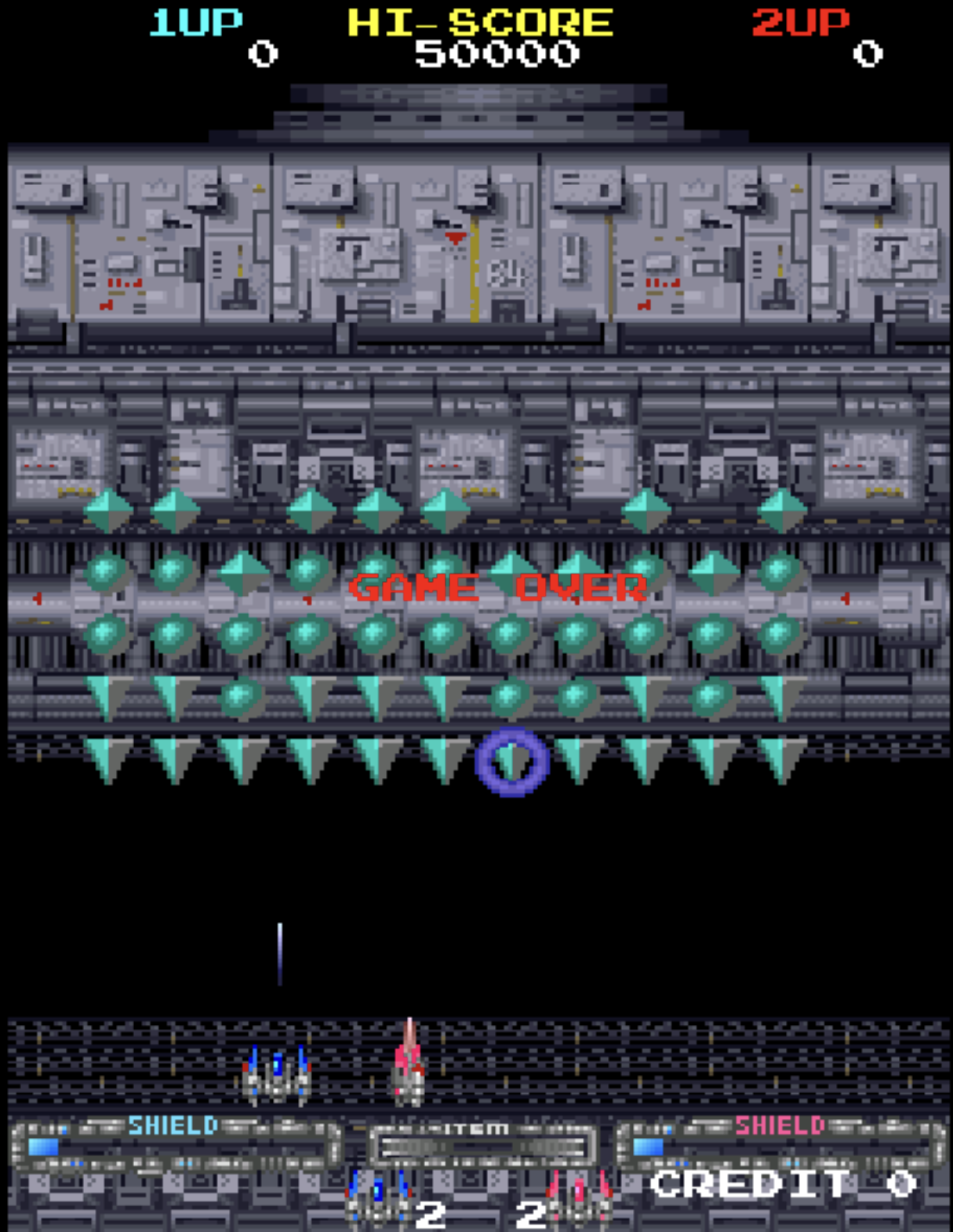


Рисунок 1.1 – Программное средство «Space Invaders Part IV»

Далее стоит рассмотреть реализацию «Space Invaders» от Google. В этой версии игры обычные барьеры заменены словом Google. Противники совершают большее количество атак. Также в этой версии есть обучение и несколько вариантов управления главной пушкой.

Внешний вид окна данного приложения представлен на рисунке 1.2.



Рисунок 1.2 – Программное средство «Space Invaders»

Еще более простую версию игры «Space Invades» от разработчика MaxMouse можно встретить в расширениях для браузера Chrome. Инопланетяне, как и сама главная пушка заменены на прямоугольники. Из недостатков можно заметить отсутствие специльных стен, как в оригинальной игре. На рисунке 1.3 представлен интерфейс игрового приложения.

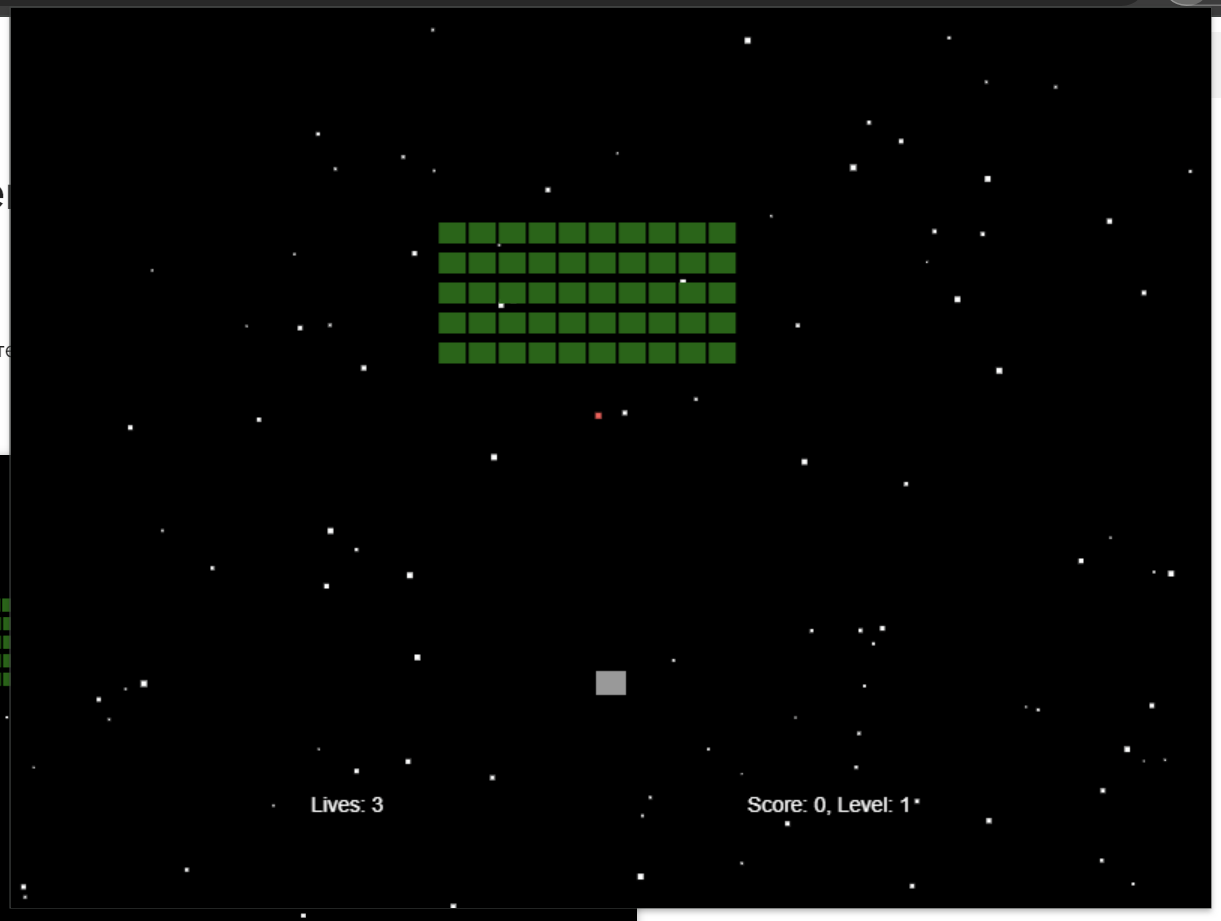


Рисунок 1.3 – Программное средство «Simple Space Invaders»

В рассмотренных аналогах можно увидеть разные механики игры, реализованные в разных стилях. Многие аналоги полностью повторяют оригинальное издание.

## Постановка задачи

В рамках данной курсовой работы планируется разработать игру «Space Invaders» для платформы Windows. После рассмотрения аналогов выявились некоторые недостатки, которые можно учесть в данной курсовой работе. На основе аналогов возможно добавление новых функций, не имеющихся в рассмотренных аналогах. Добавление новых функций отличных от похожих проектов привнесет разнообразие в игровой процесс. В процессе разработки должны быть реализованы базовые функции игры:

* выстрелы пушки;
* движение главной пушки и врагов;
* создание таблицы с рейтингом игроков.

Планируется работа с внешними устройствами:

* сохранение списка игроков в файл;
* добавление новых игроков в файл;
* удаление имеющихся игроков из файла;
* запуск списка игроков из файла.

Планируется добавление такого функционала, как:

* подсчет текущего числа очков;
* выстрелы врагов в главную пушку;
* механику жизней главной пушки.

Для разработки программного средства будет использоваться язык программирования Delphi и среда разработки Embarcadero Delphi 11 Community Edition.

# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## 2.1 Структура программы

При разработке приложения будет использовано одиннадцать модулей:

* GameFormUnit – модуль, в котором происходит процесс игры;
* GameOverUnit – модуль, обеспечивающий отрисовку окна окончания игры;
* ImageUnit – модуль, обеспечивающий удобную загрузку изображений;
* PlayersListUnit – модуль, содержащий реализацию однонаправленного списка игроков;
* StartMenuUnit – модуль, обеспечивающий отображение главного окна приложения;
* RocketUnit – модуль, содержащий реализацию логики работы пушки;
* PlayersFormUnit – модуль, обеспечивающий отображение окна взаимодействия со списком игроков;
* PauseUnit – модуль, обеспечивающий отображение окна паузы приложения;
* AddNewPlayerUnit – модуль, обеспечивающий отображение окна добавления нового игрока;
* FileUnit – модуль, обеспечивающий корректную работу с типизированными файлами;
* ScoreBoardUnit – модуль, обеспечивающий отрисоку таблицы рекордов игроков.

## 2.2 Проектирование интерфейса программного средства

При разработке программного средства за основу будет взят дизайн программного средства «Space Invaders».

2.2.1 Главное меню

Главное меню приложения будет состоять из четырех основных кнопок:

* первой должна располагаться StartButton, по нажатию которой пользователь переходит в модуль GameFormUnit;
* второй расположится кнопка PlayerButton, позволяющая управлять списком игроков;
* третьей будет ScoreListButton, нажав на которую, пользователь может просмотреть список рекордов игроков;
* четвертой будет кнопка ExitButton, которая позволяет покинуть программное средство.

Макет главного меню приложения представлен на рисунке 2.1.





Рисунок 2.1 – Макет главного меню приложения

В главном меню будет предусмотрена возможность вызова контекстного меню с информацией о разработчике.

2.2.2 Окно «Player»

Окно игроков будет содержать управление списком игроков. Кнопки в данном окне будут реализованы с помощью TImage. Для отображения игроков палнируется использовать TComboBox. Макет окна игроков представлен на рисунке 2.2.

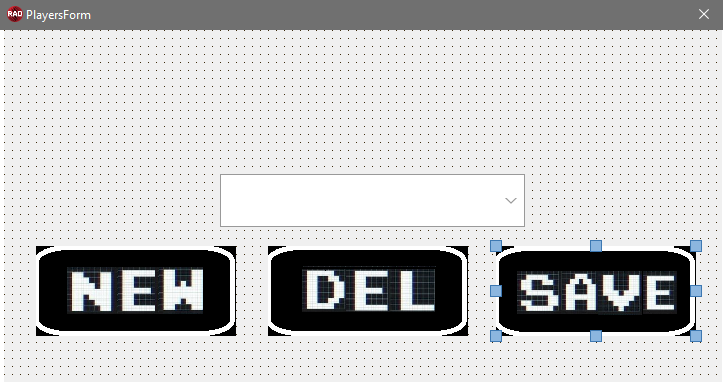


Рисунок 2.2 – Макет окна игроков

В окне игроков пользователю будет доступна возможность добавлять новых игроков и удалять уже имеющихся.

2.2.3 Окно добавления нового игрока

Окно добавления нового игрока будет содержать кнопку добавления и поле ввода имени игрока. Макет окна добавления нового игрока представлен на рисунке 2.3.

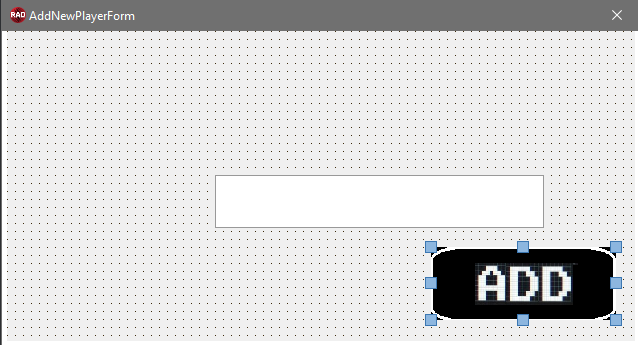


Рисунок 2.3 – Макет окна добавления нового игрока

Кнопка в окне добавления игрока будет реализована с помощью компонента TImage.

2.2.4 Окно паузы

Окно паузы должно состоять из двух основных кнопок:

* первой должна располагаться кнопка, по нажатию которой пользователь возвращается к игре;
* второй будет кнопка «Exit», нажав на которую пользователь вернется в главное меню.

Макет окна паузы представлен на рисунке 2.4.

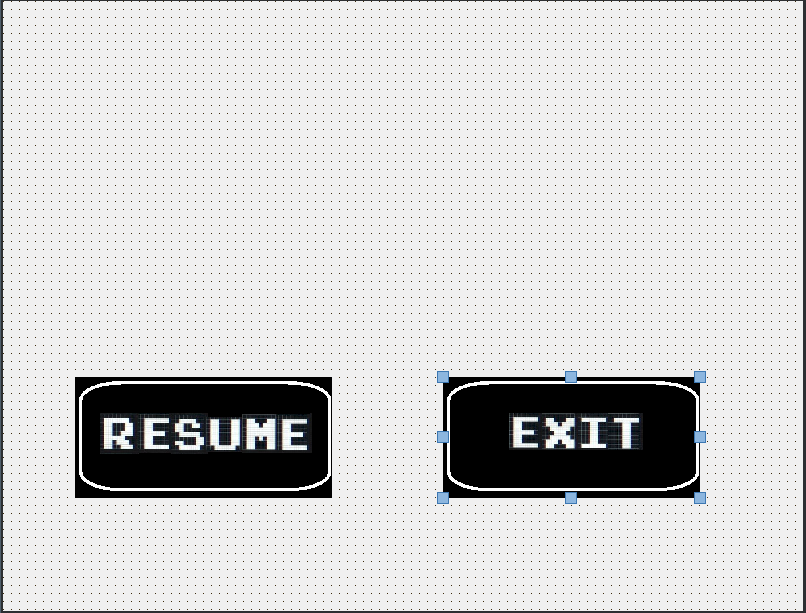


Рисунок 2.4 – Макет окна паузы

Для более удобного взаимодействия с программным средством планируется добавить вызов окна паузы с помощью нажатия на клавишу «Escape».

2.2.5 Окно окончания игры

Окно окончания игры будет содержать информацию о том, что игра закончилась. Макет окончания игры представлен на рисунке 2.5.

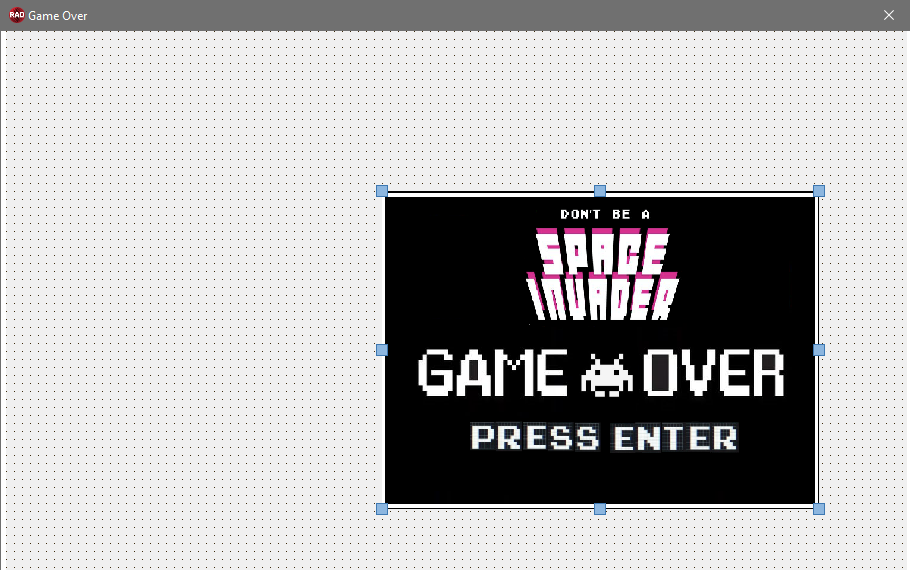


Рисунок 2.5 – Макет окна окончания игры

Для выхода из окна окончания игры будет ожидаться нажатие на клавишу Enter.

2.2.6 Окно игрового процесса

Окно игрового процесса будет состоять из множества элементов:

* информация о количестве жизней игрока;
* кнопка паузы;
* главная пушка и ее снаряды;
* враги;
* счет игрока.

На экране будут отображены все враги и главная пушка. Игровой процесс постоянно прогружает новые состояния врагов с помощью компонента TTimer, отвечающего за вывод за определенные промежутки времени нового изображения. Макет окна игрового процесса представлен на рисунке 2.6.

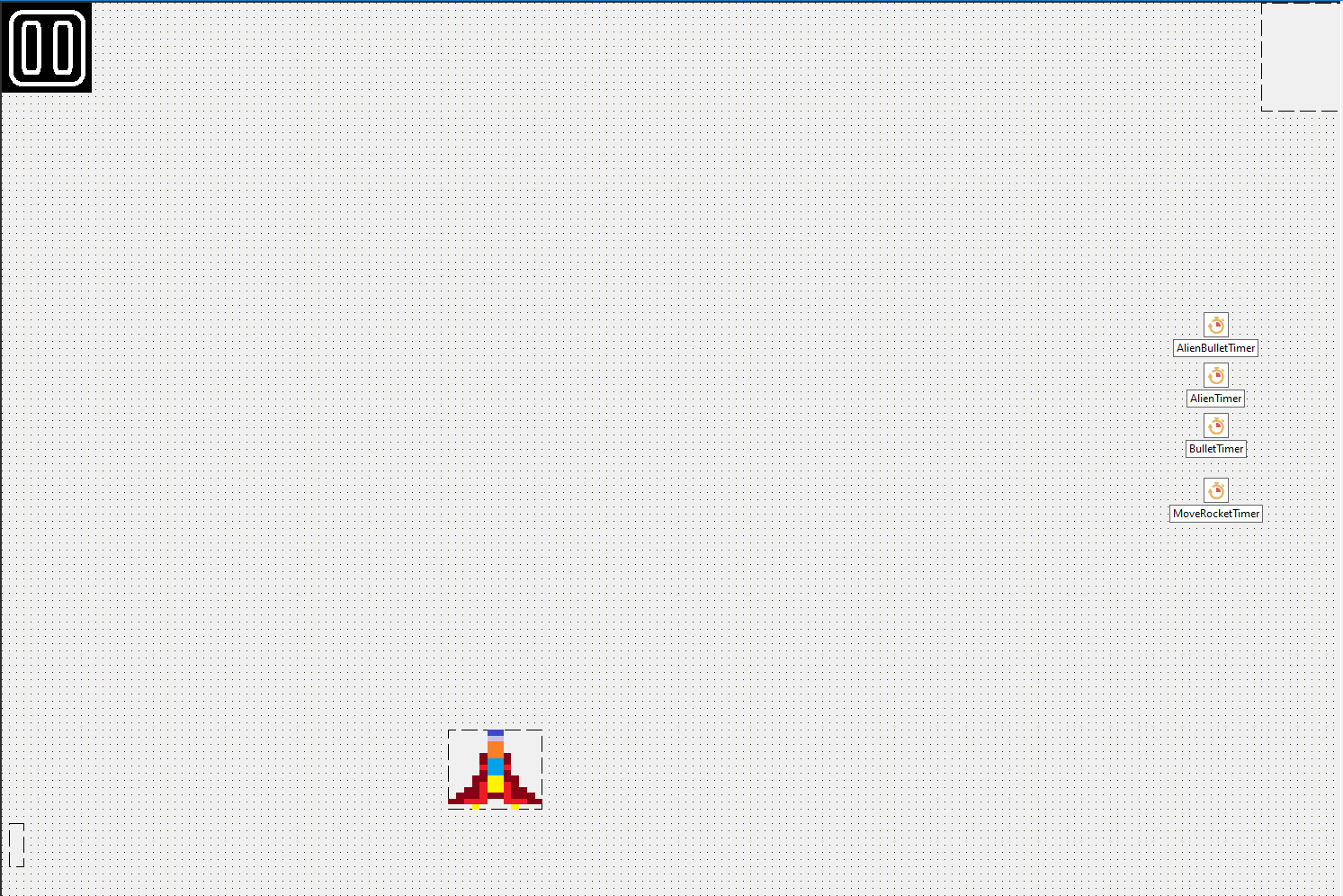


Рисунок 2.6 – Макет окна игрового процесса

Будет реализован выход в главное меню из окна игрового процесса.

2.2.7 Окно рекордов игроков

Окно рекордов игроков состоит из таблицы рекордов и кнопки подтверждения. Макет окна рекордов игроков представлен на рисунке 2.7.



Рисунок 2.7 – Макет окна рекордов игроков

В окне рекордов игроков будет предусмотрена возможность пролистывания таблицы.

## 2.3 Проектирование функционала программного средства

Разработка алгоритмов является ключевой фазой в проектировании программного средства. Игровое приложение «Space Invaders» должно предоставлять пользователю такой минимальный функуционал, как:

* отрисовка игрового поля, пушки и врагов;
* выстрелы пушки;
* движение врагов.

2.3.1 Отрисовка игрового поля, врагов и пушки

Отрисовка игрового поля, врагов и пушки в игре может быть реализована с использованием компонентов TImage и TBitMap, которые представляют собой графические контейнеры. При отрисовке игрового поля необходимо учесть размеры врагов и пушки заранее.

Первоначально отрисовываются фон, враги, пушка и количество жизней игрока на игровом поле. Затем определяются границы врагов и главной пушки.

Во время старта таймера враги принимают начальное расположение на игровом поле. Далее происходит рассчет их границ.

После отрисовки врагов на игровом поле можно перейти к их передвижению. Враги двигаются по предопределенному маршруту, постепенно приближаясь к главной пушке. Враги могут быть представлены в виде графических объектов, размещенных на игровом поле в виде таблицы. Блок-схема алгоритма процедуры SetAliensStartPos приведена на рисунке 2.8.



Рисунок 2.8 – Блок-схема процедуры SetAliensStartPos

2.3.2 Выстрелы пушки

Выстрелы пушки основаны на простом перемещении снаряда по координатам. При передвижении пули с каждым перемещением проверяется пересечение пули с врагом или с границей игового поля. После пересечения пуля пропадает с игового поля. Блок-схема алгоритма процедуры MoveRocketBullet приведена на рисунке 2.9.



Рисунок 2.9 – Блок-схема процедуры MoveRocketBullet

После выполнения этой процедуры необходимо будет проверять пересечение с границами врага и границей игрового поля.

2.3.3 Движение врагов

Алгоритм движения врагов включает в себя обновление параметров позиции каждого врага по обеим координатам и состояния. На рисунке 2.10 приведена блок-схема алгоритма процедуры MoveAliens.



Рисунок 2.10 – Блок-схема процедуры MoveAliens

Граммотное проектирование проекта поможет в будущем избежать ошибок при разработке.

1. **РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**3.1 Прорисовка материалов для игры**

Одними из важных процедур программы являются те, что связаны с прорисовкой текущего состояния игрового поля, информационной части об игре. В основу работы с отображением были добавлены компоненты TImage и TBitMap.

**3.1.1** Прорисовка движения пушки

Прорисовка пушки игрока происходит после нажатия на стрелки на клавиатуре. Отрисовка пушки производится с помощью компонета TBitMap. Код процедуры MoveRocket приведен ниже.

Procedure MoveRocket(Var RocketImage: TImage; Const Form: TForm; Var LeftPressed, RightPressed: Boolean);

Begin

// Проверка на пересечение левой границы

If (RocketImage.Left < 0) Then

// Запрет движения влево

LeftPressed := False;

// Проверка на пересечение правой границы

If (RocketImage.Left + RocketImage.Width > Form.ClientWidth) Then

// Запрет движения вправо

RightPressed := False;

// Нажата ли стрелка влево

if (LeftPressed) then

// Перемещение пушки влево

RocketImage.Left := RocketImage.Left - ROCKET\_SPEED;

// Нажата ли стрелка вправо

if (RightPressed) then

// Перемещение пушки вправо

RocketImage.Left := RocketImage.Left + ROCKET\_SPEED;

End;

**3.1.2** Прорисовка снарядов врагов

Снаряды отображаются по мере выстрелов врагов. Если количество снарядов превышает максимум, то враги перестают стрелять до момента уменьшения количества снарядов на экране. Код процедуры MoveAlienBullet приведен ниже.

Procedure MoveAlienBullet(Var RocketImage: TImage; Var AlienBullet: TAlienBullet; Сonst DownSide: Integer; Var Lives: Integer);

Var

I: Integer;

Begin

// Проходимся по всем снарядам врагов

For I := 0 To High(AlienBullet) Do

Begin

// Проверяем наличие снаряда на экране

If AlienBullet[I].Visible Then

Begin

// Смещаем снаряд вниз

AlienBullet[I].PosY := AlienBullet[I].PosY + ALIEN\_BULLET\_SPEED;

// Проверяем пересек ли снаряд пушку

If CheckAlienBulletCollision(AlienBullet[I], RocketImage) Then

Begin

// Убираем снаряд с игрового поля

AlienBullet[I].Visible := False;

// Отнимаем одну жизнь пушке

Dec(Lives);

End;

// Проверяем пересек ли снаряд нижнюю границу поля

If (AlienBullet[I].PosY > DownSide) Then

Begin

// Убираем снаряд с игрового поля

AlienBullet[I].Visible := False;

End;

End;

End;

End;

**3.1.3** Отрисовка снаряда пушки

С помощью процедуры BulletTimerTimer можно отобразить снаряд пушки. Данная процедура реализована с помощью TTimer. Код процедуры BulletTimerTimer приведен ниже.

Procedure TSpaceInvadersForm.BulletTimerTimer(Sender: TObject);

Var

CountOfkilled: Integer;

Begin

// Передвижение снаряда пушки

MoveRocketBullet(BulletImage, IsShoot);

// Установка начального значения счетчика пораженных врагов

CountOfkilled := 0;

// Проходимся по всем врагам

For Var I := 0 To 4 Do

For Var J := 0 To 9 Do

Begin

// Поражен ли враг

If Not Aliens[I, J].Killed Then

// Проверка пересечения снаряда пушки и границ врага

If CheckBulletCollision(Aliens[I, J], BulletImage) Then

Begin

// Уничтожение врага

KillAlien(Aliens[I, J], BulletImage, IsShoot, Score);

// Отрисовка картинки взрыва

Canvas.Draw(Aliens[I, J].PosX, Aliens[I, J].PosY, BoomBitMap);

End;

// Враг поражен

If Aliens[I, J].Killed Then

Begin

// Увеличение счетчика пораженных врагов на один

Inc(CountOfKilled);

End;

End;

// Поражены ли все враги

If CountOfkilled = ALL\_ALIENS Then

Begin

// Перезапись данных всех врагов

InitializeAliens(Aliens);

// Установка начального положения для врагов

SetAliensStartPos(Aliens);

End;

End;

Очень важно при отрисовке вовремя изменять параметры состояния врагов, снарядов и пушки.

**3.2 Игровой процесс**

При разработке программного средства основной упор делался на игровую логику. Ниже будут рассмотрены основные подпрограммы для реализации игрового процесса.

**3.2.1** Высчитывание новых параметров для врагов

Данная функция довольна важна для динамики игры. Код функции CalculateAlienHitBox приведен ниже.

Function CalculateAlienHitBox(Alien: TAlienRec): TRect;

Var

ImageWidth, ImageHeight: Integer;

HitBoxLeft, HitBoxRight, HitBoxTop, HitBoxBottom: Integer;

HitBox: TRect;

Begin

// Сохранение ширины врага

ImageWidth := Alien.Image.Width;

// Сохранение высоты врага

ImageHeight := Alien.Image.Height;

// Сохранение левой границы врага

HitBoxLeft := Alien.PosX;

// Сохранение правой границы врага

HitBoxRight := HitBoxLeft + ImageWidth;

// Сохранение верхней границы врага

HitBoxTop := Alien.PosY;

// Сохранение нижней границы врага

HitBoxBottom := HitBoxTop + ImageHeight;

// Рассчет границ врага

HitBox := Rect(HitBoxLeft, HitBoxTop, HitBoxRight, HitBoxBottom);

CalculateBulletHitBox := HitBox;

End;

Данная процедура обходит всех врагов для пересчета границ после их передвижения.

**3.2.2** Высчитывание крайних врагов

Для корректной реализации логики игры нужно проверять состояние крайних неуничтоженных врагов. Каждый враг, стоящий на краю, должен коснуться границы игрового поля. При касании края игрового поля, враг меняет направление на противоположное. Код функции FindBorderColumn приведен ниже.

Function FindBorderColumn(Aliens: TAlien; IsFirstCol: Boolean): Integer;

Var

I, J, LastCol, FirstCol, CountInCol: Integer;

Begin

// Установка левой границ

LastCol := ALIENS\_COL;

FirstCol := 0;

// Количество в столбце

CountInCol := 0;

// Первый ли столбец врагов

If IsFirstCol Then

Begin

// Проход по всем врагам

For J := ALIENS\_COL DownTo 0 Do

Begin

For I := 0 To ALIENS\_ROW Do

Begin

// Уничтожен ли враг в столбце

If Not Aliens[I, J].Killed Then

// Увеличиваем счетчик уничтоженых врагов

Inc(CountInCol);

End;

// Проверка количества врагов в столбце

If CountInCol > 0 Then

// Смещение первого столбца врагов

FirstCol := J;

// Обнуление счетчика врагов в столбце

CountInCol := 0;

End;

End;

FindBorderColumn := FirstCol;

End;

**3.2.4** Уничтожение врагов

Процедура реализует уничтожение врагов, которые были поражены снарядом пушки. Код процедуры KillAlien приведен ниже.

Procedure KillAlien(Var Alien: TAlienRec; Var BulletImage: TImage; Var IsShoot: Boolean; Var Score: Integer);

Begin

// Удаление снаряда пушки с игрового поля

BulletImage.Visible := False;

// Возобновление возможности выстрела пушки

IsShoot := False;

// Перевод состояния врага в уничтоженное

Alien.Killed := True;

// Очищение изображения врага

Alien.Image.Destroy;

// Обновление счета игрока

Score := Score + Alien.Score;

End;

**3.2.5** Выстрелы врагов

Переодически враги должны стрелять в сторону пушки. Стрельба врагов не должна быть предсказуема и для этого используется функция, которая возвращает случайные числа. Код процедуры ShootAlienBullet приведен ниже.

Procedure ShootAlienBullet(Var Aliens: TAlien; Var Bullets: TAlienBullet);

Var

CurrAlien: Integer;

Begin

// Изменение зерна случайности

Randomize;

// Возвращение случайнго числа в диапазоне 10

CurrAlien := RandomRange(1, 11) - 1;

// Уничтожен ли враг в верхнем ряду

If Not Aliens[0, CurrAlien].Killed Then

Begin

// Рассчет координаты Х для снаряда врага

Bullets[CurrAlien].PosX := (Aliens[0, CurrAlien].PosX + Aliens[0, CurrAlien].Image.Width);

// Рассчет координаты Y для снаряда врага

Bullets[CurrAlien].PosY := (Aliens[0, CurrAlien].PosY + Aliens[0, CurrAlien].Image.Width);

// Создание снарда на экране

Bullets[CurrAlien].Visible := True;

End;

End;



## Работа со списком игроков

Работа со списком игроков включает в себя использование однонаправленного списка. Необходимо разработать процедуры добавления, удаления элементов списка, а также очищение списка.

**3.3.1** Добавление нового элемента в список

Игроки представляют собой однонаправленный список. Данная структура данных была выбрана по той причине, чтобы более комфортно хранить информацию об игроках. Для того чтобы обновлять данные, было принято добавлять информацию о новом игроке в конец списка. Код процедуры AddPlayer приведен ниже.

Procedure AddPlayer(Var PlayerList: Player; Const Name: String; Score: Integer);

Var

NewPlayer, CurrPlayer: Player;

Begin

// Создание нового узла

NewPlayer := CreatePlayer(Name, Score);

// Список пуст

If PlayerList = Nil Then

PlayerList := NewPlayer

Else

Begin

// Получаем указатель на конец списка

CurrPlayer := PlayerList;

While CurrPlayer^.Next <> Nil Do

CurrPlayer := CurrPlayer^.Next;

// Добавляем новый элемент в конец

CurrPlayer^.Next := NewPlayer;

End;

End;

* + 1. Удаление элемента из списка

Для удаления игрока из списка надо пройтись по всему списку и извлечь элемент. Код функции DeletePlayer приведен ниже.

Procedure DeletePlayer(Var PlayerList: Player; Const Name: String);

Var

CurrPlayer, TempPlayer: Player;

Begin

// Равна ли удаляемая шашка с головой списка

If IsPlayerDataEqual(PlayerList.Name, Name) Then

Begin

TempPlayer := PlayerList;

PlayerList := PlayerList^.Next;

End

Else

Begin

// Проходимся по списку и сравниваем игроков

CurrPlayer := PlayerList;

While Not IsPlayerDataEqual(CurrPlayer^.Next^.Name, Name) Do

CurrPlayer := CurrPlayer^.Next;

TempPlayer := CurrPlayer^.Next;

CurrPlayer^.Next := CurrPlayer^.Next^.Next;

End;

// Удаляем шашку

Dispose(TempPlayer);

End;

**3.3.3** Очищение игроков

Важным аспектом использования динамической памяти является очищение. Код процедуры ClearPlayer приведен ниже.

Procedure ClearPlayer(Var PlayerList: Player);

Var

CurrPlayer, TempPlayer: Player;

Begin

// Получаем указатель на первый элемент списка

CurrPlayer := PlayerList;

While CurrPlayer <> Nil Do

Begin

// Поочередно удаляем элементы из списка

TempPlayer := CurrPlayer;

CurrPlayer := CurrPlayer^.Next;

Dispose(TempPLayer);

End;

// Разрываем указатель и обнуляем значения количества

PlayerList := Nil;

End;

Во время разработки игры необходимо корректно использовать аппаратные ресурсы и недопускать утечки памяти.

# 4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Основной проблемой была некорректная работа со списком игроков. Проблема представлена на рисунке 4.1.

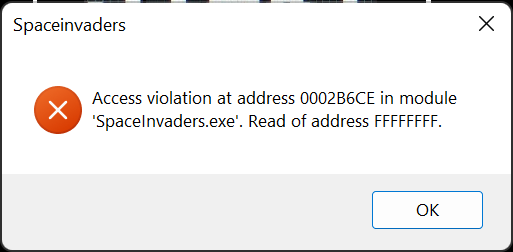


Рисунок 4.1 – Некорректная работа со списками

Возникала проблема при добавлении нового игрока. Проблема была решена дополнительной фунцией инициализации нового элемента списка. Код процедуры инициализации элемента списка представлен ниже.

Function CreatePlayer(Name: String; Score: Integer): Player;

Var

NewPlayer: Player;

Begin

// Выделение памяти для нового элемента

New(NewPlayer);

// Заполнение полей записи

NewPlayer.Name := Name;

NewPlayer.Score := Score;

NewPlayer.HighScore := Score;

// Указатель на следющий элемент равен Nil

NewPlayer^.Next := Nil;

CreatePlayer := NewPlayer;

End;

Большинство проблем возникло из-за недочетов на стадии проектирования программного средства, на стадии тестирования приложения все проблемы были исправлены.

**5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**5.1 Интерфейс программного средства**

Внешний вид программных средств играет существенную роль в обеспечении удобства пользователей и качества самого средства.

**5.1.1** Главное меню

Главное меню приложения состоит из четырех основных кнопок:

* первой располагается кнопка, по нажатию которой пользователь переходит в окно игры;
* второй располагается кнопка «Player», позволяющая управлять списком игроков;
* третьей кнопкой является «Score», нажав на которую пользователь может просмотреть таблицу рекордов игроков;
* четвертой является кнопка «Выйти», которая позволяет покинуть программное средство.

Внешний вид главного меню приложения представлен на рисунке 5.1.

****

Рисунок 5.1 – Главное окно приложения

Выход из программного средства можно осуществить нажатием на клавишу «Escape», находясь в главном окне приложения.

5.1.2 Окно «Player»

Окно «Player» содержит управление списком игроков. При открытии этого окна можно добавлять, удалять или выбирать игрока. Окно «Player» представлено на рисунке 5.2.

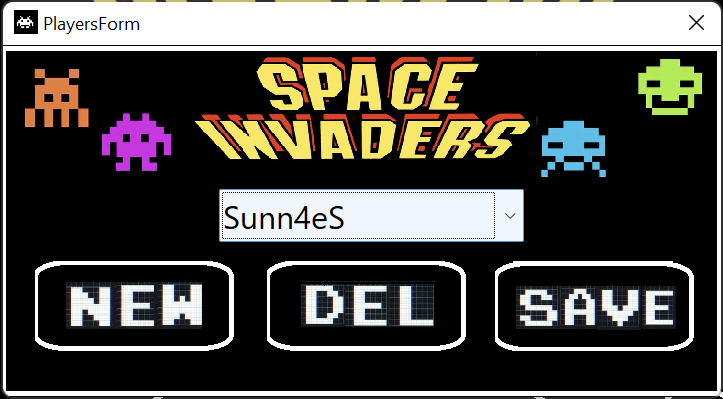


Рисунок 5.2 – Окно «Player»

Для добавления нового игрока, пользователю нужно нажать на кнопку «New». После добавления нового игрока его имя появится в выпадающем списке. При удалении игрока из списка, его информация удалится и в таблице рекордов.

5.1.3 Окно «Score»

Окно «Score» содержит таблицу рекордов игроков. Окно «Score» представлено на рисунке 5.3.



Рисунок 5.3 – Окно «Score»

Когда число игроков в таблице достигает значения большего 10, появляется полоса прокрутки. Новый рекорд заноситься в таблицу сразу после игры.

5.1.4 Окно паузы

Окно паузы состоит из двух основных кнопок:

* первой располагается кнопка, по нажатию которой пользователь возвращается к игре;
* второй кнопкой является «Exit», нажав на которую, пользователь вернется в главное меню.

Окно паузы представлено на рисунке 5.4.



Рисунок 5.4 – Окно паузы

Также для возврата в игру можно нажать клавишу «Escape» на клавиатуре.

5.1.5 Окно окончания игры

Окно окончания игры содержит сообщение о том, что игра закончена. Окно окончания игры представлено на рисунке 5.5.



Рисунок 5.5 – Окно окончания игры

Для подтверждения окончания игры пользователю необходимо нажать клавишу «Enter». После нажатия на «Enter», произойдет закрытие окна окончания игры и переход в главное меню.

5.1.6 Окно игрового процесса

Окно игрового процесса состоит из множества элементов:

* информацию о количестве жизней пушки;
* кнопка паузы;
* враги;
* снаряды врагов;
* пушка;
* снаряд пушки;
* счет игрока.

Окно игрового процесса представлено на рисунке 5.6.



Рисунок 5.6 – Окно игрового процесса

В окне игрового процесса предусмотрен выход в главное меню, с помощью окна паузы. Счет обновляется после уничтожения каждого врага.

**5.2 Управление программным средством**

**5.2.1** Взаимодействие с окном игрового процесса

Управление пушкой осуществляется при помощи нажатия на стрелки влево и вправо, а также клавиши пробела. Для передвижения пушки пользователь должен использовать стрелки влево и вправо на клавиатуре. Выстрел пушки осуществляется с помощью нажатия на пробел. Для того чтобы уничтожить врага, пользователю необходимо переместить пушку на одну линию с врагом. После того как пушка окажется под врагом необходимо произвести выстрел, нажатием на клавишу пробел.

При необходимости выйти из игры игрок может воспользоваться окном паузы по нажатию на клавишу «Escape». Для возврата из меню паузы так же можно воспользоваться клавишей «Escape». Внешний вид окна игрового процесса представлен на рисунке 5.7.



Рисунок 5.7 – Окно игрового процесса

При уничтожении всех врагов на экране появляется новая волна для продолжения игры. Игра длится до момента, пока игрок не потеряет три жизни. После утраты всех жизней открывается окно окончания игры.

**5.2.1** Взаимодействие с главным окном

Для перехода в окно игрового процесса нужно нажать конпку «Start». Для перехода в окно «Player» и выбора игрока требуется нажать на кнопку «Player». Для просмотра таблицы рекордов игроков нужно нажать кнопку «Score». После нажатия на кнопку «F1» откроется меню с информацией о разработчике. Внешний вид главного окна представлен на рисунке 5.8.



Рисунок 5.8 – Главное окно

Интерфейс должен удовлетворять следующим требованиям: простота и понятность в использовании.

# ЗАКЛЮЧЕНИЕ

В современном мире компьютерные игры играют значительную роль в обеспечении отдыха и развлечения людей. В рамках данного курсового проекта было разработано программное средство «Space Invaders», которое предоставляет пользователям операционной системы Windows возможность провести время вместе в атмосфере ретро аркадных игр.

Во время разработки программного средства были успешно выполнены все поставленные задачи:

* работа с внешними устройствами;
* выстрелы пушки;
* механика жизней главной пушки;
* движение главной пушки и врагов;
* создание таблицы с рейтингом игроков.

Реализованы следующие функции:

* подсчет текущего числа очков;
* выстрелы врагов в главную пушку.

Для успешного достижения всех поставленных целей при разработке данного приложения было необходимо углубленно изучить объектно-ориентированные и формовые возможности языка программирования Delphi. Это позволило эффективно использовать функциональность данного языка при создании пользовательского интерфейса, обеспечить визуальное представление данных и взаимодействие с пользователем. Усвоение указанных аспектов языка позволило успешно реализовать все поставленные задачи в рамках разработки приложения.

Однако существуют пути для улучшения данного программного средства. Например, можно добавить звуковое сопровождение, которое поможет создать более приятную атмосферу во время игры. Также можно реализовать разные режимы игры с разными настройками, что добавит разнообразия и дополнительные вызовы в игровой процесс.

В целом «Space Invaders» представляют собой интересное программное средство, которое предоставляет возможность насладиться игрой в атмосфере ретро акрад. За счет разработанных компонентов пользователь может полностью погрузиться в увлекательную игру.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Архангельский А.Я. Delphi 2006. Язык Delphi, классы, функции Win32. Справочное пособие / [Текст]. – М.: Бином-Пресс, 2011. – 1152 с. – ISBN 978-5-9518-0336-8.
2. Фаулер М. Предметно-ориентированные языки программирования / [Текст]. – М.: Вильямс, 2011. – 576 с. – ISBN 978-5-8459-1738-6.
3. Объектно-Ориентированное Программирование [Интернет-ресурс]. – https://www.bsuir.by/m/12\_100229\_1\_90135.pdf.
4. Григорьев А.Б. О чем не пишут в книгах по Delphi [Текст]. – СПб.: БХВ-Петербург, 2010. – 576 с. – ISBN 978-5-699-40703-3.
5. Парижский С.М. Delphi. Учимся на примерах /Под ред. Ю. А. Шпака [Текст]. – Киев: МК-Пресс, 2005. – 216 с. – ISBN 978-5-9775-0516-1.
6. Уилсон, С. Принципы проектирования и разработки программного обеспечения, yчебн. курс. [Текст]. – СПб, 2003. – 47 с.
7. Embarcadero Documentation [Интернет-ресурс]. – https://docwiki.embarcadero.com/RADStudio/Sydney/en/Documentation.
8. Шупрута, В. В. Delphi 2005. Учимся программировать: / В. В.  
   Шупрута. – Москва: изд. «НТ Пресс», 2001. – 140 с.
9. Программирование на языке Delphi / Д. А. Сурков [и др.]. – учеб.  
   пособие. – Режим доступа: http://www.rsdn.ru/?summary/3165.xml, – 2005.

# ПРИЛОЖЕНИЕ А

Текст программы

Unit StartMenuUnit;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Imaging.Pngimage;

Type

TStartMenuForm = Class(TForm)

StartMenuBackground: TImage;

StartButtonImage: TImage;

PlayerButtonImage: TImage;

ScoreListButtonImage: TImage;

SettingsButtonImage: TImage;

ExitButtonImage: TImage;

Procedure CreateForm(FormClass: TFormClass);

Procedure StartButtonImageClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Procedure FormClose(Sender: TObject; Var Action: TCloseAction);

Procedure CreateParams(Var Params: TCreateParams); Override;

Procedure ExitButtonImageClick(Sender: TObject);

Procedure PlayerButtonImageClick(Sender: TObject);

Procedure FormKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure ScoreListButtonImageClick(Sender: TObject);

Procedure SavePlayersToFile();

Procedure ReadPlayersFromFile();

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

StartMenuForm: TStartMenuForm;

InGame: Boolean = True;

Implementation

{$R \*.dfm}

Uses

GameFormUnit, GameOverUnit, PlayerFormUnit, ScoreBoardUnit, FileUnit,

PlayersListUnit;

Procedure TStartMenuForm.CreateParams(Var Params: TCreateParams);

Begin

Inherited;

Params.ExStyle := Params.ExStyle Or WS\_EX\_APPWINDOW;

End;

Procedure TStartMenuForm.ExitButtonImageClick(Sender: TObject);

Begin

IsStartOpen := False;

ShouldClose := True;

Close;

End;

Procedure TStartMenuForm.CreateForm(FormClass: TFormClass);

Var

Form: TForm;

Begin

StartMenuForm.Visible := False;

Form := FormClass.Create(StartMenuForm);

Form.Icon := StartMenuForm.Icon;

Form.ShowModal;

StartMenuForm.Visible := True;

End;

Procedure TStartMenuForm.FormClose(Sender: TObject; Var Action: TCloseAction);

Begin

ShouldClose := True;

End;

Procedure TStartMenuForm.FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Var

Confirmation: Integer;

Begin

If Not IsStartOpen Then

Begin

Confirmation := Application.MessageBox('Вы действительно хотите выйти?',

'Выход', MB\_YESNO + MB\_ICONQUESTION + MB\_DEFBUTTON2);

ShouldClose := Confirmation = IDYES;

CanClose := Confirmation = IDYES;

SavePlayersToFile();

End

Else

Begin

CanClose := True;

IsStartOpen := False;

ShouldClose := False;

SavePlayersToFile();

End;

End;

Procedure TStartMenuForm.SavePlayersToFile();

Var

OutputFile: TPlayerFile;

TempPlayerList: Player;

ProjectPath: String;

FileName: String;

Begin

ProjectPath := ExtractFilePath(ParamStr(0));

FileName := ProjectPath + 'PlayerData.dat';

AssignFile(OutputFile, FileName);

WriteFileData(OutputFile, PlayersList);

End;

Procedure TStartMenuForm.ReadPlayersFromFile();

Var

InputFile: TPlayerFile;

TempPlayerList: Player;

IsCorrect: Boolean;

ProjectPath: String;

FileName: String;

Begin

ProjectPath := ExtractFilePath(ParamStr(0));

FileName := ProjectPath + 'PlayerData.dat';

TempPlayerList := Nil;

IsCorrect := ReadFileData(InputFile, TempPlayerList, FileName);

If IsCorrect Then

Begin

ClearPlayer(PlayersList);

PlayersList := TempPlayerList;

End

Else

Begin

ClearPlayer(TempPlayerList);

Application.MessageBox('Содержимое файла повреждено!', 'Ошибка',

MB\_OK + MB\_ICONERROR);

End;

End;

Procedure TStartMenuForm.FormCreate(Sender: TObject);

Begin

IsStartOpen := False;

ShouldClose := False;

ReadPlayersFromFile;

StartButtonImage.Visible := PlayerInGame <> Nil;

ScoreListButtonImage.Visible := PlayerInGame <> Nil;

End;

Procedure TStartMenuForm.FormKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

If Key = VK\_ESCAPE Then

Begin

IsStartOpen := False;

Close;

End;

If Key = VK\_F1 Then

MessageBox(Handle, 'Бражалович Александр Иванович.' + #13#10 +

'Студент группы 351004', 'О разработчике',

MB\_OK Or MB\_ICONINFORMATION);

End;

Procedure TStartMenuForm.PlayerButtonImageClick(Sender: TObject);

Begin

PlayersForm := TPlayersForm.Create(StartMenuForm);

PlayersForm.ShowModal;

StartButtonImage.Visible := PlayerInGame <> Nil;

ScoreListButtonImage.Visible := PlayerInGame <> Nil;

End;

Procedure TStartMenuForm.ScoreListButtonImageClick(Sender: TObject);

Begin

ScoreBoardForm := TScoreBoardForm.Create(StartMenuForm);

ScoreBoardForm.ShowModal;

End;

Procedure TStartMenuForm.StartButtonImageClick(Sender: TObject);

Begin

IsStartOpen := True;

ShouldClose := False;

Lives := 3;

SpaceInvadersForm.Visible := True;

SpaceInvadersForm.FormCreate(Self);

Close;

End;

End.

Unit ScoreBoardUnit;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.ExtCtrls,

Vcl.Imaging.Pngimage;

Type

TScoreBoardForm = Class(TForm)

ScoreGrid: TStringGrid;

ScoreBackgroundImage: TImage;

OkButtonImage: TImage;

Procedure OkButtonImageClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure FormKeyPress(Sender: TObject; Var Key: Char);

Procedure ScoreGridKeyPress(Sender: TObject; Var Key: Char);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

ScoreBoardForm: TScoreBoardForm;

Implementation

Uses

PlayersListUnit, GameFormUnit;

{$R \*.dfm}

Procedure TScoreBoardForm.FormCreate(Sender: TObject);

Begin

FillScoreGrid(ScoreGrid, PlayersList);

End;

Procedure TScoreBoardForm.FormKeyPress(Sender: TObject; Var Key: Char);

Begin

If Key = Char(VK\_ESCAPE) Then

Close;

End;

Procedure TScoreBoardForm.OkButtonImageClick(Sender: TObject);

Begin

Close;

End;

Procedure TScoreBoardForm.ScoreGridKeyPress(Sender: TObject; Var Key: Char);

Begin

FormKeyPress(Self, Key);

End;

End.

Unit FileUnit;

Interface

Uses

PlayersListUnit, SysUtils;

Const

MIN\_SCORE = 0;

MAX\_SCORE = Maxint;

MAX\_NAME\_LENGTH = 20;

MAX\_PLAYER\_COUNT = 100;

Type

TPlayerFile = File Of TPlayer;

Function IsCorrectFileData(Const TempPlayerData: TPlayer): Boolean;

Procedure WriteFileData(Var OutputFile: TPlayerFile; Const PlayerList: Player);

Function ReadFileData(Var InputFile: TPlayerFile; Var TempPlayerList: Player;

Const FileName: String): Boolean;

Implementation

Function IsCorrectFileData(Const TempPlayerData: TPlayer): Boolean;

Begin

With TempPlayerData Do

IsCorrectFileData := (HighScore >= MIN\_SCORE) And

(HighScore <= MAX\_SCORE) And (Length(TempPlayerData.Name) >= 1) And

(Length(TempPlayerData.Name) <= MAX\_NAME\_LENGTH);

End;

Function ReadFileData(Var InputFile: TPlayerFile; Var TempPlayerList: Player;

Const FileName: String): Boolean;

Var

IsCorrect: Boolean;

PlayerCount: Integer;

TempPlayerData: TPlayer;

Begin

If Not FileExists(FileName) Then

Begin

AssignFile(InputFile, FileName);

Rewrite(InputFile);

IsCorrect := True;

End

Else

Begin

AssignFile(InputFile, FileName);

Reset(InputFile);

IsCorrect := True;

PlayerCount := 0;

While IsCorrect And (PlayerCount < MAX\_PLAYER\_COUNT) And

Not EOF(InputFile) Do

Begin

Read(InputFile, TempPlayerData);

IsCorrect := IsCorrectFileData(TempPlayerData) And

Not IsExistPlayer(TempPlayerList, TempPlayerData.Name);

If IsCorrect Then

Begin

AddPlayer(TempPlayerList, TempPlayerData.Name,

TempPlayerData.HighScore);

Inc(PlayerCount);

End;

End;

If IsCorrect Then

IsCorrect := PlayerCount <= MAX\_PLAYER\_COUNT;

End;

CloseFile(InputFile);

ReadFileData := IsCorrect;

End;

Procedure WriteFileData(Var OutputFile: TPlayerFile; Const PlayerList: Player);

Var

CurrPlayer: Player;

CurrPlayerData: TPlayer;

Begin

Rewrite(OutputFile);

CurrPlayer := PlayerList;

While CurrPlayer <> Nil Do

Begin

CurrPlayerData.Name := CurrPlayer.Name;

CurrPlayerData.Score := CurrPlayer.Score;

CurrPlayerData.HighScore := CurrPlayer.HighScore;

CurrPlayerData.Next := CurrPlayer.Next;

Write(OutputFile, CurrPlayerData);

CurrPlayer := CurrPlayer^.Next;

End;

CloseFile(OutputFile);

End;

End.

Unit AlienUnit;

Interface

Uses

Vcl.ExtCtrls, Vcl.Graphics, System.Types, Math, DateUtils;

Const

DOWN\_SH = 110;

SIDE\_SH = 120;

START\_X\_POS = 50;

START\_Y\_POS = 100;

ALIEN\_SPEED = 30;

ALIEN\_DOWN\_SPEED = 30;

ALIEN\_BULLET\_SPEED = 50;

SPEED\_COEFF = 1;

ALIENS\_ROW = 4;

ALIENS\_COL = 9;

ALIENS\_COUNT\_IN\_COL = 10;

ALIENS\_RED\_SCORE = 40;

ALIENS\_GREEN\_SCORE = 30;

ALIENS\_PURPLE\_SCORE = 30;

ALIENS\_BLUE\_SCORE = 20;

ALIENS\_YELLOW\_SCORE = 10;

Type

TAlienRec = Record

Image: TBitMap;

Killed: Boolean;

CanShoot: Boolean;

PosX: Integer;

PosY: Integer;

Score: Integer;

End;

TAlienBulletRec = Record

Image: TBitMap;

PosX: Integer;

PosY: Integer;

Visible: Boolean;

End;

TAlien = Array [0 .. 4] Of Array [0 .. 9] Of TAlienRec;

TAlienBullet = Array [0 .. 9] Of TAlienBulletRec;

Procedure InitializeAlienBullets(Var AliensBullets: TAlienBullet);

Procedure InitializeAliens(Var Aliens: TAlien);

Procedure SetAliensStartPos(Var Aliens: TAlien);

Function CalculateAlienHitBox(Alien: TAlienRec): TRect;

Procedure MoveAliens(Var Aliens: TAlien; LeftSide, RightSide, DownSide: Integer;

Var Direction: Boolean);

Function FindBorderColumn(Aliens: TAlien; IsFirstCol: Boolean): Integer;

Procedure ShootAlienBullet(Var Aliens: TAlien; Var Bullets: TAlienBullet);

Function CalculateAlienBulletHitBox(AlienBullet: TAlienBulletRec): TRect;

Function CheckAlienBulletCollision(Const AlienBullet: TAlienBulletRec;

Const Rocket: TImage): Boolean;

Procedure MoveAlienBullet(Var RocketImage: TImage;

Var AlienBullet: TAlienBullet; Const DownSide: Integer; Var Lives: Integer);

Procedure KillAlien(Var Alien: TAlienRec; Var BulletImage: TImage;

Var IsShoot: Boolean; Var Score: Integer);

Function CheckBorderOfRocket(Aliens: TAlien; Rocket: TImage): Boolean;

Implementation

Uses

ImageUnit;

Function CheckBorderOfRocket(Aliens: TAlien; Rocket: TImage): Boolean;

Var

RocketHitBox: TRect;

InterS: Boolean;

I, J: Integer;

Begin

InterS := False;

RocketHitBox := Rect(Rocket.Left, Rocket.Top, Rocket.Left + Rocket.Width,

Rocket.Top + Rocket.Height);

For I := 0 To ALIENS\_ROW Do

For J := 0 To ALIENS\_COL Do

If Not Aliens[I, J].Killed Then

InterS := IntersectRect(CalculateAlienHitBox(Aliens[I, J]),RocketHitBox) Or (Aliens[I, J].PosY + Aliens[I, J].Image.Width > Rocket.Top);

CheckBorderOfRocket := InterS;

End;

Procedure MoveAlienBullet(Var RocketImage: TImage; Var AlienBullet: TAlienBullet; Const DownSide: Integer; Var Lives: Integer);

Var

I: Integer;

Begin

// Проходимся по всем снарядам врагов

For I := 0 To High(AlienBullet) Do

Begin

// Проверяем наличие снаряда на экране

If AlienBullet[I].Visible Then

Begin

// Смещаем снаряд вниз

AlienBullet[I].PosY := AlienBullet[I].PosY + ALIEN\_BULLET\_SPEED;

// Проверяем пересек ли снаряд пушку

If CheckAlienBulletCollision(AlienBullet[I], RocketImage) Then

Begin

// Убираем снаряд с игрового поля

AlienBullet[I].Visible := False;

// Отнимаем одну жизнь пушке

Dec(Lives);

End;

// Проверяем пересек ли снаряд нижнюю границу поля

If (AlienBullet[I].PosY > DownSide) Then

Begin

// Убираем снаряд с игрового поля

AlienBullet[I].Visible := False;

End;

End;

End;

End;

Function CheckAlienBulletCollision(Const AlienBullet: TAlienBulletRec;

Const Rocket: TImage): Boolean;

Var

AlienBulletHitBox: TRect;

RocketHitBox: TRect;

Begin

RocketHitBox := Rect(Rocket.Left, Rocket.Top, Rocket.Left + Rocket.Width,

Rocket.Top + Rocket.Height);

AlienBulletHitBox := CalculateAlienBulletHitBox(AlienBullet);

CheckAlienBulletCollision := IntersectRect(AlienBulletHitBox, RocketHitBox);

End;

Function CalculateAlienBulletHitBox(AlienBullet: TAlienBulletRec): TRect;

Var

ImageWidth, ImageHeight: Integer;

HitBoxLeft, HitBoxRight, HitBoxTop, HitBoxBottom: Integer;

HitBox: TRect;

Begin

// Сохранение ширины врага

ImageWidth := AlienBullet.Image.Width;

// Сохранение высоты врага

ImageHeight := AlienBullet.Image.Height;

// Сохранение левой границы врага

HitBoxLeft := AlienBullet.PosX;

// Сохранение правой границы врага

HitBoxRight := HitBoxLeft + ImageWidth;

// Сохранение верхней границы врага

HitBoxTop := AlienBullet.PosY;

// Сохранение нижней границы врага

HitBoxBottom := HitBoxTop + ImageHeight;

// Рассчет границ врага

HitBox := Rect(HitBoxLeft, HitBoxTop, HitBoxRight, HitBoxBottom);

CalculateAlienBulletHitBox := HitBox;

End;

Procedure InitializeAlienBullets(Var AliensBullets: TAlienBullet);

Var

I: Integer;

Begin

For I := 0 To High(AliensBullets) Do

Begin

AliensBullets[I].Image := TBitMap.Create;;

AliensBullets[I].PosX := 0;

AliensBullets[I].PosY := 0;

AliensBullets[I].Visible := False;

LoadBitMap(AliensBullets[I].Image, 'AlienBullet');

End;

End;

Function FindShootingAliens(Aliens: TAlien): Integer;

Var

I: Integer;

Count: Integer;

Begin

Count := 0;

For I := 0 To ALIENS\_COL Do

Begin

If Not Aliens[0, I].Killed Then

Begin

Inc(Count);

End;

End;

FindShootingAliens := Count;

End;

Procedure ShootAlienBullet(Var Aliens: TAlien; Var Bullets: TAlienBullet);

Var

CurrAlien: Integer;

Begin

// Изменение семени случайности

Randomize;

// Возвращение случайнго числа в диапазоне 10

CurrAlien := RandomRange(1, 11) - 1;

// Уничтожен ли враг в верхнем ряду

If Not Aliens[0, CurrAlien].Killed Then

Begin

// Рассчет координаты Х для снаряда врага

Bullets[CurrAlien].PosX := (Aliens[0, CurrAlien].PosX + Aliens[0, CurrAlien].Image.Width);

// Рассчет координаты Y для снаряда врага

Bullets[CurrAlien].PosY := (Aliens[0, CurrAlien].PosY + Aliens[0, CurrAlien].Image.Width);

// Создание снарда на экране

Bullets[CurrAlien].Visible := True;

End;

End;

Procedure KillAlien(Var Alien: TAlienRec; Var BulletImage: TImage; Var IsShoot: Boolean; Var Score: Integer);

Begin

// Удаление снаряда пушки с игрового поля

BulletImage.Visible := False;

// Возобновление возможности выстрела пушки

IsShoot := False;

// Перевод состояния врага в уничтоженное

Alien.Killed := True;

BulletImage.Top := 0;

// Очищение изображения врага

Alien.Image.Destroy;

// Обновление счета игрока

Score := Score + Alien.Score;

End;

Procedure MoveAliensDirection(Var Aliens: TAlien;

Var Right, Left, Down: Boolean);

Var

I, J: Integer;

Begin

If Left Then

Begin

For I := 0 To ALIENS\_ROW Do

Begin

For J := 0 To ALIENS\_COL Do

Begin

Aliens[I, J].PosX := Aliens[I, J].PosX - ALIEN\_SPEED;

End;

End;

End;

If Right Then

Begin

For I := 0 To ALIENS\_ROW Do

Begin

For J := 0 To ALIENS\_COL Do

Begin

Aliens[I, J].PosX := Aliens[I, J].PosX + ALIEN\_SPEED;

End;

End;

End;

If Down Then

Begin

For I := 0 To ALIENS\_ROW Do

Begin

For J := 0 To ALIENS\_COL Do

Begin

Aliens[I, J].PosY := Aliens[I, J].PosY + ALIEN\_DOWN\_SPEED;

End;

End;

End;

End;

Function FindBorderColumn(Aliens: TAlien; IsFirstCol: Boolean): Integer;

Var

I, J, LastCol, FirstCol, CountInCol: Integer;

Begin

// Установка левой границы

LastCol := ALIENS\_COL;

// Устаноква правой границы

FirstCol := 0;

// Количество в столбце

CountInCol := 0;

// Первый ли столбец врагов

If IsFirstCol Then

Begin

// Проход по всем врагам

For J := ALIENS\_COL DownTo 0 Do

Begin

For I := 0 To ALIENS\_ROW Do

Begin

// Уничтожен ли враг в столбце

If Not Aliens[I, J].Killed Then

Begin

// Увеличиваем счетчик уничтоженых врагов

Inc(CountInCol);

End;

End;

// Проверка количества врагов в столбце

If CountInCol > 0 Then

// Смещение первого столбца врагов

FirstCol := J;

// Обнуление счетчика врагов в столбце

CountInCol := 0;

End;

FindBorderColumn := FirstCol;

End

Else

Begin

// Проход по всем врагам

For J := 0 To ALIENS\_COL Do

Begin

For I := 0 To ALIENS\_ROW Do

Begin

// Уничтожен ли враг в столбце

If Not Aliens[I, J].Killed Then

Begin

Inc(CountInCol);

End;

End;

If CountInCol > 0 Then

LastCol := J;

CountInCol := 0;

End;

FindBorderColumn := LastCol;

End;

End;

Procedure MoveAliens(Var Aliens: TAlien; LeftSide, RightSide, DownSide: Integer;

Var Direction: Boolean);

Var

Down, Left, Right: Boolean;

CrossRight, CrossLeft: Boolean;

Begin

CrossRight := Aliens[0, FindBorderColumn(Aliens, False)].PosX + Aliens[0, 9]

.Image.Width + ALIEN\_SPEED > RightSide;

CrossLeft := Aliens[0, FindBorderColumn(Aliens, True)].PosX - 30 < 0;

If CrossRight And Not CrossLeft Then

Begin

Left := False;

Right := False;

Down := True;

MoveAliensDirection(Aliens, Right, Left, Down);

Direction := False;

End;

If Not CrossRight And CrossLeft Then

Begin

Left := False;

Right := False;

Down := True;

MoveAliensDirection(Aliens, Right, Left, Down);

Direction := True;

End;

If Not CrossRight And Direction Then

Begin

Down := False;

Right := True;

Left := False;

MoveAliensDirection(Aliens, Right, Left, Down);

End;

If Not CrossLeft And Not Direction Then

Begin

Down := False;

Right := False;

Left := True;

MoveAliensDirection(Aliens, Right, Left, Down);

End;

End;

Function CalculateAlienHitBox(Alien: TAlienRec): TRect;

Var

ImageWidth, ImageHeight: Integer;

HitBoxLeft, HitBoxRight, HitBoxTop, HitBoxBottom: Integer;

HitBox: TRect;

Begin

ImageWidth := Alien.Image.Width;

ImageHeight := Alien.Image.Height;

HitBoxLeft := Alien.PosX;

HitBoxRight := HitBoxLeft + ImageWidth;

HitBoxTop := Alien.PosY;

HitBoxBottom := HitBoxTop + ImageHeight;

HitBox := Rect(HitBoxLeft, HitBoxTop, HitBoxRight, HitBoxBottom);

CalculateAlienHitBox := HitBox;

End;

Procedure InitializeAliens(Var Aliens: TAlien);

Var

I, J: Integer;

Begin

For I := 0 To 4 Do

Begin

For J := 0 To 9 Do

Begin

Aliens[I, J].Image := TBitMap.Create;

Aliens[I, J].Killed := False;

Aliens[I, J].CanShoot := False;

Aliens[I, J].PosX := 0;

Aliens[I, J].PosY := 0;

If I = 0 Then

Begin

LoadBitMap(Aliens[I, J].Image, 'RedAlien');

Aliens[I, J].CanShoot := True;

Aliens[I, J].Score := ALIENS\_RED\_SCORE;

End;

If I = 1 Then

Begin

LoadBitMap(Aliens[I, J].Image, 'GreenAlien');

Aliens[I, J].Score := ALIENS\_GREEN\_SCORE;

End;

If I = 2 Then

Begin

LoadBitMap(Aliens[I, J].Image, 'PurpleAlien');

Aliens[I, J].Score := ALIENS\_PURPLE\_SCORE;

End;

If I = 3 Then

Begin

LoadBitMap(Aliens[I, J].Image, 'BlueAlien');

Aliens[I, J].Score := ALIENS\_BLUE\_SCORE;

End;

If I = 4 Then

Begin

LoadBitMap(Aliens[I, J].Image, 'YellowAlien');

Aliens[I, J].Score := ALIENS\_YELLOW\_SCORE;

End;

End;

End;

End;

Procedure SetAliensStartPos(Var Aliens: TAlien);

Var

I, J: Integer;

Begin

For I := 0 To 4 Do

Begin

For J := 0 To 9 Do

Begin

If I = 0 Then

Begin

Aliens[I, J].PosX := J \* SIDE\_SH + START\_X\_POS;

Aliens[I, J].PosY := START\_Y\_POS;

End;

If I = 1 Then

Begin

Aliens[I, J].PosX := J \* SIDE\_SH + START\_X\_POS;

Aliens[I, J].PosY := START\_Y\_POS + DOWN\_SH \* I;

End;

If I = 2 Then

Begin

Aliens[I, J].PosX := J \* SIDE\_SH + START\_X\_POS;

Aliens[I, J].PosY := START\_Y\_POS + DOWN\_SH \* I;

End;

If I = 3 Then

Begin

Aliens[I, J].PosX := J \* SIDE\_SH + START\_X\_POS;

Aliens[I, J].PosY := START\_Y\_POS + DOWN\_SH \* I;

End;

If I = 4 Then

Begin

Aliens[I, J].PosX := J \* SIDE\_SH + START\_X\_POS;

Aliens[I, J].PosY := START\_Y\_POS + DOWN\_SH \* I;

End;

End;

End;

End;

End.

Unit GameFormUnit;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Imaging.Pngimage,

AlienUnit, GameOverUnit, PlayersListUnit;

Const

ROCKET\_BULLET = 0;

ALIEN\_BULLET = 1;

START\_LIVES = 3;

ALL\_ALIENS = 50;

SHIFTR = 70;

Type

Bullets = (RocketBullet, AlienBullet);

TBulletArr = Array [0 .. 1] Of TImage;

TSpaceInvadersForm = Class(TForm)

RocketImage: TImage;

BulletImage: TImage;

BulletTimer: TTimer;

MoveRocketTimer: TTimer;

AlienTimer: TTimer;

AlienBulletTimer: TTimer;

ScorePaintBox: TPaintBox;

PauseButtonImage: TImage;

Procedure FormCreate(Sender: TObject);

Procedure FormKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Procedure BulletTimerTimer(Sender: TObject);

Procedure FormKeyUp(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure MoveRocketTimerTimer(Sender: TObject);

Procedure FormPaint(Sender: TObject);

Procedure AlienTimerTimer(Sender: TObject);

Procedure AlienBulletTimerTimer(Sender: TObject);

Procedure CreateForm(FormClass: TFormClass);

Procedure ScorePaintBoxPaint(Sender: TObject);

Procedure PauseButtonImageClick(Sender: TObject);

Procedure DisableTimers();

Procedure EnableTimers();

Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Private

LeftPressed: Boolean;

RightPressed: Boolean;

Pressed: Boolean;

Aliens: TAlien;

AliensBullets: TAlienBullet;

AlienBulletBitMap: TBitMap;

AlienBitMap: TBitMap;

BackGroundBitMap: TBitMap;

BoomBitMap: TBitMap;

BackAliens: TBitMap;

RocketBoomBitMap: TBitMap;

LifeBitMap: TBitMap;

Public

End;

Var

SpaceInvadersForm: TSpaceInvadersForm;

Bullet: TBulletArr;

IsShoot: Boolean = False;

IsMoving: Boolean = False;

ShouldClose: Boolean = False;

Score: Integer;

Lives: Integer;

IsStartOpen: Boolean = False;

PlayersList: Player;

PlayerInGame: Player;

Implementation

Uses

ImageUnit, RocketUnit, CanvasUnit, PauseUnit, StartMenuUnit;

{$R \*.dfm}

Procedure TSpaceInvadersForm.DisableTimers();

Begin

AlienTimer.Enabled := False;

BulletTimer.Enabled := False;

AlienBulletTimer.Enabled := False;

MoveRocketTimer.Enabled := False;

End;

Procedure TSpaceInvadersForm.EnableTimers();

Begin

AlienTimer.Enabled := True;

BulletTimer.Enabled := True;

AlienBulletTimer.Enabled := True;

MoveRocketTimer.Enabled := True;

End;

Procedure TSpaceInvadersForm.FormCloseQuery(Sender: TObject;

Var CanClose: Boolean);

Var

Confirmation: Integer;

Begin

CanClose := True;

DisableTimers();

End;

Procedure TSpaceInvadersForm.FormCreate(Sender: TObject);

Begin

DisableTimers();

If Not IsStartOpen Then

Begin

SpaceInvadersForm.Visible := False;

CreateForm(TStartMenuForm);

SpaceInvadersForm.Visible := True;

If ShouldClose Then

Begin

Application.Terminate;

End;

End;

DisableTimers();

Bullet[ROCKET\_BULLET] := TImage.Create(Self);

Bullet[ALIEN\_BULLET] := TImage.Create(Self);

InitializeBullets(Bullet);

SpawnRocket(RocketImage, SpaceInvadersForm);

SetBackGround(BackGroundBitMap);

BulletImage.Enabled := False;

AlienBitMap := TBitmap.Create;

BackAliens := TBitMap.Create;

BoomBitMap := TBitmap.Create;

RocketBoomBitMap := TBitmap.Create;

LifeBitMap := TBitmap.Create;

LoadBitMap(BackAliens, 'BackAliens');

LoadBitMap(BoomBitMap, 'Boom');

LoadBitMap(RocketBoomBitMap, 'RocketBoom');

LoadBitMap(LifeBitMap, 'Heart');

Pressed := False;

Lives := 3;

Score := 0;

InitializeAlienBullets(AliensBullets);

InitializeAliens(Aliens);

SetAliensStartPos(Aliens);

EnableTimers();

End;

Procedure TSpaceInvadersForm.FormKeyDown(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

If Key = VK\_ESCAPE Then

PauseButtonImageClick(Self);

If Not Pressed Then

Begin

IsPressed(Key, LeftPressed, RightPressed, Pressed);

End;

SpawnRocketBullet(BulletImage, RocketImage, Key, IsShoot, Bullet);

End;

Procedure TSpaceInvadersForm.FormKeyUp(Sender: TObject; Var Key: Word;

Shift: TShiftState);

Begin

StopRocket(Key, LeftPressed, RightPressed, Pressed);

End;

Procedure TSpaceInvadersForm.FormPaint(Sender: TObject);

Begin

Canvas.Draw(0, 0, BackGroundBitMap);

End;

Procedure TSpaceInvadersForm.MoveRocketTimerTimer(Sender: TObject);

Begin

MoveRocket(RocketImage, SpaceInvadersForm, LeftPressed, RightPressed);

End;

Procedure TSpaceInvadersForm.PauseButtonImageClick(Sender: TObject);

Begin

DisableTimers();

CreateForm(TPauseForm);

End;

Procedure TSpaceInvadersForm.ScorePaintBoxPaint(Sender: TObject);

Begin

With ScorePaintBox Do

Begin

Canvas.Font.Name := 'Vineta BT';

Canvas.Brush.Color := ClBlack;

Canvas.FillRect(ClientRect);

Canvas.Font.Size := 25;

Canvas.Font.Color := ClWhite;

Canvas.TextOut(0, 0, 'Score:');

Canvas.TextOut(300, 0, IntToStr(Score));

End;

End;

Var

Direction: Boolean = True;

Procedure TSpaceInvadersForm.AlienBulletTimerTimer(Sender: TObject);

Begin

ShootAlienBullet(Aliens, AliensBullets);

End;

Procedure TSpaceInvadersForm.CreateForm(FormClass: TFormClass);

Var

Form: TForm;

Begin

Form := FormClass.Create(SpaceInvadersForm);

Form.Icon := SpaceInvadersForm.Icon;

Form.ShowModal;

End;

Procedure TSpaceInvadersForm.AlienTimerTimer(Sender: TObject);

Var

PrevLives: Integer;

I, J, K: Integer;

Begin

RocketImage.Visible := True;

PrevLives := Lives;

Canvas.Draw(0, 0, BackGroundBitMap);

MoveAliens(Aliens, SpaceInvadersForm.Left, SpaceInvadersForm.ClientWidth,

RocketImage.Top, Direction);

MoveAlienBullet(RocketImage, AliensBullets,

SpaceInvadersForm.Height, Lives);

ScorePaintBoxPaint(Self);

For I := 0 To Lives - 1 Do

Canvas.Draw(100 + I \* ShiftR, 10, LifeBitMap);

If CheckBorderOfRocket(Aliens, RocketImage) Then

Lives := 0;

If PrevLives <> Lives Then

Begin

RocketImage.Visible := False;

Canvas.Draw(RocketImage.Left, RocketImage.Top, RocketBoomBitMap);

For I := 0 To 4 Do

For J := 0 To 9 Do

If Aliens[I, J].Killed = False Then

Canvas.Draw(Aliens[I, J].PosX, Aliens[I, J].PosY,

Aliens[I, J].Image);

For I := 0 To Lives - 1 Do

Canvas.Draw(100 + I \* ShiftR, 10, LifeBitMap);

Sleep(500);

If Lives = 0 Then

Begin

DisableTimers();

CreateForm(TGameOverForm);

IsStartOpen := False;

FormCreate(Self);

End;

End;

For I := 0 To High(AliensBullets) Do

If AliensBullets[I].Visible Then

Canvas.Draw(AliensBullets[I].PosX, AliensBullets[I].PosY,

AliensBullets[I].Image);

For I := 0 To 4 Do

For J := 0 To 9 Do

If Aliens[I, J].Killed = False Then

Canvas.Draw(Aliens[I, J].PosX, Aliens[I, J].PosY,

Aliens[I, J].Image);

RocketImage.Repaint;

PauseButtonImage.Repaint;

End;

Procedure TSpaceInvadersForm.BulletTimerTimer(Sender: TObject);

Var

CountOfkilled: Integer;

Begin

// Передвижение снаряда пушки

MoveRocketBullet(BulletImage, IsShoot);

// Установка начального значения счетчика пораженных врагов

CountOfkilled := 0;

// Проходимся по всем врагам

For Var I := 0 To 4 Do

For Var J := 0 To 9 Do

Begin

// Поражен ли враг

If Not Aliens[I, J].Killed Then

// Проверка пересечения снаряда пушки и границ врага

If CheckBulletCollision(Aliens[I, J], BulletImage) Then

Begin

// Уничтожение врага

KillAlien(Aliens[I, J], BulletImage, IsShoot, Score);

// Отрисовка картинки взрыва

Canvas.Draw(Aliens[I, J].PosX, Aliens[I, J].PosY,

BoomBitMap);

End;

// Враг поражен

If Aliens[I, J].Killed Then

Begin

// Увеличение счетчика пораженных врагов на один

Inc(CountOfKilled);

End;

End;

// Поражены ли все враги

If CountOfkilled = ALL\_ALIENS Then

Begin

// Перезапись данных всех врагов

InitializeAliens(Aliens);

// Установка начального положения для врагов

SetAliensStartPos(Aliens);

End;

End;

End.

Unit GameOverUnit;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Imaging.Pngimage,

StartMenuUnit;

Type

TGameOverForm = Class(TForm)

GameOverImage: TImage;

Procedure FormKeyPress(Sender: TObject; Var Key: Char);

Procedure CreateParams(Var Params: TCreateParams); Override;

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

GameOverForm: TGameOverForm;

GameOverExist: Boolean = False;

Implementation

{$R \*.dfm}

Uses

GameFormUnit, PlayersListUnit;

Procedure TGameOverForm.CreateParams(Var Params: TCreateParams);

Begin

Inherited;

Params.ExStyle := Params.ExStyle Or WS\_EX\_APPWINDOW;

End;

Procedure TGameOverForm.FormKeyPress(Sender: TObject; Var Key: Char);

Begin

If Key = #13 Then

Begin

PlayerInGame.Score := Score;

IsStartOpen := False;

CheckHighScore(PlayerInGame);

StartMenuForm.SavePlayersToFile();

Close;

End;

End;

End.

Unit ImageUnit;

Interface

Uses

VCL.Graphics, VCL.ExtCtrls, System.SysUtils, Vcl.Forms, GameFormUnit;

Const

ROCKET\_BULLET = 0;

ALIEN\_BULLET = 1;

Procedure LoadImage(Var Image: TImage; NameOfImage: String);

Procedure InitializeBullets(Var Bullets: TBulletArr);

Procedure LoadBitMap(Var BitMap: TBitMap; NameOfImage: String);

Procedure SetBackGround(Var BitMap: TBitMap);

Implementation

Procedure SetBackGround(Var BitMap: TBitMap);

Begin

BitMap := TBitmap.Create;

LoadBitMap(BitMap, 'BackGrounDImage');

End;

Procedure LoadImage(Var Image: TImage; NameOfImage: String);

Var

ImagePath: String;

Begin

ImagePath := ExtractFilePath(Application.ExeName);

ImagePath := ImagePath + '\Images\' + NameOfImage + '.png';

Image.Picture.LoadFromFile(ImagePath);

End;

Procedure LoadBitMap(Var BitMap: TBitMap; NameOfImage: String);

Var

ImagePath: String;

Begin

ImagePath := ExtractFilePath(Application.ExeName);

ImagePath := ImagePath + '\Images\' + NameOfImage + '.bmp';

BitMap.LoadFromFile(ImagePath);

End;

Procedure InitializeBullets(Var Bullets: TBulletArr);

Begin

LoadImage(Bullets[ROCKET\_BULLET], 'BulletRocket');

LoadImage(Bullets[ALIEN\_BULLET], 'BulletAlien');

End;

End.

Unit PauseUnit;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, GameFormUnit, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls,

Vcl.Imaging.Pngimage;

Type

TPauseForm = Class(TForm)

PauseBackGroundImage: TImage;

ResumeButtonImage: TImage;

ExitButtonImage: TImage;

Procedure ResumeButtonImageClick(Sender: TObject);

Procedure FormKeyPress(Sender: TObject; Var Key: Char);

Procedure ExitButtonImageClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

PauseForm: TPauseForm;

Implementation

{$R \*.dfm}

Procedure TPauseForm.ExitButtonImageClick(Sender: TObject);

Begin

Close;

SpaceInvadersForm.DisableTimers();

IsStartOpen := False;

SpaceInvadersForm.FormCreate(Self);

End;

Procedure TPauseForm.FormKeyPress(Sender: TObject; Var Key: Char);

Begin

If (Key = #13) Or (Key = Char(VK\_ESCAPE)) Then

ResumeButtonImageClick(Self);

End;

Procedure TPauseForm.ResumeButtonImageClick(Sender: TObject);

Begin

SpaceInvadersForm.EnableTimers();

Close

End;

End.

Unit RocketUnit;

Interface

Uses

VCL.Forms, Vcl.ExtCtrls, Winapi.Windows, GameFormUnit, AlienUnit,

System.Types;

Const

ROCKET\_SPEED = 30;

BULLET\_SPEED = 30;

Procedure StopRocket(Var Key: Word; Var LeftPressed, RightPressed,

Pressed: Boolean);

Procedure MoveRocket(Var RocketImage: TImage; Const Form: TForm;

Var LeftPressed, RightPressed: Boolean);

Procedure IsPressed(Var Key: Word; Var LeftPressed, RightPressed,

Pressed: Boolean);

Procedure SpawnRocket(Var RocketImage: TImage; Form: TForm);

Procedure SpawnRocketBullet(Bullet: TImage; RocketImage: TImage;

Const Key: Word; Var IsShoot: Boolean; BulletType: TBulletArr);

Procedure MoveRocketBullet(Var BulletImage: TImage; Var IsShoot: Boolean);

Function CheckBulletCollision(Const Alien: TAlienRec;

Const Bullet: TImage): Boolean;

Implementation

Function CheckBulletCollision(Const Alien: TAlienRec;

Const Bullet: TImage): Boolean;

Var

AlienHitBox: TRect;

BulletHitBox: TRect;

Begin

BulletHitBox := Rect(Bullet.Left, Bullet.Top, Bullet.Left + Bullet.Width,

Bullet.Top + Bullet.Height);

AlienHitBox := CalculateAlienHitBox(Alien);

CheckBulletCollision := IntersectRect(BulletHitBox, AlienHitBox);

End;

Procedure MoveRocketBullet(Var BulletImage: TImage; Var IsShoot: Boolean);

Begin

If IsShoot Then

Begin

BulletImage.Top := BulletImage.Top - BULLET\_SPEED;

If BulletImage.Top < 0 Then

Begin

IsShoot := False;

BulletImage.Enabled := False;

BulletImage.Visible := False;

End;

End;

End;

Procedure StopRocket(Var Key: Word; Var LeftPressed, RightPressed,

Pressed: Boolean);

Begin

If Key = VK\_LEFT Then

Begin

LeftPressed := False;

Pressed := False;

End;

If Key = VK\_RIGHT Then

Begin

RightPressed := False;

Pressed := False;

End;

End;

Procedure SpawnRocket(Var RocketImage: TImage; Form: TForm);

Begin

RocketImage.Left := Round((Form.Width - RocketImage.Picture.Width) / 2) - 7

\* RocketImage.Picture.Width;

RocketImage.Top := Round((Form.Height - RocketImage.Picture.Height) / 1.2);

End;

Procedure SpawnRocketBullet(Bullet: TImage; RocketImage: TImage;

Const Key: Word; Var IsShoot: Boolean; BulletType: TBulletArr);

Begin

If Not IsShoot Then

Begin

If Key = VK\_SPACE Then

Begin

Bullet.Picture := BulletType[ROCKET\_BULLET].Picture;

Bullet.Enabled := True;

Bullet.Visible := True;

Bullet.Left := RocketImage.Left + RocketImage.Picture.Width + 46;

Bullet.Top := RocketImage.Top - 55;

IsShoot := True;

End;

End;

End;

Procedure IsPressed(Var Key: Word; Var LeftPressed, RightPressed,

Pressed: Boolean);

Begin

Case Key Of

VK\_LEFT:

Begin

LeftPressed := True;

RightPressed := False;

Pressed := True;

End;

VK\_RIGHT:

Begin

Pressed := True;

LeftPressed := False;

RightPressed := True;

End;

End;

End;

Procedure MoveRocket(Var RocketImage: TImage; Const Form: TForm;

Var LeftPressed, RightPressed: Boolean);

Begin

// Проверка на пересечение левой границы

If (RocketImage.Left < 0) Then

// Запрет движения влево

LeftPressed := False;

// Проверка на пересечение правой границы

If (RocketImage.Left + RocketImage.Width > Form.ClientWidth) Then

// Запрет движения вправо

RightPressed := False;

// Нажата ли стрелка влево

If (LeftPressed) Then

// Перемещение пушки влево

RocketImage.Left := RocketImage.Left - ROCKET\_SPEED;

// Нажата ли стрелка вправо

If (RightPressed) Then

// Перемещение пушки вправо

RocketImage.Left := RocketImage.Left + ROCKET\_SPEED;

End;

End.

Unit PlayersListUnit;

Interface

Uses

Vcl.Grids, Vcl.Forms, SysUtils, VCL.StdCtrls;

Type

Player = ^TPlayer;

TPlayer = Record

Name: String[20];

Score: Integer;

HighScore: Integer;

Next: Player;

End;

Function IsExistPlayer(Const PlayerList: Player; Name: String): Boolean;

Function CreatePlayer(Name: String; Score: Integer): Player;

Procedure AddPlayer(Var PlayerList: Player; Const Name: String; Score: Integer);

Function IsPlayerDataEqual(Const Name1, Name2: String): Boolean;

Procedure DeletePlayer(Var PlayerList: Player; Const Name: String);

Function FindCountOfPlayers(Const PlayerList: Player): Integer;

Function SearchPlayer(Const PlayerList: Player; Name: String): Player;

Procedure CheckHighScore(Var PlayerList: Player);

Procedure FillScoreGrid(Var ScoreGrid: TStringGrid; Const PlayerList: Player);

Function SelectionSort(Const PlayerList: Player): Player;

Procedure ClearPlayer(Var PlayerList: Player);

Implementation

Procedure ClearPlayer(Var PlayerList: Player);

Var

CurrPlayer, TempPlayer: Player;

Begin

// Получаем указатель на первый элемент списка

CurrPlayer := PlayerList;

While CurrPlayer <> Nil Do

Begin

// Поочередно удаляем элементы из списка

TempPlayer := CurrPlayer;

CurrPlayer := CurrPlayer^.Next;

Dispose(TempPLayer);

End;

// Разрываем указатель и обнуляем значения количества

PlayerList := Nil;

End;

Procedure CheckHighScore(Var PlayerList: Player);

Begin

If PlayerList.Score > PlayerList.HighScore Then

PlayerList.HighScore := PlayerList.Score;

End;

Function CreatePlayer(Name: String; Score: Integer): Player;

Var

NewPlayer: Player;

Begin

// Выделение памяти для нового элемента

New(NewPlayer);

// Заполнение полей записи

NewPlayer.Name := Name;

NewPlayer.Score := Score;

NewPlayer.HighScore := Score;

// Указатель на следющий элемент равен Nil

NewPlayer^.Next := Nil;

CreatePlayer := NewPlayer;

End;

Procedure AddPlayer(Var PlayerList: Player; Const Name: String; Score: Integer);

Var

NewPlayer, CurrPlayer: Player;

Begin

// Создание нового узла

NewPlayer := CreatePlayer(Name, Score);

// Список пуст

If PlayerList = Nil Then

PlayerList := NewPlayer

Else

Begin

// Получаем указатель на конец списка

CurrPlayer := PlayerList;

While CurrPlayer^.Next <> Nil Do

CurrPlayer := CurrPlayer^.Next;

// Добавляем новый элемент в конец

CurrPlayer^.Next := NewPlayer;

End;

End;

Function IsPlayerDataEqual(Const Name1, Name2: String): Boolean;

Var

IsEqual: Boolean;

Begin

IsEqual := (Name1 = Name2);

IsPlayerDataEqual := IsEqual;

End;

Function IsExistPlayer(Const PlayerList: Player; Name: String): Boolean;

Var

CurrPlayer: Player;

IsExist: Boolean;

Begin

CurrPlayer := PlayerList;

IsExist := False;

If CurrPlayer <> Nil Then

Repeat

IsExist := IsPlayerDataEqual(CurrPlayer.Name, Name);

CurrPlayer := CurrPlayer^.Next

Until (IsExist) Or (CurrPlayer = Nil);

IsExistPlayer := IsExist;

End;

Procedure DeletePlayer(Var PlayerList: Player; Const Name: String);

Var

CurrPlayer, TempPlayer: Player;

Begin

// Равна ли удаляемая шашка с головой списка

If IsPlayerDataEqual(PlayerList.Name, Name) Then

Begin

TempPlayer := PlayerList;

PlayerList := PlayerList^.Next;

End

Else

Begin

// Проходимся по списку и сравниваем игроков

CurrPlayer := PlayerList;

While Not IsPlayerDataEqual(CurrPlayer^.Next^.Name, Name) Do

CurrPlayer := CurrPlayer^.Next;

TempPlayer := CurrPlayer^.Next;

CurrPlayer^.Next := CurrPlayer^.Next^.Next;

End;

// Удаляем шашку

Dispose(TempPlayer);

End;

Function FindCountOfPlayers(Const PlayerList: Player): Integer;

Var

CurrPlayer: Player;

Count: Integer;

Begin

Count := 0;

CurrPlayer := PlayerList;

If CurrPlayer <> Nil Then

Begin

While CurrPlayer <> Nil Do

Begin

CurrPlayer := CurrPlayer^.Next;

Inc(Count);

End;

End;

FindCountOfPlayers := Count;

End;

Function SearchPlayer(Const PlayerList: Player; Name: String): Player;

Var

CurrPlayer: Player;

Begin

CurrPlayer := PlayerList;

If IsExistPlayer(PlayerList, Name) Then

Begin

While Not IsPlayerDataEqual(CurrPlayer.Name, Name) Do

CurrPlayer := CurrPlayer^.Next;

SearchPlayer := CurrPlayer;

End;

End;

Procedure InitializeGrid(Var ScoreGrid: TStringGrid);

Begin

With ScoreGrid Do

Begin

ColCount := 2;

RowCount := 1;

Cells[0, 0] := 'Player';

Cells[1, 0] := 'High Score';

ColWidths[0] := ScoreGrid.Width Div 2;

ColWidths[1] := ScoreGrid.Width Div 2;

End;

End;

Procedure FillScoreGrid(Var ScoreGrid: TStringGrid; Const PlayerList: Player);

Var

I, CurrRow: Integer;

CurrPlayer: Player;

Begin

InitializeGrid(ScoreGrid);

CurrPlayer := SelectionSort(PlayerList);

ScoreGrid.RowCount := FindCountOfPlayers(PlayerList) + 1;

CurrRow := 1;

While CurrPlayer <> Nil Do

Begin

With ScoreGrid Do

Begin

Cells[0, CurrRow] := CurrPlayer.Name;

Cells[1, CurrRow] := IntToStr(CurrPlayer.HighScore);

End;

Inc(CurrRow);

CurrPlayer := CurrPlayer^.Next;

End;

If ScoreGrid.RowCount > 10 Then

Begin

ScoreGrid.ScrollBars := TScrollStyle.SsVertical;

ScoreGrid.Height := 400

End

Else

Begin

ScoreGrid.ScrollBars := TScrollStyle.SsNone;

ScoreGrid.Height :=

(ScoreGrid.DefaultRowHeight + ScoreGrid.GridLineWidth) \*

ScoreGrid.RowCount + 5;

End;

End;

Function SelectionSort(Const PlayerList: Player): Player;

Var

I, J, MinIndex: Integer;

Temp: Integer;

TempStr: String;

CurrPlayer: Player;

Arr: Array Of Integer;

NameArr: Array Of String;

NewList: Player;

Begin

NewList := Nil;

SetLength(Arr, FindCountOfPlayers(PlayerList));

SetLength(NameArr, FindCountOfPlayers(PlayerList));

CurrPlayer := PlayerList;

I := 0;

While CurrPlayer <> Nil Do

Begin

NameArr[I] := CurrPlayer.Name;

Arr[I] := CurrPlayer.HighScore;

CurrPlayer := CurrPlayer^.Next;

Inc(I);

End;

For I := 0 To High(Arr) - 1 Do

Begin

MinIndex := I;

For J := I + 1 To High(Arr) Do

If Arr[J] > Arr[MinIndex] Then

MinIndex := J;

Temp := Arr[I];

TempStr := NameArr[I];

Arr[I] := Arr[MinIndex];

NameArr[I] := NameArr[MinIndex];

Arr[MinIndex] := Temp;

NameArr[MinIndex] := TempStr;

End;

For I := 0 To FindCountOfPlayers(PlayerList) - 1 Do

AddPlayer(NewList, NameArr[I], Arr[I]);

SelectionSort := NewList;

End;

End.

Unit PlayerFormUnit;

Interface

Uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Imaging.Pngimage,

Vcl.ExtCtrls;

Type

TPlayersForm = Class(TForm)

PlayerComboBox: TComboBox;

PlayerBackgroundImage: TImage;

NewPlayerButtonImage: TImage;

DeletePlayerButtonImage: TImage;

SavePlayerButtonImage: TImage;

Procedure NewPlayerButtonImageClick(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure PlayerComboBoxChange(Sender: TObject);

Procedure SavePlayerButtonImageClick(Sender: TObject);

Procedure DeletePlayerButtonImageClick(Sender: TObject);

Procedure FormKeyPress(Sender: TObject; Var Key: Char);

Procedure PlayerComboBoxKeyPress(Sender: TObject; Var Key: Char);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

PlayersForm: TPlayersForm;

Implementation

Uses

AddNewPlayerUnit, PlayersListUnit, GameFormUnit, StartMenuUnit;

{$R \*.dfm}

Procedure FillComboBox(ComboBox: TComboBox);

Var

CurrPlayer: Player;

Begin

ComboBox.Items.Clear;

CurrPlayer := PlayersList;

While CurrPlayer <> Nil Do

Begin

ComboBox.Items.Add(CurrPlayer.Name);

CurrPlayer := CurrPlayer^.Next;

End;

ComboBox.ItemIndex := 0;

End;

Procedure TPlayersForm.DeletePlayerButtonImageClick(Sender: TObject);

Var

Confirmation: Integer;

Begin

Confirmation := Application.MessageBox

('Вы действительно хотите удалить игрока?', 'Удаление',

MB\_YESNO + MB\_ICONQUESTION + MB\_DEFBUTTON2);

If Confirmation = IDYES Then

Begin

DeletePlayer(PlayersList, PlayerComboBox.Text);

PlayerInGame := Nil;

SavePlayerButtonImage.Visible := PlayersList <> Nil;

FillComboBox(PlayerComboBox);

PlayerComboBoxChange(Self);

StartMenuForm.SavePlayersToFile();

End;

End;

Procedure TPlayersForm.FormCreate(Sender: TObject);

Begin

FillComboBox(PlayerComboBox);

End;

Procedure TPlayersForm.FormKeyPress(Sender: TObject; Var Key: Char);

Begin

If (Key = #13) And (SavePlayerButtonImage.Enabled) And

(SavePlayerButtonImage.Visible) Then

SavePlayerButtonImageClick(Self);

If (Key = Char(VK\_DELETE)) And (DeletePlayerButtonImage.Enabled) And

(DeletePlayerButtonImage.Visible) Then

DeletePlayerButtonImageClick(Self);

If Key = Char(VK\_ESCAPE) Then

Close;

End;

Procedure TPlayersForm.NewPlayerButtonImageClick(Sender: TObject);

Begin

AddNewPlayerForm := TAddNewPlayerForm.Create(PlayersForm);

AddNewPlayerForm.ShowModal;

FillComboBox(PlayerComboBox);

PlayerComboBoxChange(Self);

StartMenuForm.SavePlayersToFile();

End;

Procedure TPlayersForm.PlayerComboBoxChange(Sender: TObject);

Begin

SavePlayerButtonImage.Visible := (PlayerComboBox.ItemIndex >= 0) And

(PlayersList <> NIl);

DeletePlayerButtonImage.Visible := (PlayerComboBox.ItemIndex >= 0) And

(PlayersList <> NIl);

End;

Procedure TPlayersForm.PlayerComboBoxKeyPress(Sender: TObject; Var Key: Char);

Begin

FormKeyPress(Self, Key);

End;

Procedure TPlayersForm.SavePlayerButtonImageClick(Sender: TObject);

Begin

PlayerInGame := SearchPlayer(PlayersList, PlayerComboBox.Text);

Close;

End;

End.

Unit AddNewPlayerUnit;

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,

Vcl.StdCtrls;

Type

TAddNewPlayerForm = Class(TForm)

AddNewPlayerBackGroundImage: TImage;

AddNewPlayerButton: TImage;

NameEdit: TEdit;

Procedure NameEditChange(Sender: TObject);

Procedure AddNewPlayerButtonClick(Sender: TObject);

Procedure FormKeyPress(Sender: TObject; Var Key: Char);

Procedure NameEditKeyPress(Sender: TObject; Var Key: Char);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

AddNewPlayerForm: TAddNewPlayerForm;

Implementation

Uses

PlayersListUnit, GameFormUnit;

{$R \*.dfm}

Procedure TAddNewPlayerForm.AddNewPlayerButtonClick(Sender: TObject);

Begin

If Not IsExistPlayer(PlayersList, NameEdit.Text) Then

Begin

AddPlayer(PlayersList, NameEdit.Text, 0);

End;

Close;

End;

Procedure TAddNewPlayerForm.FormKeyPress(Sender: TObject; Var Key: Char);

Begin

If (Key = #13) And (AddNewPlayerButton.Enabled) And

(AddNewPlayerButton.Visible) Then

AddNewPlayerButtonClick(Self);

If Key = Char(VK\_ESCAPE) Then

Close;

End;

Procedure TAddNewPlayerForm.NameEditChange(Sender: TObject);

Begin

AddNewPlayerButton.Enabled := (NameEdit.Text <> '') And

(Length(NameEdit.Text) < 11);

AddNewPlayerButton.Visible := (NameEdit.Text <> '') And

(Length(NameEdit.Text) < 11) And Not IsExistPlayer(PlayersList,

NameEdit.Text);

End;

Procedure TAddNewPlayerForm.NameEditKeyPress(Sender: TObject; Var Key: Char);

Begin

FormKeyPress(Self, Key);

End;

End.