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 “BALL BLAST”

|  |  |  |
| --- | --- | --- |
| Студент |  | Гладкий М.Г. |
| Руководитель |  | Г.В. Данилова |

Минск 2019

СОДЕРЖАНИЕ

Введение...................................................................................................................3

1. Анализ предметной области...............................................................................5

1.1 Обзор аналогов..............................................................................................5

1.2 Постановка задачи.........................................................................................7

2. Разработка программного средства...................................................................8

2.1 Структура программы...................................................................................8

2.2 Интерфейс программного средства.............................................................8

2.3 Звуковые ресурсы..........................................................................................9

2.4 Игровая логика.............................................................................................11

2.5 Внутриигровой магазин..............................................................................20

2.6 Внешний вид................................................................................................21

3. Тестирование программного средства............................................................23

4. Руководство пользователя................................................................................25

4.1 Правила игры...............................................................................................25

4.2 Интерфейс программы................................................................................26

Заключение.............................................................................................................28

Список использованных источников...................................................................29

Приложение А. Исходный код программы.........................................................30

ВВЕДЕНИЕ

21-й век считается временем развития IT-индустрии. Данные технологии всё больше внедряются в отрасли современной жизни. Также они не обошли сферу развлечений, в которую добавили нечто новое – видеоигры.

История компьютерных игр начинается ещё в далёких 1950-х с разработки простейших игр и симуляций. Данный вид времяпрепровождения не сразу стал популярным, потребовалось несколько десятков лет, чтобы он вышел в свет и стал частью жизни молодёжи.

Первая игра для компьютера Nim и игровой автомат Nimatron были созданы в 1940-м году. Игрок, делая ход, мог погасить одну или более ламп в одном из рядов; далее наступал такой же ход компьютера. Тот, кто выключал последнюю лампу, выигрывал. Изобретение компьютерных игр обычно приписывают кому-то из троих людей: Ральфу Баэру, инженеру, выдвинувшему в 1951-м году идею интерактивного телевидения, А. С. Дугласу, написавшему в 1952-м году OXO – компьютерную реализацию крестиков-ноликов, или Уильяму Хигинботаму, создавшему в 1958-м году игру Tennis for Two.

Spacewar! – одна из первых известных цифровых видеоигр. Она была выпущена в 1962-м. В данной игре два самолета на экране обстреливали друг друга, что задало такой жанр как аркада. Данный жанр долгое время оставался популярным и задал направление будущим компьютерным играм, однако даже сейчас существует много аркадных игр, которые до сих пор притягивают внимание игроков по всему миру.

В сентябре 1971-го года Биллом Питтсом был создан первый аркадный автомат Galaxy Game на базе PDP-11. В ноябре фирма Nutting Associates выпустила около 1500 аркадных автоматов Computer Space, разработанных Ноланом Бушнеллом и Тедом Дабни, из которых были проданы от 500 до 1000 штук. Таким образом Computer Space стала первой видеоигрой, изданной для широкой публики.

24-го мая 1972-го была впервые представлена и продемонстрирована публике Magnavox Odyssey – первая игровая приставка.

В дальнейшем компьютерные игры получили большое развитие. Компании по всему миру соревновались в выпуске передовых игр, приставок и консолей. Наблюдались как упадки, так и резкие возрастания популярности видеоигр. Однако индустрия никогда не стояла на месте. Разработчики всё время пытались привнести что-то новое, создать нечто уникальное. По сей день существует конкуренция между ведущими компаниями в данной сфере, однако с быстрым развитием технологий видеоигры появились и на мобильных устройствах, что привело к их бурному росту на данной платформе. В настоящее время не все могут позволить себе мощный компьютер, который позволял бы комфортно играть в современные игры. На помощь этому и пришли разработчики игр на мобильных устройствах. Почти каждый человек имеет смартфон, который поддерживает множество приложений, в том числе находятся мобильные игры. Для многих поиграть на телефоне намного проще, нежели поиграть на компьютере, ввиду нехватки времени, материальных средств и многих других причин. Большинство людей используют игры по пути на работу, в долгой дороге или когда им абсолютно нечем заняться.

На данный момент существует большое количество компаний, создающих свои приложения на мобильные устройства. В их числе: Ketchapp, Voodoo, Supercell, Rovio, Nekki и др. Все разработчики создают нечто новое, каждый день работая над совершенными проектами. Это даёт пользователям огромный выбор игр, позволяя найти что-то, что подойдёт каждому в отдельности.

Разработчики из Voodoo создали такие знаменитые игры, как: Helix Jump, Dunk Hit, Planet Bomber, Paper.io, Ball Blast и др. Все они уникальны, имеют свои особенности игрового процесса и покорили множество мобильных геймеров.

Целью данного проекта является разработка игрового приложения “Ball Blast”.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

На 2019-й год существует много вариаций “Ball Blast”: с различным оформлением, сложностью игры, уникальными модификациями в виде различных бонусов и тому подобного.

Первая версия игры “Ball Blast” появилась 14-го мая 2018-го года. С течением времени она претерпела большие изменения: была введена возможность смены фона, смены пушки, добавлены дополнительные шары, включающие в себя различные бонусы, такие как: появление дополнительной маленькой пушки, разделение выстрелов, неуязвимость и др. Обновления к этому приложению выпускаются и до сих пор компанией Voodoo.

Одним из аналогов является игра “Jump Ball Blast”. Внешний вид игры показан ниже:



Рисунок 1.1 – Внешний вид “Jump Ball Blast”

Существуют и другие подобия выбранной игры, к примеру “Balls Blast”. Вместо пушки на земле, используется модель космического корабля, который выпускает ракеты в летающие шары. Скриншот игрового процесса прилагается на рисунке 1.2.

Некоторые разработчики не пытаются привнести что-то новое, а используют всем хорошо известное старое. К примеру, создатели приложения “Cannons N Balls” изменили лишь оформление и модели в игре. Главное меню игры указывается в рисунке 1.3.

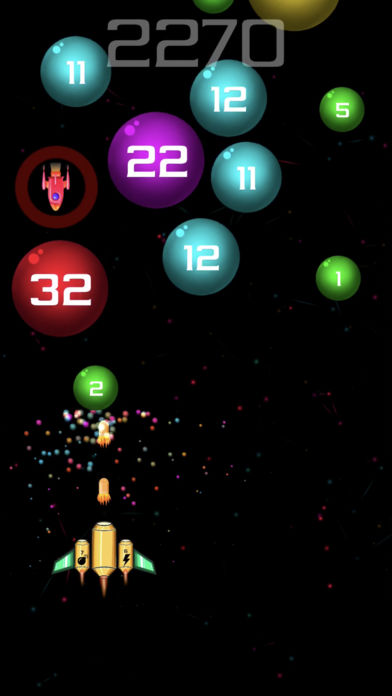


Рисунок 1.2 – Игровой процесс “Balls Blast”



Рисунок 1.3 – Главное меню “Cannons N Balls”

Была описана лишь малая часть созданных вариаций игры “Ball Blast”. Чаще всего разработчиками изменяется лишь внешний вид приложения, однако привносятся и свои коррективы, такие как бонусы.

1.2 Постановка задачи

В рамках данного курсового проекта планируется разработка игрового программного средства “Ball Blast”.

В процессе реализации программы будет разработано главное меню, музыкальное сопровождение.

Планируются следующие функции:

– смена заднего фона игрового поля;

– смена вида пушки;

– встроенный магазин, в котором будет реализована покупка улучшений;

– возможность отключения музыки;

– отображение количества заработанных очков и монет;

– сохранение игрового процесса;

– приостановка игры.

Логическая часть приложения будет представлена в виде:

– взаимодействия структурных блоков;

– появления объектов в разных точках игрового поля;

– алгоритмов проигрыша и получения очков.

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

2. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

2.1 Структура программы

В приложении используются 6 структурных блоков:

– Main – модуль, отображающий главное окно программы;

– Shop – модуль покупок улучшений;

– Skins – модуль изменения пушки;

– Backgrounds – модуль смены заднего фона игрового поля;

– Game – основной логический модуль.

2.2 Интерфейс программного средства

Оформление программного средства и удобство использования являются главными качествами, которые необходимы для работы в приложении. В связи с этим, взаимодействие с пользователем нужно организовать интуитивно понятным и удобным образом.

В качестве навигации был выбран понятный компонент TButton. Для взаимодействия с пользователем установлены следующие опции:

– “Play”

– “Shop”

– “Backgrounds”

– “Skins”

На рисунке 2.1 представлено главное окно программы.

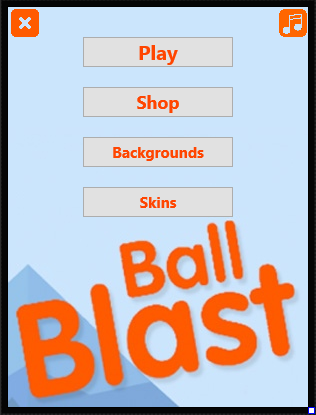


Рисунок 2.1 – Главное окно программы

Для реализации кнопки выхода из программы и выключения музыки был использован компонент TSpeedButton в совокупности с TStyleBook.

Невозможность добавления картинки в компонент TButton вынудила искать другое решение данной проблемы, в связи с чем было выбрано добавление изображения в TSpeedButton.

В результате были нарисованы кнопки “Exit” и “Music”, которые приведены на рисунке ниже:





Рисунок 2.2 – Кнопки “Exit” и “Music”

Для поддержания цветовой гаммы и стилистики игры был использован шрифт Arial, являющийся стандартным в среде разработки Embarcadero Delphi 10.2 Tokyo, 18-го размера и оранжевого цвета.

Эти настройки сделали интерфейс программного средства понятным и простым, но в то же время красивым и уместным.

2.3 Звуковые ресурсы

Программное средство взаимодействует с определенным набором процедур, обеспечивающих некоторые звуковые эффекты.

Управление звуком реализовано при помощи компонента TMediaPlayer. При этом используются такие свойства, как: FileName, Volume, CurrentTime, Play, Duration и др.

Различные случаи использования данных свойств указаны ниже.

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

**begin**

Music.FileName := 'Music\music1.wav';

Music.Volume := 0.6;

Music.Play;

**end**;

В FileName передаётся файл в виде строки, который будет проигрываться во время работы программы. Volume отвечает за громкость проигрывания музыки, принимая значения от “0” до “1”. Play начинает воспроизведение аудиозаписи.

**procedure** TMainF.MusTimerTimer(Sender: TObject);

**begin**

**if** Music.CurrentTime >= Music.Duration **then**

**begin**

Music.CurrentTime := 1;

Music.Play;

**end**;

**end**;

Данный пример показывает принцип зацикливания аудио в программном средстве. Реализуется данный процесс при помощи компонента TTimer, свойств компонента TMediaPlayer, таких как: CurrentTime, Duration. Duration показывает продолжительность аудио в секундах, а CurrentTime - текущую проигрываемую секунду песни.

**procedure** TMainF.btnStopMusicClick(Sender: TObject);

**begin**

**if** Music.CurrentTime > 0 **then**

**begin**

Music.Stop;

Music.CurrentTime := 0;

**end**

**else**

**begin**

Music.Play;

**end**;

**end**;

Указанная выше процедура отвечает за остановку проигрывания музыки во время игры. Здесь единожды используется функция Stop, которая приостанавливает песню. Активируется данная часть посредством нажатия на кнопку “Music”.

2.4 Игровая логика

Работу данной части программы обеспечивает модуль Game.

Для хранения информации о местоположении, размерах шаров было выбрано использование статического массива из элементов типа TCircle.

Balls: **array**[1..100] **of** TCircle;

Ограниченность количества шаров удовлетворяет среднему времени игры, т.к. при закрытии игрового поля, данные об удалённых объектах обновляются, что позволяет использовать их заново.

Отслеживание пуль также производится с помощью статического массива из объектов TCircle.

Bullets: **array**[1..1000] **of** TCircle;

Для отрисовки цифр внутри круга создаётся массив, включающий в себя компоненты TLabel. Для отслеживания попаданий был использован массив из чисел типа SmallInt, ввиду экономии используемой памяти программы и ограничений возможных чисел внутри шара.

Counters: **array**[1..100] **of** SmallInt;

При появлении игрового окна происходит инициализация переменных и свойств элементов. Также устанавливается начальная позиция пушки, обнуляется текущее количество заработанных очков и активируются процедуры, отвечающие за создание целей и выстрелов. Блок-схема появления игрового окна представлена на рисунке 2.4.

При создании игрового поля задаются начальные значения урону от выстрела, скорости полёта снаряда, коэффициенту начисления внутриигровых монет, интервал количества выстрелов, необходимых для уничтожения целей. Блок-схема создания игрового поля находится на рисунке 2.3.



Рисунок 2.3 – Блок-схема создания игрового поля



Рисунок 2.4 – Блок-схема появления игрового окна

Передвижение пушки по игровому полю осуществляется при нажатии стрелок “LEFT” или “RIGHT”. В процедуре происходит проверка на нужные клавиши, проверяется местоположение пушки, после происходит её смещение на некоторые координаты влево или вправо.

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

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

**begin**

**case** Key **of**

37:

**if** Canon.Position.X > -34 **then**

Canon.Position.X := Canon.Position.X - 6;

39:

**if** Canon.Position.X < 250 **then**

Canon.Position.X := Canon.Position.X + 6;

**end**;

**end**;

Стрельба из пушки происходит благодаря компоненту TTimer. Свойство Interval задаёт частоту выстрелов, что натолкнуло на идею создания внутриигрового магазина с улучшениями, о котором будет рассказано чуть позже. В данной процедуре происходит создание объекта типа TCircle, которому присваиваются начальные значения такие, как: Width, Height, Position. Последнее свойство позволило создавать пули, вылетающие прямо из дула пушки. После этого, происходит создание анимации полёта. Устанавливаются такие параметры, как: StartValue, StopValue, Duration, PropertyName. В PropertyName передаётся строка, содержащая название свойства объекта, которое будет изменяться по мере протекания анимации. Duration – время полёта снаряда в секундах, задаётся постоянным и является одной секундой.

**procedure** TGameF.BulletTimer(Sender: TObject);

**begin**

// создание пули

Obj := TCircle.Create(Self);

Obj.Parent := Self;

Obj.Width := 10;

Obj.Height := 10;

Obj.Position.X := Canon.Position.X + 38;

Obj.Position.Y := Canon.Position.Y;

// добавление пули в массив

inc(j);

Bullets[j] := Obj;

// задание анимации движения пули

AnimY := TFloatAnimation.Create(Obj);

AnimY.Parent := Obj;

AnimY.StartValue := Obj.Position.Y;

AnimY.PropertyName := 'Position.Y';

AnimY.StopValue := 0;

AnimY.Duration := 1;

AnimY.Enabled := true;

// удаление пули после окончания анимации

AnimY.OnFinish := Delete;

**end**;

По окончании времени выстрела, происходит удаление пули посредством разрушения объекта, полученного из массива пуль. Данный процесс показан в процедуре Delete.

**procedure** TGameF.Delete(Sender: TObject);

**begin**

**if** Bullets[i] <> nil **then**

**begin**

Bullets[i].Destroy;

Bullets[i] := nil;

**end**

**else**

**if** Bullets[i - 1] <> nil **then**

**begin**

Bullets[i - 1].Destroy;

Bullets[i - 1] := nil;

**end**;

inc(i);

**end**;

Создание целей реализовано при помощи компонента TTimer. В коде программы задаётся объект TCircle. Ввиду усложнения игрового процесса, используется случайное задание данных параметров в определённых рамках, чтобы не создавать слишком сложные цели. Position отвечает за место появления шара, сперва выбирается справа или слева от середины экрана создастся цель, после устанавливается случайная высота относительно уровня пушки. Затем определяется цвет объекта, посредством неопределённого выбора из списка заранее заготовленных.

// создание шара

Obj := TCircle.Create(Self);

Obj.Parent := Self;

Obj.Width := Random(30) + 51;

Obj.Height := Obj.Width;

**case** Rand **of**

0:

Obj.Position.X := 0;

1:

Obj.Position.X := 300 - Obj.Width;

**end**;

Obj.Position.X := 0;

Obj.Position.Y := Random(100);

// выбор цвета шара

**case** Random(5) **of**

0: Obj.Fill.Color := TAlphaColors.Cyan;

1: Obj.Fill.Color := TAlphaColors.White;

2: Obj.Fill.Color := TAlphaColors.Green;

3: Obj.Fill.Color := TAlphaColors.Gold;

4: Obj.Fill.Color := TAlphaColors.Pink;

**end**;

После всего вышеперечисленного происходит инициализация числа выстрелов, которые будут необходимы для уничтожения мишени. Данное значение задаётся случайно в диапазоне переменных от RandFrom до RandTo. Далее данные цифры отрисовываются в середине шара, посредством компонента TLabel, позиция которого определяется через Width и Height цели, на которой он будет находиться. Был выбран чёрный цвет текста, т.к. он более заметен на обозначенных цветах. Используется 20-й размер шрифта в записи.

// добавление числа в шар

Num := TLabel.Create(Self);

Num.Parent := Obj;

Num.Position.X := Trunc(Obj.Width) **div** 2 - 5;

Num.Position.Y := Trunc(Obj.Height) **div** 2 - 10;

Num.TextSettings.Font.Size := 20;

Num.TextSettings.FontColor := TAlphaColors.Black;

**if** TStyledSetting.Size **in** Num.StyledSettings **then**

Num.StyledSettings := Num.StyledSettings -

[TStyledSetting.Size] - [TStyledSetting.FontColor]

**else**

Num.StyledSettings := Num.StyledSettings +

[TStyledSetting.Size] + [TStyledSetting.FontColor];

Num.Text := IntToStr(Random(RandTo) + RandFrom);

Созданная цель заносится в массив мишеней, аналогично для количества выстрелов, необходимых для её уничтожения. Это позволяет отслеживать попадания по определённому шару, оставшиеся попадания, которые игрок должен сделать, уничтожать корректные цели, избегая программных ошибок.

// занесение данных в массивы

inc(jj);

Balls[jj] := Obj;

Counters[jj] := StrToInt(Num.Text);

Numbers[jj] := Num;

Создаётся две анимации передвижения шара: горизонтальная и вертикальная. Выставляются значения таких параметров, как: StartValue, StopValue, Duration, Loop, Autoreverse. Значения для анимаций задаются в зависимости от положения объекта, однако движение по вертикальной оси начинается с места появления цели и оканчивается уровнем пушки. Длительность движения всех шаров одинакова и занимает следующее время: горизонтальная - четыре секунды, вертикальная - две секунды. Также для придания реалистичности падению мишени было использовано свойство Interpolation со значением Quadratic, имитируя квадратичную зависимость ускорения. Loop и Autoreverse имеют постоянное значение True, что зацикливает их передвижение по игровому полю. После создания первой мишени включается компонент TTimer, который выполняет основные задачи игры: проверку на попадание по цели, на попадание шара по пушке.

// анимация движения по горизонтали

AnimX := TFloatAnimation.Create(Obj);

AnimX.Parent := Obj;

AnimX.PropertyName := 'Position.X';

**case** Rand **of**

0:

**begin**

AnimX.StartValue := Obj.Position.X;

AnimX.StopValue := 300 - Obj.Width;

**end**;

1:

**begin**

AnimX.StartValue := 300 - Obj.Width;

AnimX.StopValue := Obj.Position.X;

**end**;

**end**;

AnimX.Duration := 4;

AnimX.**Loop** := true;

AnimX.AutoReverse := true;

AnimX.Enabled := true;

// анимация движения по вертикали

AnimY := TFloatAnimation.Create(Obj);

AnimY.Parent := Obj;

AnimY.StartValue := Obj.Position.Y;

AnimY.PropertyName := 'Position.Y';

AnimY.StopValue := 400 - Obj.Height;

AnimY.Duration := 2;

AnimY.Interpolation := TInterpolationType.Quadratic;

AnimY.**Loop** := true;

AnimY.AutoReverse := true;

AnimY.Enabled := true;

// запуск основного таймера

Check.Enabled := true;

Следующий участок процедуры отвечает за проверку проигрыша игрока. Производится обход по массиву существующих на данный момент целей, и, если координаты мишени совпадают с расположением пушки, стрельба и создание новых шаров прекращаются, проходит очистка игрового поля, а затем появляется сообщение о том, что игра окончена. Ввиду того, что используемый для отображения пушки компонент TImage имеет прямоугольную форму, а объект сам по себе неправильную форму, было использовано уменьшение рамок путём введения числовых констант. Это позволило предотвратить попадание цели по игроку в той зоне, в которой визуально он не находился. После уведомления конца игры пользователь оказывается в главном меню программы, где может потратить заработанные монеты в магазине или снова начать играть. Сообщение о окончании игры показано на рисунке 2.5.

// проверка на проигрыш

**for** m := ii **to** jj **do**

**if** (Balls[m] <> nil)**and**(Balls[m].Position.X +

Balls[m].Width > Canon.Position.X +

20)**and**(Balls[m].Position.X < Canon.Position.X +

Canon.Width - 20) **then**

**if** (Balls[m].Position.Y + Balls[m].Height >

Canon.Position.Y + 15) **then**

**begin**

// отключение стрельбы и появления мишеней

Ball.Enabled := false;

Bullet.Enabled := false;

Check.Enabled := false;

// очистка игрового поля от шаров

**for** l := 1 **to** 100 **do**

**if** Balls[l] <> nil **then**

**begin**

Balls[l].Destroy;

Balls[l] := nil;

**end**;

// сообщение о проигрыше

MessageDlg('You lose! Try again!',

mtInformation, [mbOk], 0);

Close;

MainF.Show;

**end**;

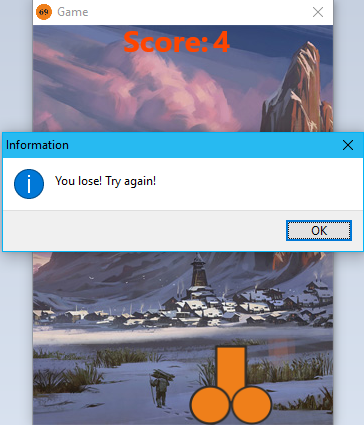


Рисунок 2.5 – Экран окончания игры

// проверка на попадание по цели

**for** k := i **to** j **do**

**for** l := ii **to** jj **do**

**if** (Bullets[k] <> nil)**and**(Balls[l] <> nil) **then**

**if** (Bullets[k].Position.X + 5 >

Balls[l].Position.X)**and**(Bullets[k].Position.X

+ 5 < Balls[l].Position.X + Balls[l].Width) **then**

**if** (Bullets[k].Position.Y + 5 >

Balls[l].Position.Y)**and**(Bullets[k].Position.Y

+ 5 < Balls[l].Position.Y +

Balls[l].Height) **then**

**begin**

// удаление пули и мишени

Delt(Sender);

DeleteBall(Sender);

**end**;

На указанном выше участке кода происходит проверка попадания по цели. Данный процесс осуществляется проходом по массивам всех существующих пуль и шаров в данный момент. При условии, что координаты какого-либо выстрела входят в местоположение одной из целей, происходит удаление определённого снаряда и мишени в процедурах Delt и DeleteBall соответственно. Под координатами выстрела полагается середина пули, что позволяет избежать неправильного отображения попадания в шар. Алгоритм прилагается ниже.

**procedure** TGameF.DeleteBall(Sender: TObject);

**begin**

// проверка на возможность уничтожения цели

**if** Counters[l] <= KoefDmg **then**

**begin**

// уничтожение мишени

Balls[l].Destroy;

Balls[l] := nil;

// увеличение количества очков

inc(Score, Koef \* Counters[l]);

Points.Text := IntToStr(Score);

**end**

**else**

**begin**

// уменьшение числа внутри шара

dec(Counters[l], KoefDmg);

Numbers[l].Text := IntToStr(Counters[l]);

// увеличение количества очков

inc(Score, Koef \* KoefDmg);

Points.Text := IntToStr(Score);

**end**;

**end**;

**procedure** TGameF.Delt(Sender: TObject);

**begin**

// удаление пули

Bullets[k].Destroy;

Bullets[k] := nil;

inc(i);

**end**;

При удалении шара сначала выясняется возможность этого действия. Если число внутри шара меньше урона пушки, происходит уничтожение цели и зачисление очков в таком количестве, что находилось внутри мишени. В ином случае, счётчик внутри шара уменьшается и начисляется столько очков, сколько составляет повреждение от выстрела.

Также здесь упоминается переменная Koef. Данное значение отвечает за множитель очков и монет, которые получает игрок с целью ускорения игрового процесса. Данный коэффициент можно увеличить путём покупки соответствующего улучшения во внутриигровом магазине. По началу игры ему присваивается значение единицы.

Вместе с этим при попадании происходит уничтожение пули, которая поразила цель. Реализована данная возможность при помощи процедуры Destroy у объекта TCircle и присваиванием пустого значения внутри массива.

Номера удаляемых снаряда и мишени получаются из процедуры CheckTimer во время проверки на попадание.

2.5 Внутриигровой магазин

В модуле Shop была реализована возможность ускорения игрового процесса путём покупки специальных улучшений, влияющих на игровой процесс. Среди таких улучшений находятся: увеличение скорости стрельбы, повышение урона от выстрела, поднятие коэффициента умножения очков при переводе их в монеты. Вид окна магазина улучшений представлен ниже:

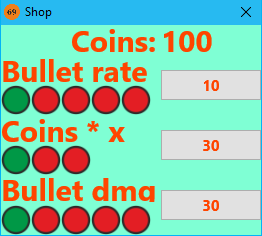


Рисунок 2.6 – Вид окна магазина улучшений

При открытии магазина происходит проверка на возможность покупки каждого улучшения по отдельности. Если монет игрока хватает на покупку улучшения, он сможет его приобрести, в ином случае кнопка выключится и пользователю потребуется заработать больше монет. При покупке улучшения кружок, обозначающий его уровень, загорается зелёным. Это является простым и эффективным способом отображения прокачки определённого умения. Также после приобретения улучшения, повышается его стоимость, если игроку не хватает монет на новый уровень, кнопка покупки заблокируется. При максимальной прокачке одного из умений его стоимость изменяется на строку “Full”, что сигнализирует о достижении наивысшей степени усиления.

В момент запуска программного средства инициализируется стоимость всех апгрейдов, отображается начальный уровень каждого и в соответствии с ним устанавливаются значения переменных, влияющих на игровой процесс. Часть данной процедуры описана ниже:

Bul := 1;

Coin := 1;

Dmg := 1;

BulletUpgr.Text := '10';

CoinsUpgr.Text := '30';

DmgUpgr.Text := '30';

2.6 Внешний вид

Для возможности кастомизации было принято решение добавить изменение модели пушки и заднего фона игрового поля. Смена заднего плана реализуется в модуле Backgrounds. Вид всплывающего окна отображён далее:

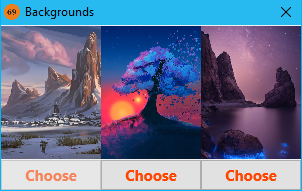


Рисунок 2.7 – Окно изменения заднего плана

При запуске программы автоматически устанавливается первый фон и блокируется его выбор. Пользователь может изменить его путём нажатия кнопки под соответствующей картинкой. При этом выбранная картинка заменит установленную, а кнопка выключится. Процедура изменения фона представлена ниже:

**procedure** TBackgrF.SecChoiceClick(Sender: TObject);

**begin**

SecChoice.Enabled := false;

FirstChoice.Enabled := true;

ThirdChoice.Enabled := true;

GameF.Background.Bitmap.LoadFromFile('Background2.jpg');

**end**;

Изменение внешнего вида пушки выполнено в модуле Skins. Окно выбора изображения следует далее:

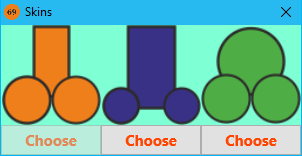


Рисунок 2.8 – Окно выбора пушки

По умолчанию используется первый вариант, однако нажатием на соответствующую кнопку пользователь сможет выбрать понравившийся ему вид и установить его. При этом кнопка выбора отключается, но оставшиеся остаются доступными. Часть кода, отвечающая за данный процесс, находится дальше:

**procedure** TSkinsF.SecChoiceClick(Sender: TObject);

**begin**

SecChoice.Enabled := false;

FirstChoice.Enabled := true;

ThirdChoice.Enabled := true;

GameF.Canon.Bitmap.LoadFromFile('Canon2.png');

**end**;

Всё это позволило дать пользователю возможность изменять внешний вид программы на свой вкус. Пример изменённых изображений пушки и заднего фона показан ниже:

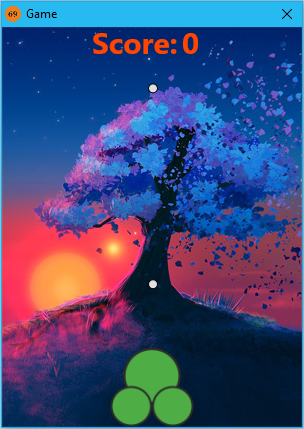


Рисунок 2.9 – Изменённое игровое поле

3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Во время тестирования были обнаружены некоторые проблемы с работой программного средства.

Была найдена проблема с удалением пуль, находящихся выше той, которая поразила цель. При попадании снаряда в шар один из оставшихся на игровом поле не исчезал, а зависал вверху экрана. Данная недоработка была исправлена добавлением дополнительной проверки существования предыдущего снаряда. Исправленный код процедуры приведён ниже:

**if** Bullets[i] <> nil **then**

**begin**

Bullets[i].Destroy;

Bullets[i] := nil;

**end**

**else**

**if** Bullets[i - 1] <> nil **then**

**begin**

Bullets[i - 1].Destroy;

Bullets[i - 1] := nil;

**end**;

Также раскрылась проблема во время проигрыша пользователя. Игровой процесс продолжался, тем самым вызывая дополнительные ошибки доступа и лишние сообщения окончания игры. Изменённый код программы представлен далее:

Ball.Enabled := false;

Bullet.Enabled := false;

Check.Enabled := false;

**for** l := 1 **to** 100 **do**

**if** Balls[l] <> nil **then**

**begin**

Balls[l].Destroy;

Balls[l] := nil;

**end**;

MessageDlg('You lose! Try again!', mtInformation, [mbOk], 0);

Close;

MainF.Show;

При тестировании программного средства обнаружилась проблема отображения чисел внутри целей. Размер и цвет шрифта не применялся к тексту и положение чисел не соответствовало задуманному. Всё это не позволяло добиться нужного внешнего вида игрового поля. В соответствии с этим внеслись некоторые правки, указанные ниже:

Num.Position.X := Trunc(Obj.Width) **div** 2 - 5;

Num.Position.Y := Trunc(Obj.Height) **div** 2 - 10;

Num.TextSettings.Font.Size := 20;

Num.TextSettings.FontColor := TAlphaColors.Black;

**if** TStyledSetting.Size **in** Num.StyledSettings **then**

Num.StyledSettings := Num.StyledSettings -

[TStyledSetting.Size] - [TStyledSetting.FontColor]

**else**

Num.StyledSettings := Num.StyledSettings +

[TStyledSetting.Size] + [TStyledSetting.FontColor];

По окончании проверки работоспособности программы серьёзных недоработок обнаружено не было. В программном средстве были устранены ошибки, допущенные во время его написания. Это позволило исключить появление всевозможных ошибок Windows и загромождению ими экрана во время работы приложения.

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

4.1 Правила игры

Игра заключается в том, чтобы выстрелами сбивать как можно больше шаров. За нанесение урона цели пользователю начисляются очки, которые после игры переводятся в монеты, за которые игрок сможет приобрести специальные улучшения, ускоряющие зарабатывание монет. Однако с покупкой улучшением сложность игры также повышается, увеличивается количество ударов, необходимых для уничтожения мишеней.

Человек управляет пушкой, изображение которой может быть изменено в самой игре в специальной вкладке меню “Skins”. Ниже указан пример одной из моделей:



Рисунок 4.1 – Одна из моделей пушки

Передвижение по полю осуществляется посредством нажатия клавиш со стрелками “Left” и “Right”.

Попадание по шару засчитывается, если выстрел поразил цель. После этого игроку добавляются очки, умноженные на коэффициент зарабатывания монет и урон от пули.

При соприкосновении мишени и пушки игра оканчивается, заработанные очки переводятся в монеты, за которые можно покупать улучшения.

Имеется возможность отключить музыку во время игры путём нажатия на соответствующий значок в правом верхнем углу экрана. При повторном нажатии воспроизведение песни начнётся с её начала.

4.2 Интерфейс программы

Главное окно программы, изображённое на рисунке 4.3, содержит следующие элементы:

– “Play”;

– “Shop”;

– “Backgrounds”;

– “Skins”.

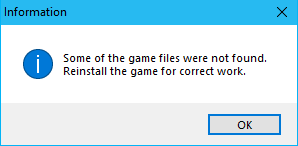
Для начала игры следует выбрать пункт меню “Play”. Позиция пушки задаётся при появлении игрового поля и равна его середине. После отсчёта, равному пяти секундам, шары начинают появляться по обе стороны от игрока с периодом в пять секунд. Внешний вид игрового поля показан на рисунке 4.4.

После окончания игры, пользователь сможет приобрести некоторые улучшения во внутриигровом магазине во вкладке меню “Shop”.

В главном меню программного средства игрок может изменить внешний вид пушки или заднего фона игрового поля. Происходит это по открытию соответствующих пунктов меню “Skins” и “Backgrounds”. Предоставляется по три варианта внешнего вида каждого элемента, что является достаточным для кастомизации игры пользователем.

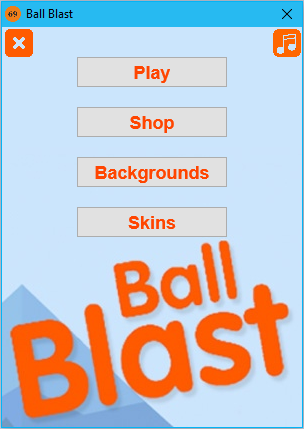
В программном средстве присутствует музыкальное и визуальное сопровождение. Проверка и загрузка ресурсов происходит в момент запуска приложения. При этом поиск файлов осуществляется непосредственно в директории программы.

В случае некорректной загрузки какого-либо из файлов будет выведено соответствующее сообщение, которое представлено на рисунке 4.2, и некоторые эффекты станут недоступны.



4.2 – Сообщение при некорректной загрузке файла

Указанная выше проблема может быть исправлена путем добавления или замены группы ресурсов, соответствующих каждому элементу сопровождения.



4.3 – Главное окно программы



4.4 – Внешний вид игрового поля

ЗАКЛЮЧЕНИЕ

Видеоигры стали значительной частью жизни множества людей. Большинство игр никак не влияет на развитие человека, однако существуют проекты, благодаря которым пользователи развивают логическое мышление, улучшают реакцию, учатся думать наперёд. Это помогает игрокам в их повседневной жизни, помимо виртуальной.

В рамках данного курсового проекта было разработано программное средство “Ball Blast”, которое поможет пользователю скрасить время, проведённое за компьютером. Согласно поставленным задачам, в данном приложении были реализованы следующие возможности:

– смена заднего фона игрового поля;

– смена вида “пушки”;

– встроенный магазин с покупкой улучшений;

– возможность отключения музыки;

– отображение количества заработанных очков и монет;

Для выполнения всех поставленных задач потребовалось тщательно изучить возможности языка Delphi, освоить взаимодействие между компонентами внутри него, научиться основам объектно-ориентированного программирования.

Существует большое количество возможных улучшений приложения. Одним из них является создание новых уровней сложности, возможности дальнейшего ускорения игрового процесса путём добавления новых умений во внутриигровой магазин. Другим вариантом развития является адаптация проекта для запуска на устройствах с низкой разрешающей способностью экрана и переход к кроссплатформенной разработке.

Использование данного программного средства 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. Фараонов, В. В. Delphi 6. Учебный курс: / В. В. Фараонов. – СПб: изд. С. В. Малгачёва, 2001. – 231 с.
2. Программирование на языке Delphi / Д. А. Сурков [и др.]. – учеб. пособие. – Режим доступа: http://www.rsdn.ru/?summary/3165.xml, – 2005.
3. Орлов, С. А. Технологии разработки программного обеспечения: учеб. Пособие. – СПб, 2003.
4. 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.
5. Шупрута, В. В. Delphi 2005. Учимся программировать: / В. В. Шупрута. – Москва: изд. «НТ Пресс», 2001. – 140 с.
6. Уилсон, С. Принципы проектирования и разработки программного обеспечения, yчебн. курс. – СПб, 2003.
7. RAD Studio Product Documentation – Embarcadero Technologies [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://docs.embarcadero.com/products/rad_studio>.
8. Серебряная, Л.В. Марина, И.М. Структуры и алгоритмы обработки данных: учебно-метод. пособие для студ. спец. «Программное обеспечение информационных технологий» всех форм обуч. / Л. В. Серебряная, И. М. Марина. – Минск: БГУИР, 2012. – 49 с;
9. Графические возможности Делфи (электронный ресурс). – Электронные данные. – Режим доступа: <http://delphi7.gym5cheb.ru>;
10. Культин, Н. А. Основы программирования в Delphi 2007 2-е издание – учеб. пособие. – Москва: изд. «НТ Пресс», 2008. – 182с.
11. Хлебостроев, В. С. Программирование графики в Delphi – учеб. пособие. – Москва: изд. «НТ Пресс», 2008. – 243с;

ПРИЛОЖЕНИЕ А

Исходный код программы

**Main.pas**

**unit** Main;

**interface**

**uses**

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

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

FMX.Dialogs, FMX.Menus, FMX.Controls.Presentation, VCL.Dialogs,

FMX.StdCtrls, FMX.Objects, System.ImageList, FMX.ImgList, FMX.Media;

**type**

TMainF = **class**(TForm)

BlankPop: TPopupMenu;

btnExit: TSpeedButton;

Imgs: TImageList;

Style1: TStyleBook;

btnStopMusic: TSpeedButton;

btnPlay: TButton;

btnScore: TButton;

btnBackGr: TButton;

btnSkins: TButton;

Background: TImage;

Music: TMediaPlayer;

MusTimer: TTimer;

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

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

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

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

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

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

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

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

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

**private**

{ Private declarations }

**Public**

{ Public declarations }

**end**;

**var**

MainF: TMainF;

**implementation**

{$R \*.fmx}

**uses** Game, Backgrounds, Skins, Shop;

**procedure** TMainF.btnBackGrClick(Sender: TObject);

**begin**

BackgrF.Showmodal;

**end**;

**procedure** TMainF.btnExitClick(Sender: TObject);

**begin**

Close;

**end**;

**procedure** TMainF.btnPlayClick(Sender: TObject);

**begin**

Visible := false;

**with** GameF **as** TGameF **do**

**begin**

Showmodal;

**end**;

**end**;

**procedure** TMainF.btnScoreClick(Sender: TObject);

**begin**

ShopF.ShowModal;

**end**;

**procedure** TMainF.btnStopMusicClick(Sender: TObject);

**begin**

**if** Music.CurrentTime > 0 **then**

**begin**

Music.Stop;

Music.CurrentTime := 0;

**end**

**else**

**begin**

Music.Play;

**end**;

**end**;

**procedure** TMainF.btnSkinsClick(Sender: TObject);

**begin**

SkinsF.ShowModal;

**end**;

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

**var**

btnSelected: Integer;

**begin**

btnSelected := MessageDlg('Are you sure you want to exit?',

mtConfirmation, [mbYes, mbNo], 0);

**if** btnSelected <> mrYes **then**

CanClose := false;

**end**;

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

**begin**

**if not**(FileExists('Materials\Music.wav'))

**or not**(FileExists('Materials\Exit.png'))

**or not**(FileExists('Materials\OffMusic.png'))

**or not**(FileExists('Materials\MainMenu.jpg'))

**or not**(FileExists('Materials\Background1.jpg'))

**or not**(FileExists('Materials\Background2.jpg'))

**or not**(FileExists('Materials\Background3.jpg'))

**or not**(FileExists('Materials\Canon1.png'))

**or not**(FileExists('Materials\Canon2.png'))

**or not**(FileExists('Materials\Canon3.png'))

**or not**(FileExists('Materials\Btn1.png'))

**or not**(FileExists('Materials\Btn1.png')) **then**

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

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

[mbOk], 0);

Music.FileName := 'Materials\Music.wav';

Music.Volume := 0.6;

Music.Play;

Background.Bitmap.LoadFromFile('Materials\MainMenu.jpg');

Imgs.Source[0].MultiResBitmap[0].Bitmap.

LoadFromFile('Materials\Exit.png');

Imgs.Source[0].MultiResBitmap[0].Scale := 1;

Imgs.Source[1].MultiResBitmap[0].Bitmap.

LoadFromFile('Materials\OffMusic.png');

Imgs.Source[1].MultiResBitmap[0].Scale := 1;

btnExit.ImageIndex := 0;

btnStopMusic.ImageIndex := 1;

**end**;

**procedure** TMainF.MusTimerTimer(Sender: TObject);

**begin**

**if** Music.CurrentTime >= Music.Duration **then**

**begin**

Music.CurrentTime := 1;

Music.Play;

**end**;

**end**;

**end**.

**Game.pas**

**unit** Game;

**interface**

**uses**

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

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

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

FMX.StdCtrls;

**type**

TGameF = **class**(TForm)

Canon: TImage;

Background: TImage;

Bullet: TTimer;

Ball: TTimer;

Check: TTimer;

ScoreLbl: TLabel;

Points: TLabel;

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

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

**procedure** Delete(Sender: Tobject);

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

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

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

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

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

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

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

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

**private**

{ Private declarations }

**public**

Koef, KoefDmg, RandFrom, RandTo: ShortInt;

**end**;

**var**

GameF: TGameF;

Obj: TCircle;

AnimX: TFloatAnimation;

AnimY: TFloatAnimation;

Bullets: **array**[1..1000] **of** TCircle;

Balls: **array**[1..100] **of** TCircle;

Counters: **array**[1..100] **of** SmallInt;

Numbers: **array**[1..100] **of** TLabel;

Score: Integer;

i, j, ii, jj, k, l: Integer;

**implementation**

{$R \*.fmx}

**uses** Main, Backgrounds, Shop;

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

**begin**

**for** i := 1 **to** 100 **do**

**begin**

**if** Balls[i] <> nil **then**

**begin**

Balls[i].Destroy;

Balls[i] := nil;

**end**;

Counters[i] := 0;

**end**;

**for** i := 1 **to** 1000 **do**

**if** Bullets[i] <> nil **then**

**begin**

Bullets[i].Destroy;

Bullets[i] := nil;

**end**;

i := 1;

ii := 1;

j := 0;

jj := 0;

Canon.Position.X := 108;

Canon.Position.Y := 320;

Score := 0;

Points.Text := '0';

Ball.Enabled := true;

Bullet.Enabled := true;

**end**;

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

**begin**

Check.Enabled := false;

Ball.Enabled := false;

Bullet.Enabled := false;

ShopF.Coins.Text := IntToStr(StrToInt(ShopF.Coins.Text) +

StrToInt(Points.Text));

Close;

MainF.Show;

**end**;

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

**begin**

Background.Bitmap.LoadFromFile('Materials\Background1.jpg');

Canon.Bitmap.LoadFromFile('Materials\Canon1.png');

Koef := 1;

KoefDmg := 1;

RandFrom := 2;

RandTo := 5;

Bullet.Interval := 600;

**end**;

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

Shift: TShiftState);

**begin**

**case** Key **of**

37:

**if** Canon.Position.X > -34 **then**

Canon.Position.X := Canon.Position.X - 6;

39:

**if** Canon.Position.X < 250 **then**

Canon.Position.X := Canon.Position.X + 6;

**end**;

**end**;

**procedure** TGameF.BulletTimer(Sender: TObject);

**begin**

// создание пули

Obj := TCircle.Create(Self);

Obj.Parent := Self;

Obj.Width := 10;

Obj.Height := 10;

Obj.Position.X := Canon.Position.X + 38;

Obj.Position.Y := Canon.Position.Y;

// добавление пули в массив

inc(j);

Bullets[j] := Obj;

// задание анимации движения пули

AnimY := TFloatAnimation.Create(Obj);

AnimY.Parent := Obj;

AnimY.StartValue := Obj.Position.Y;

AnimY.PropertyName := 'Position.Y';

AnimY.StopValue := 0;

AnimY.Duration := 1;

AnimY.Enabled := true;

// удаление пули после окончания анимации

AnimY.OnFinish := Delete;

**end**;

**procedure** TGameF.BallTimer(Sender: TObject);

**var**

Num: TLabel;

Rand: SmallInt;

**begin**

Rand := Random(2);

// создание шара

Obj := TCircle.Create(Self);

Obj.Parent := Self;

Obj.Width := Random(30) + 51;

Obj.Height := Obj.Width;

**case** Rand **of**

0:

Obj.Position.X := 0;

1:

Obj.Position.X := 300 - Obj.Width;

**end**;

Obj.Position.X := 0;

Obj.Position.Y := Random(100);

// выбор цвета шара

**case** Random(5) **of**

0: Obj.Fill.Color := TAlphaColors.Cyan;

1: Obj.Fill.Color := TAlphaColors.White;

2: Obj.Fill.Color := TAlphaColors.Green;

3: Obj.Fill.Color := TAlphaColors.Gold;

4: Obj.Fill.Color := TAlphaColors.Pink;

**end**;

// добавление числа в шар

Num := TLabel.Create(Self);

Num.Parent := Obj;

Num.Position.X := Trunc(Obj.Width) **div** 2 - 5;

Num.Position.Y := Trunc(Obj.Height) **div** 2 - 10;

Num.TextSettings.Font.Size := 20;

Num.TextSettings.FontColor := TAlphaColors.Black;

**if** TStyledSetting.Size **in** Num.StyledSettings **then**

Num.StyledSettings := Num.StyledSettings - [TStyledSetting.Size] -

[TStyledSetting.FontColor]

**else**

Num.StyledSettings := Num.StyledSettings + [TStyledSetting.Size] +

[TStyledSetting.FontColor];

Num.Text := IntToStr(Random(RandTo) + RandFrom);

// занесение данных в массивы

inc(jj);

Balls[jj] := Obj;

Counters[jj] := StrToInt(Num.Text);

Numbers[jj] := Num;

// анимация движения по горизонтали

AnimX := TFloatAnimation.Create(Obj);

AnimX.Parent := Obj;

AnimX.PropertyName := 'Position.X';

**case** Rand **of**

0:

**begin**

AnimX.StartValue := Obj.Position.X;

AnimX.StopValue := 300 - Obj.Width;

**end**;

1:

**begin**

AnimX.StartValue := 300 - Obj.Width;

AnimX.StopValue := Obj.Position.X;

**end**;

**end**;

AnimX.Duration := 4;

AnimX.**Loop** := true;

AnimX.AutoReverse := true;

AnimX.Enabled := true;

// анимация движения по вертикали

AnimY := TFloatAnimation.Create(Obj);

AnimY.Parent := Obj;

AnimY.StartValue := Obj.Position.Y;

AnimY.PropertyName := 'Position.Y';

AnimY.StopValue := 400 - Obj.Height;

AnimY.Duration := 2;

AnimY.Interpolation := TInterpolationType.Quadratic;

AnimY.**Loop** := true;

AnimY.AutoReverse := true;

AnimY.Enabled := true;

// запуск основного таймера

Check.Enabled := true;

**end**;

**procedure** TGameF.CheckTimer(Sender: TObject);

**var**

m: Integer;

**begin**

// проверка на проигрыш

**for** m := ii **to** jj **do**

**if** (Balls[m] <> nil)**and**(Balls[m].Position.X + Balls[m].Width >

Canon.Position.X + 20)**and**(Balls[m].Position.X <

Canon.Position.X + Canon.Width - 20) **then**

**if** (Balls[m].Position.Y + Balls[m].Height >

Canon.Position.Y + 15) **then**

**begin**

// отключение стрельбы и появления мишеней

Ball.Enabled := false;

Bullet.Enabled := false;

Check.Enabled := false;

// очистка игрового поля от шаров

**for** l := 1 **to** 100 **do**

**if** Balls[l] <> nil **then**

**begin**

Balls[l].Destroy;

Balls[l] := nil;

**end**;

// сообщение о проигрыше

MessageDlg('You lose! Try again!', mtInformation, [mbOk], 0);

Close;

MainF.Show;

**end**;

// проверка на попадание по цели

**for** k := i **to** j **do**

**for** l := ii **to** jj **do**

**if** (Bullets[k] <> nil)**and**(Balls[l] <> nil) **then**

**if** (Bullets[k].Position.X + 5 > Balls[l].Position.X)

**and**(Bullets[k].Position.X + 5 < Balls[l].Position.X +

Balls[l].Width) **then**

**if** (Bullets[k].Position.Y + 5 > Balls[l].Position.Y)

**and**(Bullets[k].Position.Y + 5 < Balls[l].Position.Y +

Balls[l].Height) **then**

**begin**

// удаление пули и мишени

Delt(Sender);

DeleteBall(Sender);

**end**;

**end**;

**procedure** TGameF.DeleteBall(Sender: TObject);

**begin**

// проверка на возможность уничтожения цели

**if** Counters[l] <= KoefDmg **then**

**begin**

// уничтожение мишени

Balls[l].Destroy;

Balls[l] := nil;

// увеличение количества очков

inc(Score, Koef \* Counters[l]);

Points.Text := IntToStr(Score);

**end**

**else**

**begin**

// уменьшение числа внутри шара

dec(Counters[l], KoefDmg);

Numbers[l].Text := IntToStr(Counters[l]);

// увеличение количества очков

inc(Score, Koef \* KoefDmg);

Points.Text := IntToStr(Score);

**end**;

**end**;

**procedure** TGameF.Delete(Sender: TObject);

**begin**

**if** Bullets[i] <> nil **then**

**begin**

Bullets[i].Destroy;

Bullets[i] := nil;

**end**

**else**

**if** Bullets[i - 1] <> nil **then**

**begin**

Bullets[i - 1].Destroy;

Bullets[i - 1] := nil;

**end**;

inc(i);

**end**;

**procedure** TGameF.Delt(Sender: TObject);

**begin**

// удаление пули

Bullets[k].Destroy;

Bullets[k] := nil;

inc(i);

**end**;

**end**.

**Shop.pas**

**unit** Shop;

**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.Objects;

**type**

TShopF = **class**(TForm)

CoinsLbl: TLabel;

Coins: TLabel;

BullRate: TLabel;

BulletUpgr: TButton;

Bull1: TImage;

Bull2: TImage;

Bull3: TImage;

Bull4: TImage;

Bull5: TImage;

CoinsKoef: TLabel;

CoinsUpgr: TButton;

Coins1: TImage;

Coins2: TImage;

Coins3: TImage;

BullDmg: TLabel;

DmgUpgr: TButton;

Dmg1: TImage;

Dmg2: TImage;

Dmg3: TImage;

Dmg4: TImage;

Dmg5: TImage;

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

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

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

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

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

**private**

{ Private declarations }

**public**

{ Public declarations }

**end**;

**var**

ShopF: TShopF;

Bul, Coin, Dmg: ShortInt;

**implementation**

{$R \*.fmx}

**uses** Game, Main;

**procedure** TShopF.BulletUpgrClick(Sender: TObject);

**begin**

inc(Bul);

**case** Bul **of**

2:

**begin**

GameF.Bullet.Interval := 550;

inc(GameF.RandFrom, 1);

inc(GameF.RandTo, 1);

Bull2.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(BulletUpgr.Text));

BulletUpgr.Text := '15';

**if** StrToInt(Coins.Text) < StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false;

**if** CoinsUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(CoinsUpgr.Text) **then**

CoinsUpgr.Enabled := false;

**if** DmgUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(DmgUpgr.Text) **then**

DmgUpgr.Enabled := false;

**end**;

3:

**begin**

GameF.Bullet.Interval := 500;

inc(GameF.RandFrom, 1);

inc(GameF.RandTo, 1);

Bull3.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(BulletUpgr.Text));

BulletUpgr.Text := '20';

**if** StrToInt(Coins.Text) < StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false;

**if** CoinsUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(CoinsUpgr.Text) **then**

CoinsUpgr.Enabled := false;

**if** DmgUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(DmgUpgr.Text) **then**

DmgUpgr.Enabled := false;

**end**;

4:

**begin**

GameF.Bullet.Interval := 450;

inc(GameF.RandFrom, 1);

inc(GameF.RandTo, 2);

Bull4.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(BulletUpgr.Text));

BulletUpgr.Text := '25';

**if** StrToInt(Coins.Text) < StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false;

**if** CoinsUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(CoinsUpgr.Text) **then**

CoinsUpgr.Enabled := false;

**if** DmgUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(DmgUpgr.Text) **then**

DmgUpgr.Enabled := false;

**end**;

5:

**begin**

GameF.Bullet.Interval := 400;

inc(GameF.RandFrom, 2);

inc(GameF.RandTo, 2);

Bull5.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(BulletUpgr.Text));

BulletUpgr.Text := 'Full';

BulletUpgr.Enabled := false;

**if** CoinsUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(CoinsUpgr.Text) **then**

CoinsUpgr.Enabled := false;

**if** DmgUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(DmgUpgr.Text) **then**

DmgUpgr.Enabled := false;

**end**;

**end**;

**end**;

**procedure** TShopF.CoinsUpgrClick(Sender: TObject);

**begin**

inc(Coin);

**case** Coin **of**

2:

**begin**

GameF.Koef := 2;

inc(GameF.RandFrom, 2);

inc(GameF.RandTo, 3);

Coins2.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(CoinsUpgr.Text));

CoinsUpgr.Text := '40';

**if** StrToInt(Coins.Text) < StrToInt(CoinsUpgr.Text) **then**

CoinsUpgr.Enabled := false;

**if** BulletUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false;

**if** DmgUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(DmgUpgr.Text) **then**

DmgUpgr.Enabled := false;

**end**;

3:

**begin**

GameF.Koef := 3;

inc(GameF.RandFrom, 2);

inc(GameF.RandTo, 4);

Coins3.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(CoinsUpgr.Text));

CoinsUpgr.Text := 'Full';

CoinsUpgr.Enabled := false;

**if** BulletUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false;

**if** DmgUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(DmgUpgr.Text) **then**

DmgUpgr.Enabled := false;

**end**;

**end**;

**end**;

**procedure** TShopF.DmgUpgrClick(Sender: TObject);

**begin**

inc(Dmg);

**case** Dmg **of**

2:

**begin**

GameF.KoefDmg := 2;

inc(GameF.RandFrom, 2);

inc(GameF.RandTo, 3);

Dmg2.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(DmgUpgr.Text));

DmgUpgr.Text := '40';

**if** StrToInt(Coins.Text) < StrToInt(DmgUpgr.Text) **then**

DmgUpgr.Enabled := false;

**if** BulletUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false;

**if** CoinsUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(CoinsUpgr.Text) **then**

CoinsUpgr.Enabled := false;

**end**;

3:

**begin**

GameF.KoefDmg := 3;

inc(GameF.RandFrom, 2);

inc(GameF.RandTo, 4);

Dmg3.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(DmgUpgr.Text));

DmgUpgr.Text := '50';

**if** StrToInt(Coins.Text) < StrToInt(DmgUpgr.Text) **then**

DmgUpgr.Enabled := false;

**if** BulletUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false;

**if** CoinsUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(CoinsUpgr.Text) **then**

CoinsUpgr.Enabled := false;

**end**;

4:

**begin**

GameF.KoefDmg := 4;

inc(GameF.RandFrom, 3);

inc(GameF.RandTo, 5);

Dmg4.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(DmgUpgr.Text));

DmgUpgr.Text := '60';

**if** StrToInt(Coins.Text) < StrToInt(DmgUpgr.Text) **then**

DmgUpgr.Enabled := false;

**if** BulletUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false;

**if** CoinsUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(CoinsUpgr.Text) **then**

CoinsUpgr.Enabled := false;

**end**;

5:

**begin**

GameF.KoefDmg := 5;

inc(GameF.RandFrom, 4);

inc(GameF.RandTo, 7);

Dmg5.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins.Text := IntToStr(StrToInt(Coins.Text) -

StrToInt(DmgUpgr.Text));

DmgUpgr.Text := 'Full';

DmgUpgr.Enabled := false;

**if** BulletUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false;

**if** CoinsUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) < StrToInt(CoinsUpgr.Text) **then**

CoinsUpgr.Enabled := false;

**end**;

**end**;

**end**;

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

**begin**

Bul := 1;

Coin := 1;

Dmg := 1;

BulletUpgr.Text := '10';

CoinsUpgr.Text := '30';

DmgUpgr.Text := '30';

Bull1.Bitmap.LoadFromFile('Materials\Btn2.png');

Bull2.Bitmap.LoadFromFile('Materials\Btn1.png');

Bull3.Bitmap.LoadFromFile('Materials\Btn1.png');

Bull4.Bitmap.LoadFromFile('Materials\Btn1.png');

Bull5.Bitmap.LoadFromFile('Materials\Btn1.png');

Coins1.Bitmap.LoadFromFile('Materials\Btn2.png');

Coins2.Bitmap.LoadFromFile('Materials\Btn1.png');

Coins3.Bitmap.LoadFromFile('Materials\Btn1.png');

Dmg1.Bitmap.LoadFromFile('Materials\Btn2.png');

Dmg2.Bitmap.LoadFromFile('Materials\Btn1.png');

Dmg3.Bitmap.LoadFromFile('Materials\Btn1.png');

Dmg4.Bitmap.LoadFromFile('Materials\Btn1.png');

Dmg5.Bitmap.LoadFromFile('Materials\Btn1.png');

**end**;

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

**begin**

**if** BulletUpgr.Text <> 'Full' **then**

**if** StrToInt(Coins.Text) <= StrToInt(BulletUpgr.Text) **then**

BulletUpgr.Enabled := false

**else**

BulletUpgr.Enabled := true;

**if** CoinsUpgr.Text <> 'Full' **then**

**if** (StrToInt(Coins.Text) <= StrToInt(CoinsUpgr.Text)) **then**

CoinsUpgr.Enabled := false

**else**

CoinsUpgr.Enabled := true;

**if** DmgUpgr.Text <> 'Full' **then**

**if** (StrToInt(Coins.Text) <= StrToInt(DmgUpgr.Text)) **then**

DmgUpgr.Enabled := false

**else**

DmgUpgr.Enabled := true;

**end**;

**end**.

**Backgrounds.pas**

**unit** Backgrounds;

**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.Objects;

**type**

TBackgrF = **class**(TForm)

FirstChoice: TButton;

SecChoice: TButton;

ThirdChoice: TButton;

BackGr1: TImage;

BackGr2: TImage;

BackGr3: TImage;

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

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

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

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

**private**

{ Private declarations }

**public**

{ Public declarations }

**end**;

**var**

BackgrF: TBackgrF;

**implementation**

{$R \*.fmx}

**uses** Game, Main;

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

**begin**

BackGr1.Bitmap.LoadFromFile('Materials\Background1.jpg');

BackGr2.Bitmap.LoadFromFile('Materials\Background2.jpg');

BackGr3.Bitmap.LoadFromFile('Materials\Background3.jpg');

FirstChoice.Enabled := false;

SecChoice.Enabled := true;

ThirdChoice.Enabled := true;

**end**;

**procedure** TBackgrF.FirstChoiceClick(Sender: TObject);

**begin**

FirstChoice.Enabled := false;

SecChoice.Enabled := true;

ThirdChoice.Enabled := true;

GameF.Background.Bitmap.LoadFromFile('Materials\Background1.jpg');

**end**;

**procedure** TBackgrF.SecChoiceClick(Sender: TObject);

**begin**

SecChoice.Enabled := false;

FirstChoice.Enabled := true;

ThirdChoice.Enabled := true;

GameF.Background.Bitmap.LoadFromFile('Materials\Background2.jpg');

**end**;

**procedure** TBackgrF.ThirdChoiceClick(Sender: TObject);

**begin**

ThirdChoice.Enabled := false;

FirstChoice.Enabled := true;

SecChoice.Enabled := true;

GameF.Background.Bitmap.LoadFromFile('Materials\Background3.jpg');

**end**;

**end**.

**Skins.pas**

**unit** Skins;

**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.Objects;

**type**

TSkinsF = **class**(TForm)

FirstChoice: TButton;

SecChoice: TButton;

ThirdChoice: TButton;

Canon1: TImage;

Canon2: TImage;

Canon3: TImage;

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

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

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

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

**private**

{ Private declarations }

**public**

{ Public declarations }

**end**;

**var**

SkinsF: TSkinsF;

**implementation**

{$R \*.fmx}

**uses** Game, Main;

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

**begin**

Canon1.Bitmap.LoadFromFile('Materials\Canon1.png');

Canon2.Bitmap.LoadFromFile('Materials\Canon2.png');

Canon3.Bitmap.LoadFromFile('Materials\Canon3.png');

FirstChoice.Enabled := false;

SecChoice.Enabled := true;

ThirdChoice.Enabled := true;

**end**;

**procedure** TSkinsF.FirstChoiceClick(Sender: TObject);

**begin**

FirstChoice.Enabled := false;

SecChoice.Enabled := true;

ThirdChoice.Enabled := true;

GameF.Canon.Bitmap.LoadFromFile('Materials\Canon1.png');

**end**;

**procedure** TSkinsF.SecChoiceClick(Sender: TObject);

**begin**

SecChoice.Enabled := false;

FirstChoice.Enabled := true;

ThirdChoice.Enabled := true;

GameF.Canon.Bitmap.LoadFromFile('Materials\Canon2.png');

**end**;

**procedure** TSkinsF.ThirdChoiceClick(Sender: TObject);

**begin**

ThirdChoice.Enabled := false;

FirstChoice.Enabled := true;

SecChoice.Enabled := true;

GameF.Canon.Bitmap.LoadFromFile('Materials\Canon3.png');

**end**;

**end**.