Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Филиал

«Минский радиотехнический колледж»

ХРАНИТЕЛЬ ЭКРАНА “ПУЗЫРЬКИ”

Пояснительная записка

к курсовому проекту по дисциплине

« Основы алгоритмизации и программирования»

КП 52493.002101.081 ПЗ

Руководитель (С.А. Апанасевич)

Учащийся гр. 52493 (Н.А. Бандарук)

2017

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

КП 52493.002101.081

Разраб.

Бандарук

Провер.

Апанасевич

Реценз.

Н. Контр.

Утверд.

Хранитель экрана “Пузырьки”

Лит.

Листов

34

МРК

[Введение 4](#_Toc483046440)

[1 Постановка 5](#_Toc483046441)

[1.1 Описание предметной области 5](#_Toc483046442)

[1.2 Обзор существующих решений 5](#_Toc483046443)

[1.3 Входные и выходные данные 6](#_Toc483046444)

[2 Проектирование задачи 7](#_Toc483046445)

[2.1 Диаграмма вариантов использования 7](#_Toc483046446)

[2.2 Разработка алгоритма работы программы 8](#_Toc483046447)

[2.3 Разработка алгоритмов решения задачи 8](#_Toc483046448)

[3 Программная реализация 9](#_Toc483046449)

[3.1 Выбор и обоснование инструментов разработки 9](#_Toc483046450)

[3.2 Структура программы 11](#_Toc483046451)

[3.3 Описание разработанных процедур и функций 11](#_Toc483046452)

[3.4 Назначение и условие применения программы 12](#_Toc483046453)

[4 Системные требования 13](#_Toc483046454)

[4.1 Руководство пользователя 14](#_Toc483046455)

[4.2 Тестирование 18](#_Toc483046456)

[Заключение 20](#_Toc483046457)

[Список использованной литературы 21](#_Toc483046458)

[Приложение](#_Toc483046459) 22

# Введение

Тема проекта – Хранитель экрана “Пузырьки”. Используя средства Borland Delphi 7 необходимо разработать программу, которая будет выполнять анимацию хранителя.

Как только его не называют, заставка, хранитель экрана, скринсейвер (Screensavers), конечно же на ум приходит всем известный разноцветный трубопровод, но почему бы не сделать себе приятно и превратить экран компьютера в реалистичный 3D аквариум с помощью заставки, а может быть больше понравится горный водопад, для практичных людей и тех кто не любит засорять рабочий стол всякими излишествами отлично подойдёт заставка часы.

Не даром заставки называются хранителями экрана, в заставках есть и практический смысл, ведь при постоянной работе даже самые современные мониторы со временем начинают тускнеть и терять четкость светопередачи, не говоря уже о глазах, которые, как все знают, достаточно сильно устают от долгой работы за компьютером.

Год от года скринсейверы становятся ярче, красочнее и это хорошо, ведь приятная глазу анимированная заставка значительно интереснее чем просто черный экран и у нас таких красивых очень много. Хранитель экрана (Screensaver) — функция или отдельная программа гашения экрана при простое компьютера (или иного устройства), призванная снизить непроизводительный износ оборудования и его отдельных частей, а иногда и энергопотребление.

Заставки призваны снизить яркость изображения (в том числе погасить экран полностью) и снизить общий износ, и/или устранить статичность рабочего изображения для снижения локального износа — выгорания люминофора на статических элементах, что актуально для устройств на основе электронно-лучевой трубки и плазменных экранов. Для жидкокристаллических (ЖК) мониторов это не обязательно, так как в ЖК-дисплее практически нет износа (износ заметен только в случае, если изображение на мониторе статично в течение очень длительного срока), а чёрный цвет соответствует наибольшим, а не наименьшим затратам энергии (хотя разница таких затрат очень маленькая). Но на ЖК-дисплеях скринсейверы тоже часто используются в эстетических целях и как дань традиции.

Заставки могут нести энергосберегающие функции (если в процессе работы явно переключают оборудование в соответствующие режимы), а также функции безопасности, если комбинируют в себе блокировку рабочей консоли паролем. Часто заставки делаются в виде анимации.

Однако часто скринсейверы могут являться еще и разносчиками компьютерных вирусов, свидетельствует о том, насколько подобные программы популярны. Например, в январе 2001 года по Сети прошлась эпидемия вирусов СALIFORNIA IBM и GIRL THING. Они пожирали всю информацию на жестком диске, разрушали браузеры. А распространялись вирусы по электронной почте в виде сообщения с приложением файла со скринсейвером (расширение .scr).

Если вам хотя бы раз в жизни надоедали штатные Windows-заставки, пора поинтересоваться программными продуктами, с помощью которых без особых усилий и не набирая ни одной строчки кода, можно создать собственный оригинальный скринсейвер.

# 1 Постановка

## 1.1 Описание предметной области

Тема проекта – Хранитель экрана “Пузырьки”. Используя средства Borland Delphi 7 необходимо разработать программу, которая будет выполнять анимацию хранителя. Типовая структура интерфейса включает ряд элементов:

* Рабочее поле — это всё пространство на экране дисплея на котором выполняется анимация.
* Меню настроек – это меню в котором можно настроить Хранитель экрана по различным параметрам.

## 1.2 Обзор существующих решений

В настоящее время существует огромное количество Хранителей экрана. В начале 21 века Хранители получили большую популярность (причины описаны выше), из-за этого есть огромное количество различных Хранителей. Даже по умолчанию в любой версии Windows (начиная с XP) есть Хранители Экрана. Поэтому чаще всего пользователь выбирает именно данные Хранители экрана.

Обзор существующих решений:

* экранная заставка “Геометрический вальс”;
* экранная заставка “Ленты”;
* экранная заставка “Мыльные пузыри”;
* экранная заставка “Объёмный текст”;
* экранная заставка “Пустой экран”;
* экранная заставка “Фотографии”.

Перечисленные выше заставки установлены по умолчанию в Windows. Далее будут перечислены самые известные заставки, находящиеся в свободном доступе в сети интернет.

* экранная заставка “Clock Grass ScreenSaver”;
* экранная заставка “White Christmas 3d”;
* экранная заставка “Box Clock”;
* экранная заставка “Коралловые часы”;
* экранная заставка “Sandy Beach”.

Критика:

В Хранителях экрана от кампании Microsoft относительно мало настроек, также как и в остальных.

## 1.3 Входные и выходные данные

С первым запуском Хранителя экрана создаётся файл (.ini), в который заносятся входные данные по умолчанию. Следовательно, входные данные – данные настройки Хранителя, чтобы их изменить нужно после запуска Хранителя нажать комбинацию клавиш: «Ctrl + Tab» и изменить данные вручную, либо через кнопку «Параметры» в меню настройки хранителя экрана Windows. Пункты настроек:

* цвет фона (BGColor);
* цвет рамки (PenColor);
* цвет заливки (BrushColor);
* текстура (TextureNumber);
* сheckBox вкл./выкл. рамку;
* толщина рамки (PenWidth);
* radioGroup с включением квадратиков (circles);
* radioGroup с выбором фона (screenshot);
* скорость (speed);

Выходными данными является анимация полета пузырьков и при изменении настроек и их сохранении это переменные настроек, сохраняемые в файл (.ini) (эта функция выполняется во время работы программы). Также, если использовать программу, как хранитель то можно изменить настройки в интервале запуска хранителя в самой ОС, это также является входными данными. Вот такими способами можно изменить заставку в Windows. Все методы, описанные выше используют стандартные возможности ОС без сторонних программ, и отличается от процесса изменения стартовой заставки на Windows, с которым рекомендую ознакомиться. Для того чтобы отключить экранную заставку, в выпадающем списке "Заставка" выберите "(Нет)", после чего нажмите на кнопку "ОК", а чтобы включить заставку, нужно в том же выпадающем списке выбрать любую имеющуюся заставку и нажать на кнопку "ОК". Также диалог параметров экранных заставок позволяет просмотреть, как будет выглядеть заставка еще до того, как пройдет указанный промежуток времени, в течение которого пользователь не выполняет никаких действий. Для этого достаточно нажать на кнопку "Просмотр" текущего диалога. Чтобы закончить просмотр экранной заставки просто подвигайте мышью или нажмите на любую клавишу на клавиатуре. Также диалог параметров экранных заставок позволяет просмотреть, как будет выглядеть заставка еще до того, как пройдет указанный промежуток времени, в течение которого пользователь не выполняет никаких действий. Для этого достаточно нажать на кнопку "Просмотр" текущего диалога. Чтобы закончить просмотр экранной заставки просто подвигайте мышью или нажмите на любую клавишу на клавиатуре. Функции энергосбережения, могут настраиваться здесь же при необходимости. Если необходимости нет, отключите их. Данная процедура интуитивно понятна и подробных объяснений не требует. Спорный момент — Почему стоит и не стоит отключить систему энергосбережения? Разве не надо беречь электроэнергию?

Во-первых, еще не было случаев, чтобы система энергосбережения помогла бы работе какой-нибудь программы, а сбои в программах из-за ее неожиданного вмешательства отнюдь не редкость.

# 2 Проектирование задачи

## 2.1 Диаграмма вариантов использования

Оператор

Запуск Хранителя

Пользовательские настройки из файла

Анимация

Сохранить в файл

Диаграмма вариантов использования программы

Стандартные настройки

Изменить настройки

После запуска Хранителя (автономно или вручную) идёт задача настроек и переменных. Как видно на диаграмме есть 2 возможных варианта. Программа проверяет на наличие файла настроек, если файл есть читает его, если нет создает его и читает. Первый запуск происходит только с стандартными настройками (если вручную не задать в файле корректные настройки). После вышеперечисленных действий запускается анимация. В процессе выполнения анимация можно вызвать меню настроек с различными пунктами настроек соответственно (смотреть пункт 1.3). Программа завершается по нажатию на клавиатуре клавиши или нажатию по кнопке мыши или её движению.

## 

## 2.2 Разработка алгоритма работы программы

Создание Хранителя экрана “Пузырьки” делится на три части. Первая - создание алгоритма хранителя. Вторая - создание анимации “Пузырьки”. Третья – создание настроек.

Создание анимации “Пузырьки”. Для реализации данной задачи используются компоненты timer и shape с помощи которых и выполняется анимация (изменение действия с определённым интервалом). Далее будет описан алгоритм анимации “Пузырьки”. При запуске хранителя экрана берутся параметры анимации из файла (какие переменные описано в пункте 1.3) + форме задаются размеры экрана (форма увеличивается на весь экран), далее происходит выключение курсора. Потом выделяется память на массивы шариков и включается двойная буферизация. Если включен параметр “прозрачный” фон, то делается скриншот при создании формы. Далее применяются настройки взятые из файла (.ini). Когда все готова для запуска анимации, запускается алгоритм, который расставляет “шарики” (shape) по форме. Чтоб “шарики” не появились друг на друге, для каждого шарика происходит проверка и сравнение с позицией других шариков. После этого случайно задается скорость от -3 до 6 (скорость – расстояние, проходимое за один “тик” таймера). Теперь всё подготовлено для выполнения алгоритма анимации. За каждый “тик” таймера “шарик” перемещается по X и по Y на расстояние, которое задано параметром скорости (speed). Далее происходят проверки на столкновения со стенками и столкновения друг с другом. Если происходит столкновение со стенкой, то шарик меняет направление и скорость на противоположное. А если происходит столкновение шарика с шариком, то шарик меняет направление и скорость в зависимости направления и скорости шарика, с который он сталкивается. Алгоритм проверки на столкновение друг с другом таков: сначала происходит вычисление расстояния между двумя шариками, потом проверка происходит ли столкновение, если происходит, то вычисляем угол полета и меняем углы шариков после столкновения на противоположные. В результате данного алгоритма происходит иллюзия полета пузырьков по экрану. Пузырьки выглядят, как настоящие потому что при столкновениях соблюдаются геометрические и физические законы.

## 2.3 Разработка алгоритмов решения задачи

Для облегчения понимания написанной программы большая часть алгоритма была разбита на процедуры под алгоритмы.

Unit1 – отвечает за Form1 (форму), главная часть программы: выполнение анимации, создание и проверка файла, считывание файла, присвоение настроек и т.д. происходит в первом Unit.

Unit2 – отвечает за Form2 (форму), часть программы отвечающая за меню настроек (категории настроек описаны в пункте 1.3). Так же на второй форма есть описание программы.

# 3 Программная реализация

## 3.1 Выбор и обоснование инструментов разработки

КП написан на языке программирования Delphi, который поддерживает объектно-ориентированные конструкции. Среда разработки – Borland Delphi 7.

Borland Delphi 7, выпущенная в августе 2002 года, стала стандартом для многих разработчиков Delphi.

Используя среду разработки Borland Delphi 7 можно быстро и эффективно создавать, как простые, так и сложные программы. Borland Delphi 7 - среда предназначена для быстрой (RAD) разработки прикладного ПО для операционных систем Windows. Благодаря уникальной совокупности простоты языка и генерации машинного кода, позволяет непосредственно, и, при желании, достаточно низкоуровневого взаимодействия с операционной системой, а также с библиотеками, написанными на C/C++. Выделение и освобождение памяти контролируется в основном пользовательским кодом, что, с одной стороны, ужесточает требования к качеству кода, а с другой — делает возможным создание сложных приложений с высокими требованиями к отзывчивости (работа в реальном времени).

Система программирования Delphi версии 7 фирмы Enterprise (Borland) предоставляет наиболее широкие возможности для программирования приложений ОС Windows.

Delphi – это продукт Borland International для быстрого создания приложений. Процесс создания интерфейса будущей программы напоминает забаву с игровым компьютерным конструктором. Поэтому RAD-среды еще называют визуальными средами разработки: какими мы видим рабочие и диалоговые окна программы при проектировании, такими они и будут, когда программа заработает.

Высокопроизводительный инструмент визуального построения приложений включает в себя настоящий компилятор кода и предоставляет средства визуального программирования, несколько похожие на те, что можно обнаружить в Microsoft Visual Basic (она не является RAD-системой) или в других инструментах визуального проектирования. В основе Delphi лежит язык Delphi, который является расширением объектно-ориентированного языка Pascal. В Delphi также входят локальный SQL-сервер, генераторы отчетов, библиотеки визуальных компонентов, и прочее, необходимое для того, чтобы чувствовать себя совершенно уверенным при профессиональной разработке информационных систем или просто программ для Windows-среды.

Прежде всего Delphi предназначен для профессиональных разработчиков, желающих очень быстро разрабатывать приложения в архитектуре клиент-сервер. Delphi производит небольшие по размерам высокоэффективные исполняемые модули (.exe и .dll), поэтому в Delphi должны быть, прежде всего, заинтересованы те, кто разрабатывает продукты на продажу. С другой стороны, небольшие по размерам и быстро исполняемые модули означают, что требования к клиентским рабочим местам существенно снижаются – это имеет немаловажное значение и для конечных пользователей.

Преимущества Delphi по сравнению с аналогичными программными продуктами:

* быстрота разработки приложения (RAD);
* высокая производительность разработанного приложения;
* низкие требования разработанного приложения к ресурсам компьютера;
* наращиваемость за счет встраивания новых компонент и инструментов в среду Delphi;
* Возможность разработки новых компонентов и инструментов собственными средствами Delphi (существующие компоненты и инструменты доступны в исходных кодах);

– удачная проработка иерархии объектов.

Система программирования Delphi рассчитана на программирование различных приложений и предоставляет большое количество компонентов для этого. К тому же работодателей интересует, прежде всего, скорость и качество создания программ, а эти характеристики может обеспечить только среда визуального проектирования, способная взять на себя значительные объемы рутинной работы по подготовке приложений, а также согласовать деятельность группы постановщиков, кодировщиков, тестеров и технических писателей. Возможности Delphi полностью отвечают подобным требованиям и подходят для создания систем любой сложности.

Основным конкурентом Borland Delphi 7 является её родной брат – RAD-среда Borland C++ Builder, технология работы с которой полностью совпадает с технологией, принятой в Delphi 7. Только в Delphi программный код пишется на языке программирования Паскаль, точнее на его объектно-ориентированной версии Delphi, а не на языке C++.

Для того чтобы обосновать, почему наш выбор остановился на Borland Delphi 7, достаточно просто перечислить некоторые недостатки языка С++ по сравнению с Delphi:

* + Надо делать много инициализации (регистрировать класс окна, организовывать цикл обработки сообщений, создавать оконную функцию, пиктограмму и прочее…) и частично быть системным программистом. На Delphi-же системное программирование уже встроено и инициализация работает по умолчанию, поэтому программист главный упор делает на своих алгоритмах, а не на организации вспомогательных работ.
  + Значительно большая, по сравнению с Delphi, сложность языка, даже, несмотря на компактность кода, возникают сложности в его восприятии.
  + Одна особенность, на мой взгляд, языка С++ очень портит этот язык - он чувствителен к регистру символов, т.е. переменная A и переменная a - это разные переменные.
  + В Delphi классы (объекты) могут располагаться только в динамической памяти, а в C++ в любой памяти (статическая, стек, динамическая). Это добавляет безопасности программирования в Delphi (а вообще мы просто изучаем Delphi, если бы мне дали выбрать язык изучения, я точно бы не выбрал Delphi).

Также существует среда программирования Lazarus, относительно молодая, внешне похожая на Delphi. Данный продукт - IDE для компилятора FreePascal Compiler. Распространяется бесплатно по GNU General Public License (или просто GPL), но Lazarus ещё не является средой программирования профессионального уровня, для него разработано мало компонентов, при стандартных настройках. Также размеры разрабатываемых приложений тоже оставляют желать лучшего. В первую очередь это связано с особенностью компилятора FreePascal, который не дружит с динамическими библиотеками. А потому должен включать в себя все используемые пакеты. Тоже самое касается и собственно среды разработки, которую вы должны пересобрать каждый раз при добавлении нового пакета.

Компиляция проекта в IDE Lazarus, как и во всех средах разработки подразделяется на два этапа: компиляция и сборка. Хотя они и реализованы в виде вызова компилятора FreePascal отдельным процессом, и мы не можем построчно (как в Delphi) наблюдать за компиляцией проекта.

## 3.2 Структура программы

Программа состоит из двух форм: главная (первая), настройки (вторая). На главной форме находятся следующие элементы: таймер, 10 элементов Shape, компонент Image. На главной форме реализуется анимация и с второй формой она взаимодействует строго после её вызова (нажатием на комбинацию клавиш “Ctrl + Tab”). На форме с настройками находятся 3 страницы: “Основные”, “Дополнительные” и “Об авторе”. На странице “Основные” находятся 3 ColorBox’а, которые отвечают за изменение цветов заднего фона, рамки и самого шарика. Так же на первой странице есть компонент ComboBox, который отвечает за изменение текстуры шариков. Еще есть CheckBox, который включает и отключает рамку. И последний компонент это Edit с регулирующим UpDown с помощью которого можно задавать толщину рамки. На второй странице настроек находится Edit c UpDown с помощью которого можно выбирать количество шариков. Далее на вкладке “Дополнительно” находятся 2 компонента RadioGroup, с помощь которых можно выбрать что будет летать, шарики или квадратики и можно выбрать отображение заднего фона. Снизу находится компонент TrackBar, помощью которого можно задать скорость полета пузырьков. На третьей странице находится описание программы. Изменения применяются после нажатия на кнопку “Применить”. Форма с настройками зависит от главной, так как в результате настроек изменяются переменные (заданы глобально) находящиеся в главной форме, также реализована функция сохранить настройки в файл и запустить с файла, то есть все настройки сохраняются в файл и при переходе на первую форму используются из этого файла.

Данная программа представляет собой анимированную экранную заставку, предназначенную для снижения электропотребления, снижения общего износа и замены статического рабочего окна во время простоя компьютера.

## 3.3 Описание разработанных процедур и функций

Процедуры и функции Unit1:

procedure setBallsSquares;

Процедура setBallsSquares делает из щариков квадратики.

procedure setBallsCircles;

Процедура setBalls Circles делает из квадратиов шарики.

Процедуры setDiagonalLinesLeft, setDiagonalLinesRight, setCrossLines, setDiagonalCrossLines, setHorizontalLines, setVerticalLines, setTextureNone;

Данные процедуры отвечают за изменение текстуры шариков.

procedure createScreenShot;

Процедура createScreenShot отвечает за создание скриншота, если он включен в параметрах.

procedure ballPositionSet;

Процедура ballPositionSet расставляет шарики по форме.

procedure ballsPos;

Процедура ballsPos перемещает шарики по форме.

procedure bumpWalls;

Процедура bumpWalls делает проверки на столкновение шариков со стенками, в случает столкновения шарик меняет направление полета и скорость на противоположную.

procedure bumpEachOther;

Процедура bumpEachOther делает проверки на столкновение шариков друг с другом, в случае столкновения шарик меняет направление полета и скорость на противоположную.

procedure delBallPen;

Процедура delBallPen удаляет рамку у пузырьков.

Procedure BallsFly;

Процедура BallsFly вызывает процедуры отвечающие за шаг и проверки на столкновение в таймере.

Процедуры и функции Unit2:

procedure toDefault;

Процедура toDefault сбрасывает настройки к стандартным.

procedure speedChange;

Процедура speedChange изменяет скорость полета шариков.

procedure ballCountSet;

Процедура ballCountSet изменяет количество шариков.

## 3.4 Назначение и условие применения программы

Скринсейверы реагируют на любое сиюминутное явление - будь то выход фильма, альбома любимой песни и даже личного события. Более того, экранные заставки уже давно перестали быть просто красивыми движущимися картинками и выполняют самые разнообразные функции. Это и промоушн исполнителей и фильмов, и реклама, и даже проекты по распределенным вычислениям - например, научный эксперимент SETI@home.

Общество поиска внеземного разума (Search for Extraterrestrial Intelligence, SETI) ведет поиск сигналов от инопланетных цивилизаций. Уже сейчас получаются и анализируются огромные массивы данных с помощью самого большого в мире радиотелескопа и 40-процессорного суперкомпьютера, которые находятся в пуэрториканкском городе Аресибо. Но даже этот суперкомпьютер, выполняющий 200 миллиардов операций в секунду, не в состоянии обрабатывать поток данных, получаемых при помощи телескопа. Поэтому был создан проект SETI@home, который позволил объединить вычислительную мощь миллионов пользователей Интернета, чтобы помочь ученым "прочесать" небеса в поисках следов разумной жизни. На компьютер пользователя устанавливается "красивый скринсейвер, который ищет инопланетян", то есть программа, получающая свою порцию распределенных вычислений от центрального компьютера и после выполнения отправляющая данные обратно. А тот факт, что скринсейверы могут являться еще и разносчиками компьютерных вирусов, свидетельствует о том, насколько подобные программы популярны. Например, в январе 2001 года по Сети прошлась эпидемия вирусов СALIFORNIA IBM и GIRL THING. Они пожирали всю информацию на жестком диске, разрушали браузеры. А распространялись вирусы по электронной почте в виде сообщения с приложением файла со скринсейвером (расширение .scr).

Заставки призваны снизить яркость изображения (в том числе погасить экран полностью) и снизить общий износ, и/или устранить статичность рабочего изображения для снижения локального износа — выгорания люминофора на статических элементах, что актуально для устройств на основе электронно-лучевой трубки и плазменных экранов. Для жидкокристаллических (ЖК) мониторов это не обязательно, так как в ЖК-дисплее практически нет износа (износ заметен только в случае, если изображение на мониторе статично в течение очень длительного срока), а чёрный цвет соответствует наибольшим, а не наименьшим затратам энергии (хотя разница таких затрат очень маленькая). Но на ЖК-дисплеях скринсейверы тоже часто используются в эстетических целях и как дань традиции.

Несут энергосберегающие функции (если в процессе работы явно переключают оборудование в соответствующие режимы), а также функции безопасности, если комбинируют в себе блокировку рабочей консоли паролем. Часто заставки делаются в виде анимации.

Система программирования Delphi рассчитана на программирование различных приложений и предоставляет большое количество компонентов для этого. К тому же работодателей интересует, прежде всего, скорость и качество создания программ, а эти характеристики может обеспечить только среда визуального проектирования, способная взять на себя значительные объемы рутинной работы по подготовке приложений, а также согласовать деятельность группы постановщиков, кодировщиков, тестеров и технических писателей. Возможности Delphi полностью отвечают подобным требованиям и подходят для создания систем любой сложности.

Заставки призваны снизить яркость изображения (в том числе погасить экран полностью) и снизить общий износ, и/или устранить статичность рабочего изображения для снижения локального износа — выгорания люминофора на статических элементах, что актуально для устройств на основе электронно-лучевой трубки и плазменных экранов. Для жидкокристаллических (ЖК) мониторов это не обязательно, так как в ЖК-дисплее практически нет износа (износ заметен только в случае, если изображение на мониторе статично в течение очень длительного срока), а чёрный цвет соответствует наибольшим, а не наименьшим затратам энергии (хотя разница таких затрат очень маленькая). Но на ЖК-дисплеях скринсейверы тоже часто используются в эстетических целях и как дань традиции.

Заставки могут нести энергосберегающие функции (если в процессе работы явно переключают оборудование в соответствующие режимы), а также функции безопасности, если комбинируют в себе блокировку рабочей консоли паролем. Часто заставки делаются в виде анимации.

# 4 Руководство пользователя

## 4.1 Системные требования

Хранитель экрана “Пузырьки”, имеет довольно маленький размер, он быстро загружается, при этом он потребляет минимум ресурсов компьютера. Требования к операционной системе тоже минимальны. Она должна быть семейства Windows и не ниже версии XP. Для воспроизведения программы на мониторе компьютера достаточно встроенного в процессор графического чипа. Все минимальные системные требования для комфортной работы с программой приведены в таблице 4.1.

Таблица 4.1 – Системные требования Хранителя экрана “Пузырьки”

|  |  |
| --- | --- |
| Компонент | Требование |
| ПК и процессор | Процессор с тактовой частотой 500 МГц или выше; |
| Память | Не менее 256 МБ оперативной памяти; для работы с графикой и использования мгновенного поиска, |
| Жесткий диск | 2 МБ свободного дискового пространства. |
| Монитор | Требования к видеоадаптеру: построитель текстуры 20 и вершинный построитель текстуры 2.0. Выпуск драйверов не ранее 1 ноября 2004 г. |
| Операционная система | Windows XP и выше. |
| Дополнительно | Требования и функциональные возможности продуктов могут различаться в зависимости от конфигурации системы и используемой операционной системы. |

При этом Хранитель экрана не работает на операционных системах семейства Unix и Linux. Так как среда разработки Borland Delphi 7 не поддерживает эти операционные системы, однако, программу можно запустить через популярное приложение Wine, которое эмулирует работу Windows.

Всего существует восемь версий программы. Восьмая версия программы является окончательной версией и обновляться не будет (скорее всего). Предыдущие версии программы полностью рабочие и не имеют ошибок, но в них меньше функционала. Программа не используется в коммерческих целях. И при необходимости вы можете получить исходный код.

Руководство пользователя делится на две части: первая – установка Хранителя, вторая – про анимацию и её настройку. Если вам не надо использовать программу в качестве хранителя экрана – переходите ко второму пункту.

Для работы с программой можно сразу запустить .scr файл (не важно где он находится), сразу запуститься программа-анимация…

Для работы в качестве Хранителя экрана требуется поместить данный файл по пути C:\Windows\System32 → зайти в настройки Операционной системы, параметры Windows (Рисунок 4.1.1)

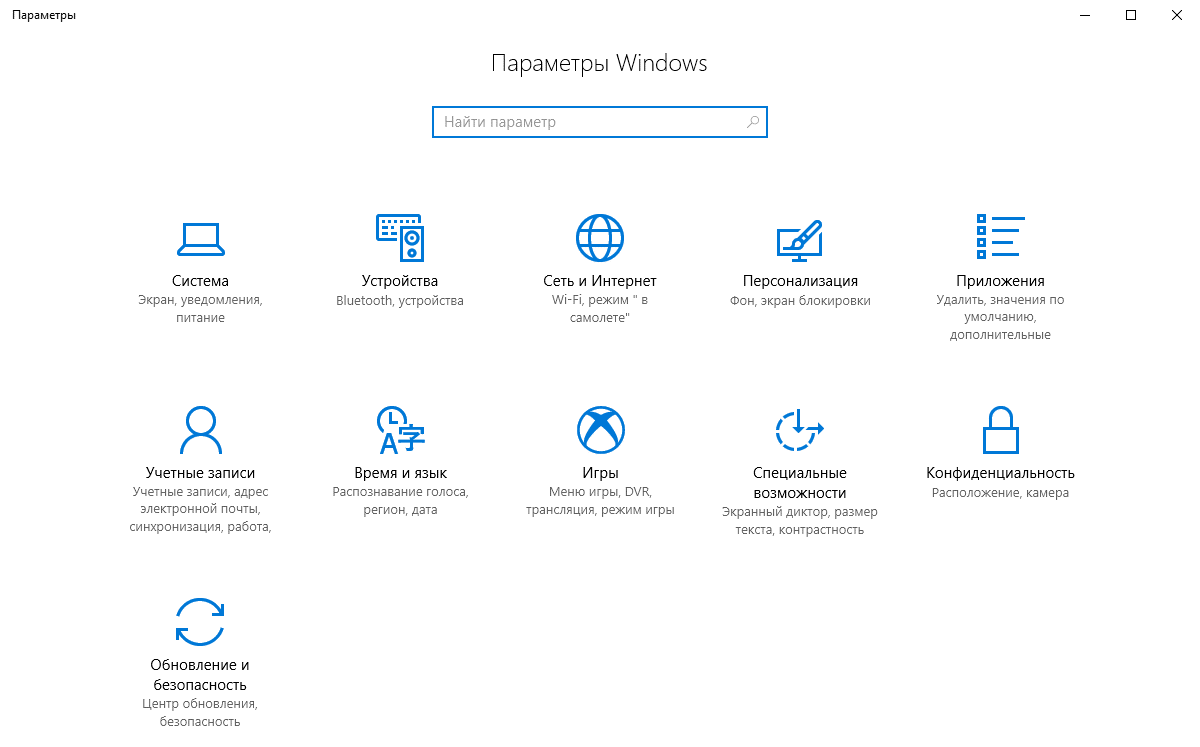


Рисунок 4.1.1 – Параметры Windows

Далее перейти в раздел “Персонализация” и выбрать раздел Экран блокировки

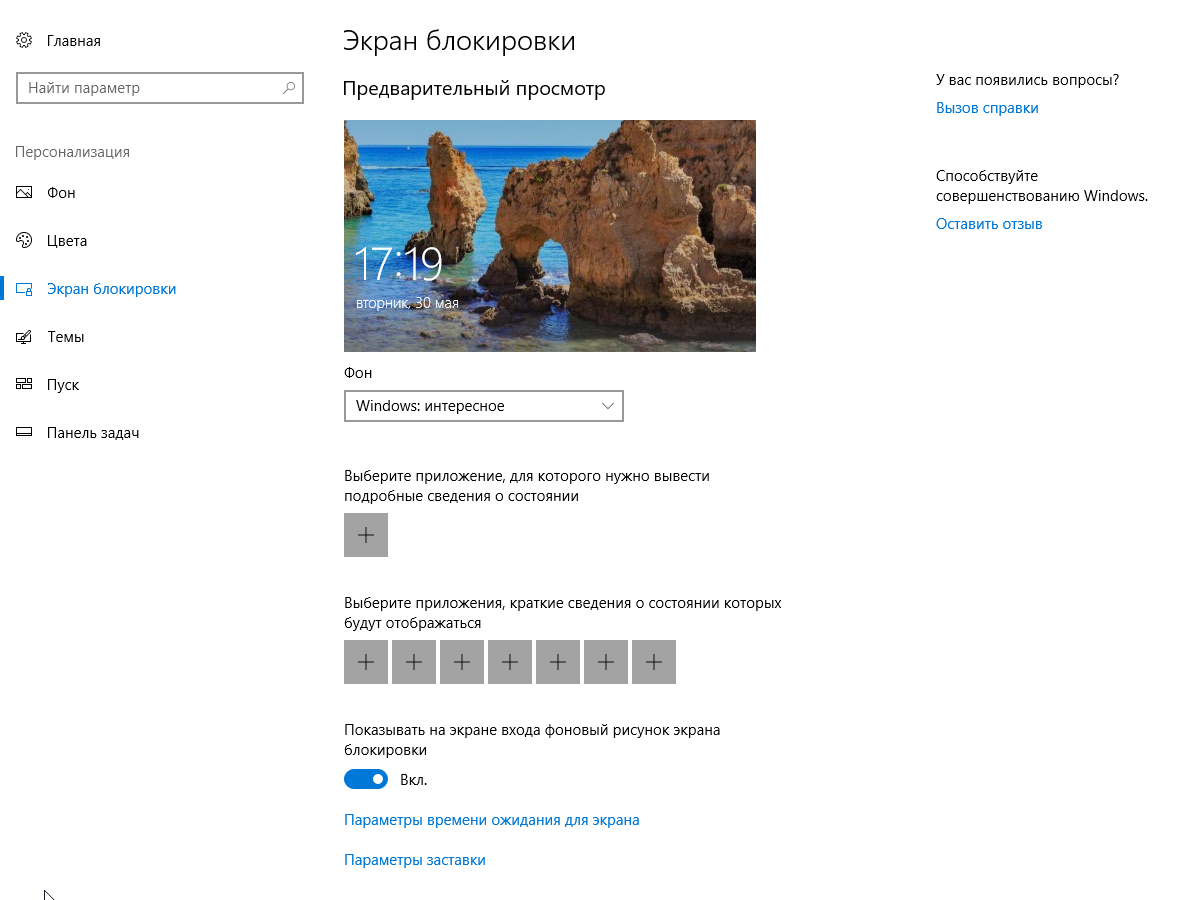


Рисунок 4.1.2 – Экран блокировки

И перейти в пункт “Параметры заставки” (Рисунок 4.1.3)

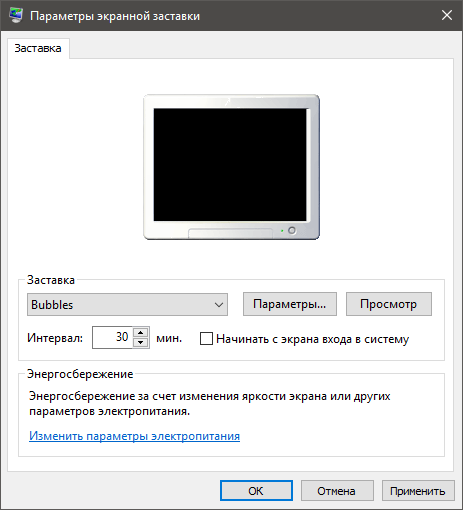


Рисунок 4.1.3 – Параметры оконной заставки

Выбрать из списка хранителей “Bubbles” → настроить интервал включения при бездействии → нажать кнопку “применить” → нажать кнопку “Oк”.

Для добавления хранителя в исключения антивирусной программы, выполните следующие шаги (для примера будет описано добавление в исключение файла в антивирусе “Avast”):

1. Открыть программу-антивирус.
2. Перейти в раздел настройки.
3. Во вкладке общие найти подпункт “Исключения”
4. Нажать на кнопку “Обзор”
5. Выбрать файл хранителя, “Bubbles.scr”.
6. Нажать на кнопку “Добавить”.

Далее идёт вторая часть руководства.

После первого запуска программы автономно, либо после нажатия кнопки “Просмотр” в меню выбора заставки Windows сразу же запускается анимация (Рисунок 4.1). Примечание 1: если вы установили программу, как хранитель экрана вы можете в любое время изменять настройки хранителя либо выбрав в параметрах экранной заставки пункт “Параметры” и настроить программу в соответствии с критериями, либо при простом запуске также нажать комбинацию клавиш “Ctrl + Tab”.

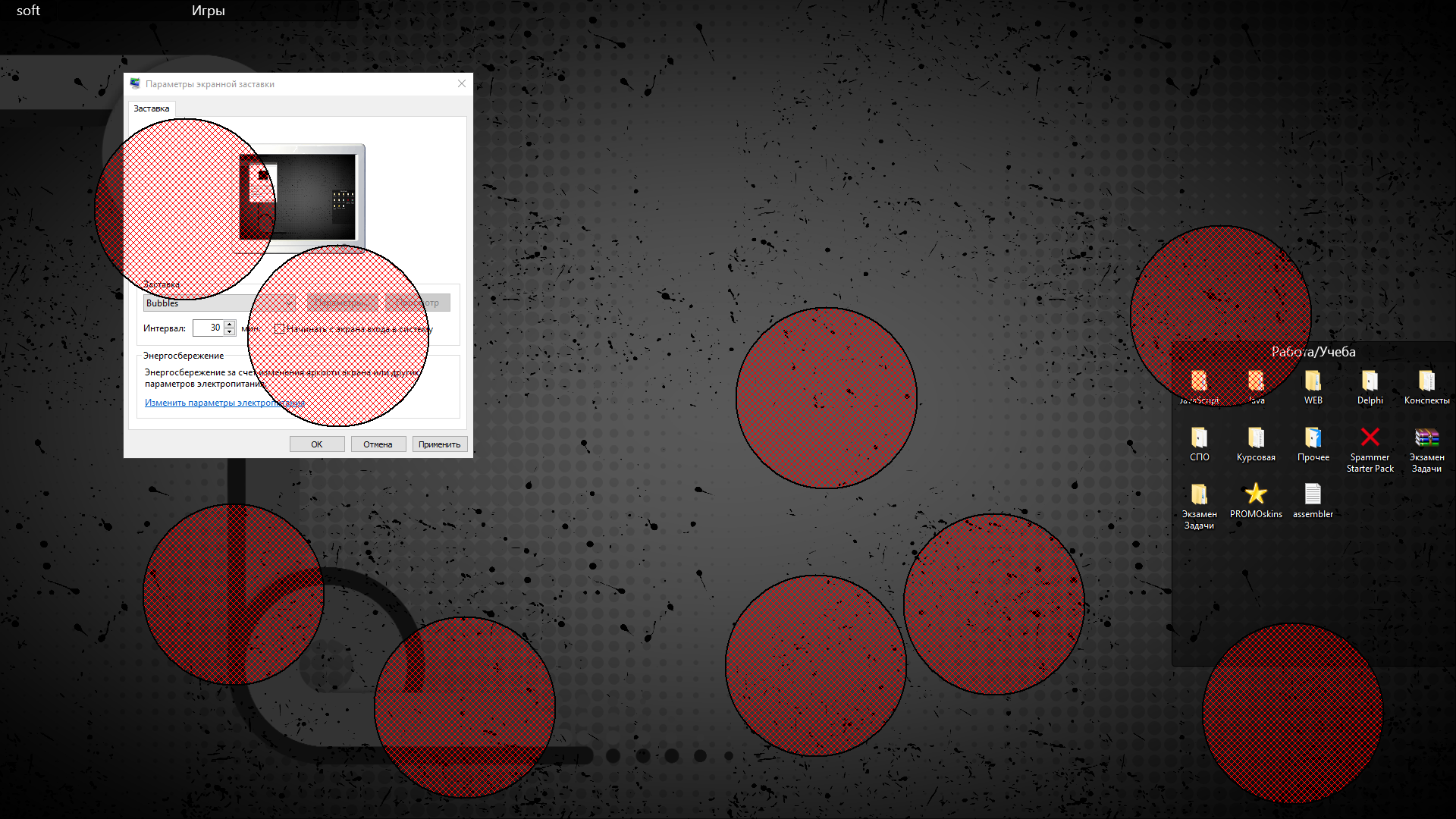
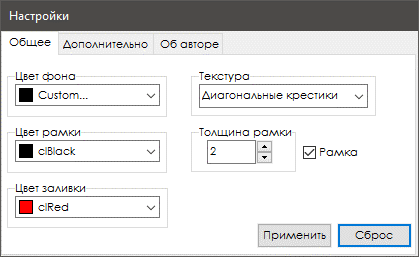


Рисунок 4.1.4 – Главная форма программы

Если вам нравится всё без изменений, то можете всё оставить на этом этапе.

Для открытия меню настроек нажмите комбинацию клавиш “Ctrl + Tab” (Рисунок 4.1.2).

* Для изменения цвета фона надо выбрать понравившийся вам цвет в разделе “Цвет фона”.
* Для изменения цвета рамки надо выбрать понравившийся вам цвет в разделе “Цвет рамки”.
* Для изменения цвета заливки надо выбрать понравившийся вам цвет в разделе “Цвет заливки”.
* Для изменения текстуры надо выбрать текстуру в разделе “Текстура”.
* Для включения/отключения рамки и изменения толщины надо поменять соответствующие параметры в настройках.
* Для изменения количества надо написать количество шариков в разделе “Количество”.
* Для изменения шариков на квадратики надо отметить соответствующий пункт в настройках.
* Для изменения прозрачного фона на фон, который заливается цветом надо выбрать соответствующий пункт в настройках.
* Для изменения скорости полета шариков надо перетянуть трекбар в настройках.



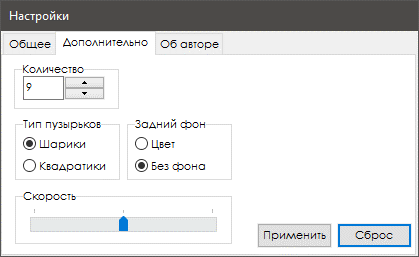


Рисунок 4.1.5 – Меню настроек Хранителя Экрана “Пузырьки”

Примечание 1: в некоторых случаях антивирус может выдавать сообщение, что файл “Bubbles.scr” является угрозой или вирусом. Автор данной программы рекомендует для устранения различных ошибок добавить данный файл в исключение антивируса, или на время работы программы выключать антивирус (как это сделать описано в пункте 3.2). Программа не является мошеннической или вирусом. Выполняйте данную процедуру если уверены, что программа была получена из надёжного источника.

Для сохранения и/или принятия настроек нужно нажать на кнопку “Применить”, всё готово!

## 4.2 Тестирование

В процессе разработки программы возникали следующие ошибки и недочёты: появление шариков друг на друге, неестественное отталкивание шариков, неправильная прорисовка шариков.

Появление шариков друг на друге. Для решения этой проблемы я разработал алгоритм, который вычисляет позицию шарика и если шарики находится на другом шарики, то он меняет его позицию. Сначала мы генерируем рандомную позицию шарика в пределах размера экрана пользователя.

masPosX[i] := Random(form1.ClientWidth);

masPosY[i] := Random(form1.ClientHeight);

Далее мы находим расстояние между шариками по формуле (Рисунок 4.2.1).

for k := 0 to i - 1 do //проверка с шарами

begin

//находим расстояние между двумя шариками (точками)

a := MasPosX[i] - MasPosX[k];

b := MasPosY[i] - MasPosY[k];

if sqrt(sqr(a) + sqr(b)) <= 241 then //формула: квадратный sqrt(sqr(x2-x1)+sqr(y2-y1))

n := True;

end;



Рисунок 4.2.1 – Формула нахождения расстояния между двумя точками

Если расстояние между шариками меньше чем 241 (диаметр шара), значит шарики накладываются друг на друга и мы запускаем алгоритм заново и генерируем новые значения.

Неестественное отталкивание шариков. Изначально, в начальных версиях программы отталкивание шариков было реализовано очень просто. При столкновении шарика просто менялось его скорость и направление. Чтоб отталкивание выглядело максимально правдоподобно, я использовал формулы для вычисления угла под каким шарик врезается в другой шар, на основе этих формул происходило отталкивание:

a := MasPosX[h0] + MasVelX[h0] - MasPosX[h1] - MasVelX[h1];

b := MasPosY[h0] + MasVelY[h0] - MasPosY[h1] - MasVelY[h1];

dist := sqrt(sqr(a) + sqr(b));

if dist <= 241 then //если удар то

begin

//СМЕНА НАПРАВЛЕНИЯ

bet := arctan2(MasPosY[h1] - MasPosY[h0], MasPosX[h1] - MasPosX[h0]);

//вычисляем углы удара

x1 := masvelx[h0] \* cos(-bet) - MasVelY[h0] \* sin(-bet);

y1 := MasVelX[h0] \* sin(-bet) + MasVelY[h0] \* cos(-bet);

x2 := MasVelX[h1] \* cos(-bet) - MasVelY[h1] \* sin(-bet);

y2 := MasVelX[h1] \* sin(-bet) + MasVelY[h1] \* cos(-bet);

//направляем шарики в противоположные углы

MasVelX[h0] := x2 \* cos(bet) - y1 \* sin(bet);

MasVelY[h0] := x2 \* sin(bet) + y1 \* cos(bet);

masVelX[h1] := x1 \* cos(bet) - y2 \* sin(bet);

MasVelY[h1] := x1 \* sin(bet) + y2 \* cos(bet);

Неправильная прорисовка шариков. В начальных версиях программы прорисовка шариков в движении происходила неправильно, а именно шарик не успевал обработаться и прорисовывался неполностью. Эта проблема была решена с помощью подключения двойной буферизации:

DoubleBuffered := True;

# Заключение

В результате выполнения курсового проекта была разработана программа-анимация демонстрирующая пример хранителя экрана. В программе были использованы основные типы данных, структуры данных.

В результате проделанной работы был повторен и использован при разработке программы пройденный материал по программированию в среде разработки Borland Delphi 7.

Также была проделана немаловажная работа по написанию пояснительной записки, опыт которой, безусловно, пригодиться в будущем при написании всевозможной документации.

Достаточно интересно было работать над самой поставленной задачей. Во время работы был прочитан материал, связанный непосредственно с языком программирования Delphi.

Единственное его отличие что он был полностью написан на языке Delphi, что позволило интегрировать настройки. Язык Delphi работал без наладок, был прост в обращении, возникающие ошибке при программировании легко исправлялись. Delphi зарекомендовал себя с лучшей стороны, и я бы порекомендовал его как для начинающих программистов, так и для профессионалов.

В последнее время компьютерные технологии продвигаются очень интенсивно, и это способствует бурному развитию программного обеспечения. Каждые полгода выходят продукты с множеством нововведений. С каждым разом все больше и больше функций заключают в себе данные программы. Но их развитие поставлено таким образом, что с каждой новой версией программа сохраняет предыдущий набор возможностей и пользователь может использовать как старые, так и новые функции, последние введены лишь для облегчения работы с программой.

# Однако сейчас популярность хранителей экрана или анимированных заставок падает. Так как в век новых технологий нет тех старых мониторов, которым желательно, а иногда и обязательно во время простоя нужно было выводить на экран динамически изменяющуюся картинку. Сейчас хранитель это всего лишь анимация требующаяся только тем, кто хочет каким-нибудь образом кастомизировать свой персональный компьютер.

# Список использованной литературы

1. Архангельский, А.Я. Программирование в Delphi 7. – М.: ЗАО "Бином - Пресс", 2010. – 1152с.

2. Галисеев, Г. В. Программирование в среде Delphi 7. Самоучитель. – М.: Дрофа, 2012 – 765 с.

3. Гофман, В. Э., Хомоненко, А. Д. Delphi 7. – СПб.: БХВ-Петербург, 2012. – 1152 с.

4. Фаронов, В. В. Система программирования Delphi. – СПб.: БХВ - Петербург, 2012. – 912 с.: ил.

5. Фаронов, В. В. Delphi. Программирование на языке высокого уровня: Учебник для вузов. – СПб.: Питер, 2010. – 640 с.

Список интернет-ресурсов

1. <http://beluch.ru/main/formclear.html> (05/06/2017)
2. <http://www.delphisources.ru/pages/faq/base/screensaver_creating.html> (08/05/2017)
3. <http://www.delphimaster.ru/articles/screensaver.html> (12/05/17)



Изм.

Лист

№ докум.

Подпись

Дата

Лист

22

КП 52493.002101.081

Разраб.

Бандарук

Провер.

Апанасевич

Реценз.

Н. Контр.

Утверд.

Приложение А

Лит.

Листов

34

МРК

Оператор

Запуск Хранителя

Пользовательские настройки из файла

Анимация

Сохранить в файл

Диаграмма вариантов использования программы

Стандартные настройки

Изменить настройки

# Приложение Б

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, ExtCtrls, Math, XPMan, ComCtrls, IniFiles;

type

TForm1 = class(TForm)

shp1: TShape;

shp2: TShape;

shp3: TShape;

shp4: TShape;

shp5: TShape;

tmr1: TTimer;

shp6: TShape;

shp7: TShape;

shp8: TShape;

shp9: TShape;

shp10: TShape;

img1: TImage;

procedure FormCreate(Sender: TObject);

procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

procedure tmr1Timer(Sender: TObject);

procedure FormClick(Sender: TObject);

procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);

procedure img1Click(Sender: TObject);

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

private

{ Private declarations }

procedure BallsFly();

public

{ Public declarations }

end;

type

Counter = 0..9;

var

Form1: TForm1;

PosX, Posy, VelX, Vely: Real;

masPosX, masPosY, masVelX, masVelY: array of Real;

Ball: array[0..9] of TShape;

//BallCount, PenWidth, TextureNumber, circles, screenshot, speed: Cardinal;

BallCount, PenWidth, TextureNumber, circles, screenshot, speed: Counter;

PenEn: Boolean;

BrushColor, PenColor, BGColor: TColor;

ini: TIniFile;

procedure setBallsSquares;

procedure setBallsCircles;

procedure setDiagonalLinesLeft;

procedure setDiagonalLinesRight;

procedure setCrossLines;

procedure setDiagonalCrossLines;

procedure setHorizontalLines;

procedure setVerticalLines;

procedure setTextureNone;

procedure ballsVisibleFL;

procedure delBallPen;

procedure BallPenWidth(width: Cardinal);

implementation

uses

Unit2;

{$R \*.dfm}

procedure Memory; //процедура выделения памяти и заполнения массива шариков

begin

//выделяем память массивам координат

SetLength(masPosX, 10);

SetLength(masPosY, 10);

SetLength(masVelX, 10);

SetLength(masVelY, 10);

with form1 do //заносим шарики в массив

begin

Ball[0] := shp1;

Ball[1] := shp2;

Ball[2] := shp3;

Ball[3] := shp4;

Ball[4] := shp5;

Ball[5] := shp6;

Ball[6] := shp7;

Ball[7] := shp8;

Ball[8] := shp9;

Ball[9] := shp10;

end;

end;

procedure createScreenShot;

var

bmp: TBitmap;

begin

//скриншот на задний фон

bmp := TBitmap.Create;

bmp.Width := Screen.Width;

bmp.Height := Screen.Height;

BitBlt(bmp.Canvas.Handle, 0, 0, Screen.Width, Screen.Height, GetDC(0), 0, 0, SRCCOPY); //GetDC(FindWindow('ProgMan', nil)), 0,0,SRCCOPY);

Form1.Img1.Width := Screen.Width;

Form1.Img1.Height := Screen.Height;

Form1.Img1.Picture.Assign(bmp);

bmp.Free;

end;

procedure ballPositionSet;

var

i: Counter;

k: Integer;

n: Boolean;

a, b: real;

begin

Randomize;

for i := 0 to 9 do

begin

repeat //генерим координаты шара, пока он не попадет внутрь окна на свободное место от других шаров

masPosX[i] := Random(form1.ClientWidth);

masPosY[i] := Random(form1.ClientHeight);

n := False;

for k := 0 to i - 1 do //проверка с шарами

begin

//находим расстояние между двумя шариками (точками)

a := MasPosX[i] - MasPosX[k];

b := MasPosY[i] - MasPosY[k];

if sqrt(sqr(a) + sqr(b)) <= 241 then //формула: квадратный sqrt(sqr(x2-x1)+sqr(y2-y1))

n := True;

end;

if (masPosX[i] > form1.ClientWidth - form1.Shp1.Width) or (masPosX[i] < 0) or //проверка с краем

(masPosY[i] > form1.ClientHeight - form1.Shp1.Width) or (masPosY[i] < 0) then

n := True;

until n = False;

masVelX[i] := RandomRange(1, 6) - 3 + speed; //скорость чтобы в разные стороны могла быть

masVelY[i] := RandomRange(1, 6) - 3 + speed;

if masVelx[i] = 0 then

masVelx[i] := -1;

if masVelY[i] = 0 then

masVelY[i] := 1;

end;

end;

procedure ballsVisibleTR; //если добавляют шары делаем их видимыми

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Visible := True;

end;

end;

procedure ballsPos;

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Left := Round(masPosX[i]);

Ball[i].Top := Round(masPosY[i]);

end;

end;

procedure ballsVisibleFL; //если шары отключают, делаем их невидимыми

var

i: Counter;

begin

for i := ballcount to 9 do

begin

Ball[i].Visible := False;

end;

end;

procedure bumpWalls; //столкновения со стенками

var

i: Cardinal;

begin

with Form1 do

begin

for i := 0 to BallCount - 1 do

begin

masPosX[i] := masPosX[i] + masVelX[i]; //шаг по Х

masPosY[i] := masPosY[i] + masVelY[i]; //шаг по У

if masPosX[i] > ClientWidth - Shp1.Width then

begin

masPosX[i] := ClientWidth - Shp1.Width;

masVelX[i] := -masVelX[i];

end

else if masPosX[i] < 0 then

begin

masPosX[i] := 0;

masVelX[i] := -masVelX[i];

end;

if masPosY[i] > ClientHeight - Shp1.Width then

begin

masPosY[i] := ClientHeight - Shp1.Width;

masVelY[i] := -masVelY[i];

end

else if masPosY[i] < 0 then

begin

masPosY[i] := 0;

masVelY[i] := -masVelY[i];

end;

end;

end;

end;

procedure bumpEachOther; //столкновения друг с другом

var

h0, h1, j: Cardinal;

a, b, dist, bet, x1, x2, y1, y2: extended;

n: integer;

xx0, yy0, xx1, yy1: single;

begin

for h0 := 0 to ballcount - 1 do

begin

for h1 := 0 to ballcount - 1 do

if h0 <> h1 then

begin

//вычисляем расстояние между шариками

a := MasPosX[h0] + MasVelX[h0] - MasPosX[h1] - MasVelX[h1];

b := MasPosY[h0] + MasVelY[h0] - MasPosY[h1] - MasVelY[h1];

dist := sqrt(sqr(a) + sqr(b));

if dist <= 241 then //если удар то

begin

//СМЕНА НАПРАВЛЕНИЯ

bet := arctan2(MasPosY[h1] - MasPosY[h0], MasPosX[h1] - MasPosX[h0]);

//вычисляем углы удара

x1 := masvelx[h0] \* cos(-bet) - MasVelY[h0] \* sin(-bet);

y1 := MasVelX[h0] \* sin(-bet) + MasVelY[h0] \* cos(-bet);

x2 := MasVelX[h1] \* cos(-bet) - MasVelY[h1] \* sin(-bet);

y2 := MasVelX[h1] \* sin(-bet) + MasVelY[h1] \* cos(-bet);

//направляем шарики в противоположные углы

MasVelX[h0] := x2 \* cos(bet) - y1 \* sin(bet);

MasVelY[h0] := x2 \* sin(bet) + y1 \* cos(bet);

masVelX[h1] := x1 \* cos(bet) - y2 \* sin(bet);

MasVelY[h1] := x1 \* sin(bet) + y2 \* cos(bet);

n := 0;

xx0 := masvelx[h0];

yy0 := masvely[h0];

xx1 := masvelx[h1];

yy1 := masvely[h1];

repeat //пытаемся раздвинуть шары,

MasPosX[h0] := MasPosX[h0] + xx0 \* 0.0011;

MasPosY[h0] := MasPosY[h0] + yy0 \* 0.0011;

MasPosX[h1] := MasPosX[h1] + xx1 \* 0.0011;

MasPosY[h1] := MasPosY[h1] + yy1 \* 0.0011;

a := MasPosX[h0] - MasPosX[h1];

b := MasPosY[h0] - MasPosY[h1];

dist := sqrt(sqr(a) + sqr(b));

inc(n);

until (dist > 241) or (n > 50000);

end;

end;

MasPosX[h0] := MasPosX[h0] + MasVelX[h0] \* 0.001;

MasPosY[h0] := MasPosY[h0] + MasVelY[h0] \* 0.001;

end;

end;

//ФОРМЫ ШАРИКОВ

procedure setBallsSquares; //сделать из шариков квадратики

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Shape := stSquare;

end;

end;

procedure setBallsCircles; //сделать шарики круглыми

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Shape := stCircle;

end;

end;

//ТЕКСТУРЫ

procedure setDiagonalLinesLeft; //сделать текстуру Диагональные линии(Л)

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Brush.Style := bsBDiagonal;

end;

end;

procedure setDiagonalLinesRight; //сделать текстуру Диагональные линии (П)

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Brush.Style := bsFDiagonal;

end;

end;

procedure setCrossLines; //сделать текстуру Крестики

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Brush.Style := bsCross;

end;

end;

procedure setDiagonalCrossLines; //сделать текстуру Диагональные крестики

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Brush.Style := bsDiagCross;

end;

end;

procedure setHorizontalLines; //сделать текстуру Горизонтальные линии

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Brush.Style := bsHorizontal;

end;

end;

procedure setVerticalLines; //сделать текстуру Вертикальные линии

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Brush.Style := bsVertical;

end;

end;

procedure setTextureNone; //убрать текстуру (сделать прозрачным)

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Brush.Style := bsClear;

end;

end;

procedure setTextureSolid; //текстуры нет (заливка выбранным цветом)

var

i: Counter;

begin

for i := 0 to BallCount - 1 do

begin

Ball[i].Brush.Style := bsSolid;

end;

end;

//ЦВЕТА

procedure BallBrushColor(Bcolor: TColor);

var

i: Counter;

begin

if Bcolor = 1337 then //невидимый цвет (прозрачный)

begin

for i := 0 to BallCount-1 do

begin

Ball[i].Brush.Style := bsClear;

end;

end

else

begin

for i := 0 to BallCount-1 do

begin

Ball[i].Brush.Color := Bcolor;

end;

end;

end;

procedure BallPenColor(Pcolor: TColor); //цвет рамки

var

i: Counter;

begin

for i := 0 to BallCount-1 do

begin

Ball[i].Pen.Color := Pcolor;

end;

end;

//ТЕКСТУРА

procedure BallTexture(num: Cardinal);

begin

with form2 do

begin

if num = 0 then

setTextureSolid;

if num = 1 then

setDiagonalLinesLeft;

if num = 2 then

setDiagonalLinesRight;

if num = 3 then

setCrossLines;

if num = 4 then

setDiagonalCrossLines;

if num = 5 then

setHorizontalLines;

if num = 6 then

setVerticalLines;

if num = 7 then

setTextureNone;

end;

end;

//ТОЛЩИНА РАМКИ

procedure BallPenWidth(width: Cardinal);

var

i: Counter;

begin

for i := 0 to BallCount-1 do

begin

Ball[i].Pen.Style := psSolid;

Ball[i].Pen.Width := width;

end;

end;

//УБРАТЬ РАМКУ

procedure delBallPen;

var

i: Counter;

begin

for i := 0 to BallCount-1 do

begin

Ball[i].Pen.Style := psClear;

end;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

Sleep(150); //ждем 1.5 секунды, чтоб успела закрыться форма 2 и не попала в скриншот

//создаем файл со стандартными настройками, если такого нет

ini := TIniFile.Create(ExtractFilePath(ParamStr(0)) + 'settings.ini');

//извлекаем настройки из файла

BallCount := ini.ReadInteger('Дополнительно', 'Количество', 5);

BGColor := ini.ReadInteger('Общее', 'Цвет фона', 0);

PenColor := ini.ReadInteger('Общее', 'Цвет рамки', 0);

BrushColor := ini.ReadInteger('Общее', 'Цвет заливки', 1337);

TextureNumber := ini.ReadInteger('Общее', 'Текстура', 0);

PenEn := ini.ReadBool('Общее', 'Рамка', True);

PenWidth := ini.ReadInteger('Общее', 'Толщина рамки', 2);

circles := ini.ReadInteger('Дополнительно', 'Шарики', 0);

screenshot := ini.ReadInteger('Дополнительно', 'Без фона', 1);

speed := ini.ReadInteger('Дополнительно', 'Скорость', 2);

Memory; //выделяем память и заполняем массив

DoubleBuffered := True; //включаем двойную буферизацию

BallBrushColor(BrushColor); //цвет шаров

BallPenColor(PenColor); //цвет рамки

BallTexture(TextureNumber); //текстура

if PenEn = False then //удалить рамку

delBallPen

else

BallPenWidth(PenWidth); //ширина рамки

if circles = 0 then

setBallsCircles //шарики

else

setBallsSquares; //квадратики

form1.Color := BGColor; //фон

Form1.BorderStyle := bsNone; //без бордюров

Form1.ClientHeight := Screen.Height; //по размеру экрана

Form1.ClientWidth := Screen.Width; //по ширине экрана

Form1.Position := poScreenCenter; //по центру

if screenshot = 1 then

begin

createScreenShot; //скриншот на задний фон (если без фона)

img1.Visible := True;

end

else

img1.Visible := False; //цвет формы

showcursor(false); //без курсосра

ballsVisibleFL; //прячем неиспользуемые шарики

ballsVisibleTR; //показываем используемые шарики

ballPositionSet; //расставляем шарики по экрану

end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);

begin

//СОХРАНЯЕМ НАСТРОКИ при закрытии

ini := TIniFile.Create(ExtractFilePath(ParamStr(0)) + 'settings.ini');

ini.WriteInteger('Общее', 'Цвет фона', BGColor);

ini.WriteInteger('Общее', 'Цвет рамки', PenColor);

ini.WriteInteger('Общее', 'Цвет заливки', BrushColor);

ini.WriteInteger('Общее', 'Текстура', TextureNumber);

ini.WriteBool('Общее', 'Рамка', PenEn);

ini.WriteInteger('Общее', 'Толщина рамки', PenWidth);

ini.WriteInteger('Дополнительно', 'Количество', BallCount);

ini.WriteInteger('Дополнительно', 'Шарики', circles);

ini.WriteInteger('Дополнительно', 'Без фона', screenshot);

ini.WriteInteger('Дополнительно', 'Скорость', speed);

end;

procedure ballPenRandom; //рандомный цвет шариков

var

i: Counter;

begin

for i := 0 to 9 do

begin

if form2.clrbx2.ItemIndex = 16 then

begin

Ball[i].Pen.Color := Random($FFFFFF);

end;

end;

end;

procedure TForm1.BallsFly;

begin

ballsPos; //делаем шаг

bumpWalls; //проверка на удары об стену

bumpEachOther; //проверка на удары друг об друга

ballPenRandom; //рандомная рамка (если включено)

end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

begin

//открытие настроек по комбинации клавиш (CTRL + TAB)

if (GetKeyState(ord(VK\_CONTROL)) < 0) then

begin

if (GetKeyState(ord(VK\_TAB)) < 0) then

begin

form2.show;

showcursor(true);

end;

end

else

begin

Application.Terminate;

end;

end;

procedure TForm1.FormClick(Sender: TObject);

begin

Application.Terminate;

end;

procedure TForm1.tmr1Timer(Sender: TObject);

begin

BallsFly;

end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);

begin

//Application.Terminate;

end;

procedure TForm1.img1Click(Sender: TObject);

begin

form1.Close;

end;

end.

unit Unit2;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, ExtCtrls, Unit1, ComCtrls, XPMan, Menus, IniFiles,

jpeg;

type

TForm2 = class(TForm)

clrbx1: TColorBox;

clrbx2: TColorBox;

clrbx3: TColorBox;

btn4: TButton;

TrackBar1: TTrackBar;

edt1: TEdit;

ud1: TUpDown;

btn1: TButton;

grp1: TGroupBox;

grp2: TGroupBox;

grp3: TGroupBox;

grp4: TGroupBox;

pgc1: TPageControl;

ts1: TTabSheet;

ts2: TTabSheet;

ts4: TTabSheet;

grp5: TGroupBox;

rg1: TRadioGroup;

cbb1: TComboBox;

grp6: TGroupBox;

ud2: TUpDown;

edt2: TEdit;

grp7: TGroupBox;

rg2: TRadioGroup;

img1: TImage;

lbl1: TLabel;

lbl2: TLabel;

lbl3: TLabel;

lbl4: TLabel;

lbl5: TLabel;

chk1: TCheckBox;

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

procedure FormCreate(Sender: TObject);

procedure btn4Click(Sender: TObject);

procedure btn1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

clrBxMas: array[0..2] of TColorBox;

implementation

{$R \*.dfm}

procedure ballCountSet;

begin

with Form2 do

begin

BallCount := StrToInt(edt1.Text);

end;

end;

procedure speedChange; //меняем скорость (к основной прибавляется выбранная пользователем)

var

i: Counter;

begin

speed := Form2.TrackBar1.Position;

for i := 0 to 9 do

begin

masVelX[i] := masvelx[i] + speed;

masVelY[i] := masvely[i] + speed;

end;

end;

procedure toDefault; //сброс настроек к стандартным

var

i, j: Counter;

begin

Form2.edt1.text := IntToStr(5);

Form2.edt2.text := IntToStr(2);

if speed <> 2 then

begin

Form2.TrackBar1.Position := 2;

speed := 2;

for j := 0 to 9 do

begin

masVelX[j] := speed;

masVelY[j] := speed;

end;

end;

form1.Color := clBlack;

for i := 0 to 9 do

begin

if i <= 2 then

begin

clrBxMas[i].Selected := clBlack;

if i = 1 then

clrBxMas[i].Selected := clBlack;

if i = 2 then

clrBxMas[i].ItemIndex := 16;

end;

Ball[i].Pen.Color := clBlack;

Ball[i].Brush.Style := bsClear;

BallCount := 5;

end;

form2.rg1.ItemIndex := 0;

form2.cbb1.ItemIndex:= 7;

form2.chk1.Checked:= True;

BallPenWidth(2);

ballsVisibleFL;

setBallsCircles;

end;

procedure TForm2.FormCreate(Sender: TObject);

begin

clrbx3.Items.Add('None');

clrbx2.Items.Add('Random');

ud1.Associate := edt1;

ud1.Position := ini.ReadInteger('Дополнительно', 'Количество', 5);

clrbx1.Selected := ini.ReadInteger('Общее', 'Цвет фона', BGColor);

clrbx2.Selected := ini.ReadInteger('Общее', 'Цвет рамки', PenColor);

if BrushColor = 1337 then //цвет залики не выбран, стави прозрачный (1337)

clrbx3.ItemIndex := 16

else

clrbx3.Selected := ini.ReadInteger('Общее', 'Цвет заливки', BrushColor);

Form2.cbb1.ItemIndex := ini.ReadInteger('Общее', 'Текстура', TextureNumber);

ud2.Associate := edt2;

chk1.Checked:= ini.ReadBool('Общее', 'Рамка', PenEn);

ud2.Position := ini.ReadInteger('Общее', 'Толщина рамки', 2);

rg1.ItemIndex := ini.ReadInteger('Дополнительно', 'Шарики', circles);

rg2.ItemIndex := ini.ReadInteger('Дополнительно', 'Без фона', 1);

TrackBar1.Position := ini.ReadInteger('Дополнительно', 'Скорость', speed);

form2.Position := poScreenCenter; //форма по центру

//заносим colorBox в массив для удобства использования

clrBxMas[0] := clrbx1;

clrBxMas[1] := clrbx2;

clrBxMas[2] := clrbx3;

end;

procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);

begin

//сохраняем настроки при закрытии формы

ini := TIniFile.Create(ExtractFilePath(ParamStr(0)) + 'settings.ini');

ini.WriteInteger('Общее', 'Цвет фона', Form2.clrbx1.ItemIndex);

ini.WriteInteger('Общее', 'Цвет рамки', Form2.clrbx2.Selected);

if clrbx3.ItemIndex = 16 then

ini.WriteInteger('Общее', 'Цвет заливки', 1337)

else

ini.WriteInteger('Общее', 'Цвет заливки', Form2.clrbx3.Selected);

ini.WriteInteger('Общее', 'Текстура', Form2.cbb1.ItemIndex);

ini.WriteBool('Общее', 'Рамка', form2.chk1.Checked);

ini.WriteInteger('Общее', 'Толщина рамки', StrToInt(Form2.edt2.Text));

ini.WriteInteger('Дополнительно', 'Количество', StrToInt(Form2.edt1.Text));

ini.WriteInteger('Дополнительно', 'Шарики', form2.rg1.ItemIndex);

ini.WriteInteger('Дополнительно', 'Без фона', form2.rg2.ItemIndex);

ini.WriteInteger('Дополнительно', 'Скорость', Form2.TrackBar1.Position);

end;

procedure TForm2.btn4Click(Sender: TObject);

begin

toDefault; //сброс настроек

end;

procedure TForm2.btn1Click(Sender: TObject);

begin

speedChange; //задаем скорость

Form2.Close;

if paramCount = 2 then

form1.Close

else

Form1.FormCreate(Sender);

end;

end.

program Project1;

uses

Forms,

Graphics,

Windows,

Messages,

Unit1 in 'Unit1.pas' {Form1},

Unit2 in 'Unit2.pas' {Form2};

{$E scr}

{$R \*.res}

begin

// no params -> Context menu -> Configure

// /p HWND -> preview; hwnd - parent window

// /c:HWND -> Configure, parent - modal

// /s - view

if (paramCount = 0) then

begin

Application.Initialize;

Application.Title := 'Bubbles';

Application.CreateForm(TForm1, Form1);

Application.CreateForm(TForm2, Form2);

Application.Run;

end

else if {(paramCount = 2) and} (CharLower(PChar(ParamStr(1))) = '/p') then

begin

Application.Initialize;

Application.CreateForm(TForm1, Form1);

Form1.Width:= 0;

Application.CreateForm(TForm2, Form2);

form2.Show;

ShowCursor(True);

Application.Run;

end

else if {(ParamCount = 1) and} (CharLower(PChar(ParamStr(1))) = '/s') then

begin

Application.Initialize;

Application.CreateForm(TForm1, Form1);

Application.CreateForm(TForm2, Form2);

Application.Run;

end;

end.