**Введение**

На курсовое проектирование была поставлена задача разработать программный продукт на тему «Марио».

Цель курсового проекта заключается в реализации игрового процесса как в игре Марио.

Создаваемая программа будет рассчитана на любого рода пользователей. Она будет особенно полезна людям, увлекающимся играми в жанре «платформер».

Далее приведем краткое описание разделов пояснительной записки.

Первый раздел носит название “Анализ задачи”. В нем вы сможете ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи, определение ее организационно-экономической сущности. Также в этом разделе вы сможете узнать о том, как данная задача решается в настоящее время. Все входные и выходные данные тоже будут описаны в первом разделе. В подразделе “Инструменты разработки” будет рассмотрена среда, в которой создается данный курсовой проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачи.

В разделе «Проектирование задачи» будут рассмотрены основные аспекты разработки программного продукта. Здесь можно будет узнать об организации данных в контексте среды разработки. В данном разделе будет четко описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации, описана разработка системы справочной информации.

«Реализация задачи» – это третий раздел пояснительной записки, в котором описываются все элементы и объекты, которые будут использованы при реализации данного приложения. В этом разделе будут четко описаны функции пользователя и их структура. Здесь можно будет найти таблицу, в которой будет представлена полная аннотация файлов используемых в данном проекте.

Четвертый раздел – «Тестирование». В нем будет описано полное и функциональное тестирование данной программы, т.е. будет оттестирован каждый пункт меню, каждая операция, которая выполняется приложением. Будут смоделированы все возможные действия пользователя при работе с программой, начиная от запуска до выхода.

В разделе «Применение» будет описано назначение, область применения, среда функционирования курсовой программы. Также в нем будет описано использование справочной системы.

«Заключение» будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В приложении к пояснительной записке будет приведен листинг программы с необходимыми комментариями.

Схема работы системы будет представлена в графической части.

**1 Анализ задачи**

**1.1 Постановка задачи**

Темой данного курсового проекта является «Разработка программного продукта «Марио».

Видео игры играют большую роль в нашем современном мире. Сейчас они служат способом отвлечься от реального мира.

В моём проекте вы увидите игру, заставляющую почувствовать чувство ностальгии. И название ей игровое приложение «Марио». Оно заключается в том, что вы проходите уровень. Уровень является достаточно простым для прохождения, но вызовет приятные эмоции.

Существуют аналоги данного приложения. Существуют различные онлайн реализации игры «Марио» на сайтах, офлайн приложений с такой возможностью, а также много статей на эту тему.

Периодичность использования данного программного продукта не ограничена. Пользователь может в любое время установить, а также удалить приложение.

Цель данного курсового проекта – разработать программный продукт, который даст возможности снова сыграть в игру «Марио».

В приложении должен быть простой и понятный интерфейс, который будет интуитивно понятен любому пользователю, который ранее не имел большого опыта с шифрованием файлов.

Разрабатываемый программный продукт должен позволять выполнять следующие действия:

- игра в Марио;

- просмотр справки;

В разрабатываемой программе будут использоваться один вид данных.

Постоянной информацией в проекте будут являться текстовые файлы, картинки, медиафайлы и др.

Программный продукт предоставляет функционал для следующего ряда пользователей: простой пользователь.

**1.2 Инструменты разработки**

Для разработки данного проекта будет выбрана среда Delphi 11.3, так как это самое удобная и доступная среда разработки на данный момент. Delphi 11.3 - среда разработки, относящийся к классу RAD- (Rapid Application Development – «Средство быстрой разработки приложений») средств CASE – технологии. Delphi 11.3 сделал разработку приложений для Windows быстрым и приятным процессом. Теперь разрабатывать сложные и интересные проекты можно только одним человеком, использующим Delphi 11.3.

Интерфейс Windows обеспечивает полное перенесение CASE-технологии в интегральную систему поддержки работ по созданию прикладной системы на всех фазах жизненного цикла работы и проектирования системы.

Delphi 11.3 обладает широким набором возможностей, начиная от проектировщика форм и заканчивая поддержкой всех форматов популярных баз данных. Среда устраняет необходимость программировать такие компоненты Windows общего назначения, как метки, программы и даже диалоговые панели. Работая в Windows, можно видеть одинаковые «объекты» во многих разнообразных приложениях. Диалоговые панели (например, Choose File и Save File) являются примерами многократно-используемых компонентов, встроенных непосредственно в Delphi 11.3, который позволяет приспособить эти компоненты к имеющийся задаче, чтобы они работали именно так, как требуется создаваемому приложению. Также здесь имеются предварительно- определенные визуальные и не визуальные объекты.

Три основные части разработки интерфейса следующие: проектирование панели, проектирование диалога и представление окон. Для общего пользовательского доступа также должны учитываться условия применения архитектуры прикладных систем.

Сегодня появилась реальная возможность с помощью моделирования на современных многофункциональных средствах обработки и отображения информации таких как Delphi 11.3 конкретизировать тип и характеристики используемых информационных моделей, выявить основные особенности будущей деятельности операторов, сформулировать требования к параметрам аппаратно-программных средств интерфейса взаимодействия и т.д.

Delphi 11.3 позволяет создать различные виды программ: консольные приложения, оконные приложения, приложения для работы с Интернетом и базами данных. То есть, Delphi 11.3 является не только средствами для работы с языком программирования Паскаль, но дополнительные инструменты, призванные для максимального упрощения и ускорения создание приложений.

К дополнительным инструментам можно отнести визуальный редактор форм, благодаря которому можно с легкостью создать полноценную программу, и другие визуальные составляющие разработки программного обеспечения. С Delphi вам не нужно вручную просчитывать расположение каждого элемента интерфейса пользователя, поэтому при разработке программы значительно экономится время.

Выгоды от проектирования в среде Windows с помощью Delphi 11.3:

- устраняется необходимость в повторном вводе данных;

- обеспечивается согласованность проекта и его реализации;

- увеличивается производительность разработки и переносимость программ.

Ни одно серьезное программное обеспечение не обходится без модуля справочной информации и руководства пользователя. Это придает программе законченный вид и показывает заботу о пользователе.

Help + Manual 9– легкий в использовании и функциональный инструмент, упрощающий создание справочных файлов Windows, печать справочных руководств и документации в целом. Программа имеет интуитивно понятный интерфейс. Все созданные проекты можно сохранить в различных форматах:

Для создания инсталлятора будет использоваться мощное и удобное средство - Smart Install Maker.

Программа обладает удобным и интуитивно понятным интерфейсом, а также полным набором необходимых функций для создания профессиональных инсталляторов с минимальным размером, высокой степенью сжатия файлов и приятным интерфейсом.

Помимо стандартного минимума, Smart Install Maker позволяет редактировать системный реестр и INI-файлы, создавать программные ярлыки, запускать ассоциируемые и исполняемые файлы, регистрировать новые шрифты и ActiveX компоненты, отображать тексты информации и лицензионного соглашения. Также, с помощью этой утилиты, можно создать мультиязыковые инсталляторы с поддержкой более 20-ти популярных языков мира.

Microsoft Word 2022 – редактор текста для написания документации.

Разработка ведется на ноутбуке MSI Katana GF66. У данного ноутбука следующие параметры:

- процессор Intel Core i7;

- объем ОЗУ 16 гб;

- объем места на HDD – 1.5 тб ;

- видеоподсистема 1920х1080 точек с глубиной цвета 32 Bit;

- ОС – Windows 10.

Как видно разрабатываемое приложение не очень требовательно к аппаратным ресурсам, что, является большим плюсом.

**1.3 Требования к приложению**

На этапе исследования предметной области был установлен целый ряд требований, которые предъявляются к разрабатываемой программе.

При моделировании форм следует учесть такой момент: приложение нацелено на недолгую игру, из чего следует, что оно не нуждается в экстраординарном интерфейсе.

Требования к интерфейсу: в связи с частым использование программы она должна быть с приятной цветовой гаммой и понятной для пользователя. Следовательно, каждое окно должно иметь ясную визуальную иерархию своих элементов. Фрагменты текста должны располагаться на экране так, чтобы пользователя было просто и понятно принимать информацию.

Пользователь не должен испытывать какого-либо дискомфорта в плане восприятия информация, отображённой на экране. Объекты (рисунки и символы) не должны быть слишком мелкие. Все окна приложения по возможности должны помещаться на экран полностью, так как использование в процессе работы полос прокруток достаточно неудобно.

На одной форме нельзя допускать избытка и нагромождения данных.

Формы должны быть эффектно оформлены согласно тематике разрабатываемого проекта.

Требования к надежности: cпециальных требований к надежности не предъявляется.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбое (не крахом) операционной системы, не должно превышать 30-ти минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Отказы программы вследствие некорректных действий пользователя при взаимодействии с программой через графический интерфейс не должны влиять на конечный результат. ПП должен иметь средства фиксации всех действий в процессе изготовления программного продукта. Это необходимо для восстановления состояния процесса разработки на любом его этапе (при комплексной отладке).

**2 Проектирование задачи**

**2.1 Организация данных**

Проектирование задачи – это очень важный и ответственный этап в разработке любого приложения.

Важным является он вследствие того, что методы, по средствам которых пользователь управляет формами, построены на высокой степени специализации каждого из компонентов.

Необходимым условием при разработке данного приложения является описание организации данных, т.е. логическая и физическая структура данных в контексте среды разработки. В разрабатываемой программе будут использоваться один вида данных.

Первым видом являются данные, которые будут введены разработчиком на этапе реализации задачи. Сюда можно отнести изображения (иконки, кнопки), описание.

Таким образом, организация данных является важной задачей при разработке данной и любой программы.

**2.2 Процессы**

Согласно всем перечисленным требованиям и указаниям, которые были рассмотрены в разделе «Анализ задачи», было определено, чем конкретно должна заниматься разрабатываемая курсовая программа. Главной ее задачей будет являться шифрование и дешифрование файлов с использованием шифра «Вернама».

Программа будет создавать ключ, потом на основе этого ключа кодировать файл, после чего записывать закодированный файл в заранее выбранное изображение. Так же будет реализовано декодирование. Имея нужный ключ, пользователь сможет декодировать файл из изображения.

**2.3 Описание внешнего пользовательского интерфейса**

Важным при выполнении курсового проекта является организация диалога между пользователем и самой программой. Во многом это зависит от того, как программист разработает данную программу, какие компоненты будут использованы и какие методы будут автоматизированы.

Особое внимание следует уделить интерфейсу. Разработчик должен так организовать внешний вид своей программы, что бы пользователь понял, что от него требуется.

Для организации эффектной работы пользователя нужно создать целостное приложение данной предметной области, в которой все компоненты приложения будут сгруппированы по функциональному назначению. При этом необходимо обеспечить удобный графический интерфейс пользователя.

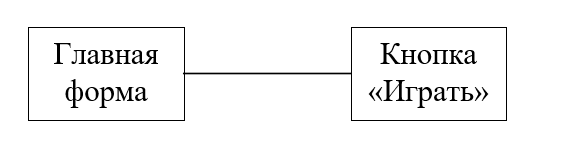
Таким образом, для успешной работы всего проекта в целом следует обеспечить интуитивно понятный интерфейс с приятной гаммой цветов и шрифтами.

Рисунок 1 – Структура навигации по проекту

**3 Реализация**

**3.1 Реализация проекта**

**3.1.1 Структура программы**

Данный курсовой проект содержит 3 модулей. Далее рассмотрим

назначение каждого модуля:

1 Модуль Unit1 – отвечает за главное меню, начало игры;

2 Модуль Unit2 – отвечает за загрузку проекта;

3 Модуль Main – отвечает за игровой процесс;

**3.1.2 Структура и описание процедур и функций пользователя**

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 – Процедуры и функции |  |  |  |
| Имя процедуры (функции) | В каком модуле  находится | За каким  компонентом  закреплена | Назначение |
| 1 | 2 | 3 | 4 |
| 1 procedure TForm1.FormActivate(Sender: TObject); | Unit1 | TForm1 | закрывает форму ScreenForm |
| 2 procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction); | Unit1 | TForm1 | закрывает форму Form2 |
| 3 procedure TForm1.FormCreate(Sender: TObject); | Unit1 | TForm1 | запускает MediaPlayer |
| 4 procedure TForm1.MediaPlayer1Notify(Sender: TObject); | Unit1 | MediaPlayer1 | повторное проигрывание мелодии |
| 5 procedure TForm1.SpeedButton1Click(Sender: TObject); | Unit1 | SpeedButton1 | переход на форму Main по нажатию кнопки |
| 6 procedure TForm1.SpeedButton2Click(Sender: TObject); | Unit1 | SpeedButton2 | закрытие всего проекта по нажатию кнопки |
| Продолжение таблицы 1 |  |  |  |
| 1 | 2 | 3 | 4 |
| 7 procedure TForm2.FormActivate(Sender: TObject); | Unit2 | TForm2 | загружает изображение при запуске формы |
| 8 procedure TForm2.Timer1Timer(Sender: TObject); | Unit2 | Timer1 | изменение шкалы загрузки со временем |
| 9 constructor TScreenForm.Create(AOwner: TComponent); | Main | TScreenForm | загрузка карты, прорисовка карты |
| 10 procedure TScreenForm.DisplayPaint(Sender: TObject); | Main | TScreenForm | отрисовка цветной карты |
| 11 procedure TScreenForm.DrawRect(ATarget, ASource: TCanvas; AX, AY: Integer;  ASRect: TRect); | Main | TScreenForm | отображение прямоугольного изобаражения на холсте TCanvas |
| 12 procedure TScreenForm.FormCreate(Sender: TObject); | Main | TScreenForm | закрывает Form1, Form2 при создании данной формы |
| 13 procedure TScreenForm.FormKeyDown(Sender: TObject; var Key: Word;  Shift: TShiftState); | Main | TScreenForm | проверка клавиш, используемых в игровом процессе |
| 14 procedure TScreenForm.FormResize(Sender: TObject); | Main | TScreenForm | изменяет размер экрана под подходящие параметры |
| 15 procedure TScreenForm.ToggleFullScreen; | Main | TScreenForm | возможность переключить игру в полноэкранный режим |
| 16 procedure TScreenForm.LoadSprite(AIndex: Integer; AStates: array of string); | Main | TScreenForm | загрузка изображений в необходимые места |
| 17 procedure TScreenForm.LoadSprites; | Main | TScreenForm | загрузка изображений |
| 18 procedure TScreenForm.RenderTimerTimer(Sender: TObject); | Main | TScreenForm | физика игры |

**3.1.3 Описание использованных компонентов**

Таблица 2 – Использованные компоненты

|  |  |  |
| --- | --- | --- |
| Компонент | На какой форме расположен | Назначение |
| 1 TImage1 | Form1,Form2 | Используется для создания  заднего фона |
| 2 TImage2 | Form1 | Используется для создания изображения кнопки |
| 3 TImage3 | Form1 | Используется для создания изображения кнопки |
| 4 TMediaPlayer1 | Form1 | Используется для музыкального сопровождения |
| 5 TSpeedButton1 | Form1 | Используется для создания кнопок-изображений |
| 6 TSpeedButton2 | Form1 | Используется для создания кнопок-изображений |
| 7 TProgressBar1 | Form2 | Используется для  отображения степени загрузки |
| 8 TTimer1 | Form2 | Используется для степени загрузки |
| 9 TDisplay | ScreenForm | Используется для экрана игры |
| 10 TRenderTimer | ScreenForm | Используется для обновления кадров в игре |

**3.2 Спецификация программы**

Таблица 3 – Спецификация программы

|  |  |
| --- | --- |
| Имя файла | Назначение |
| 1 | 2 |
| 1 mario.exe | Исполняемый файл проекта |
| 2 Unit1.dcu | Скомпилированный код модуля Unit1 |
| 3 Unit2.dcu | Скомпилированный код модуля Unit2 |
| 4 Main.dcu | Скомпилированный код модуля Main |
| 5 mario.mp3 | Мелодия |
| 6 Level\_1\_1Entities.dcu | Скомпилированный код модуля Level\_1\_1Entities |
| 7 Sprite-0001.png | Изображение кнопки «Играть» |
| 8 Sprite-0002.png | Изображение кнопки «Закрыть» |
| 9 Image1.jpg | Изображение заднего фона |
| 10 Image2.gif | Изображение заднего фона на загрузочном экране |
|  |  |
|  |  |
| 9 Rand.exe | Исполняемый файл вспомогательной программы |
| 10 data.dat | Файл для сохранения сгенерированных байтов сгенерированных Rand.exe |
| 11 settings.txt | Файл с настройками программы |
| 12 key.png | Картинка, хранящая ключ |
| 13 source.png | Картинка, хранящая зашифрованное сообщение |
| 14 test.txt | Текстовый файл для тестирования работы программы |
| 15 HID.rsm | Хранит настройки проекта |
| 16 c\_key.png | Картинки для отображения режима создания ключа |
| 17 decode.png | Картинки для отображения режима дешифрования |
| 18 encode.png | Картинки для отображения режима шифрования |
| 19 Conf.dfm | Файл |
| 20 Conf.pas | Файл программного модуля для формы Conf |
| 21 Picture.pas | Файл программного модуля |
| 22 Cypher.pas | Файл программного модуля |
| 23 Main.pas | Файл программного модуля для формы Main |
| 24 Main.dfm | Главная форма |
| 25 Loading.dfm | Форма с заставкой |
| 26 Loading.pas | Файл программного модуля для формы Loading |
| 27 Rand.pas | Файл программного модуля |
| 28 Saves.pas | Файл программного модуля |
| 29 HID.res | Файл для хранения ресурсов проекта |
| 30 HID.dpr | Файл проекта, связывает все файлы, из которыx состоит приложение |
| 31 HID.dproj | Файл, который служит для связи всего проекта |
| 32 HID.identcache | Кэш файл для хранения информации об идентификационных данных файлов |
| 33 HID.local | Файл с локальными данными проекта |
| 34 HID.chm | Файл справки, содержит помощь по работе с программой |
| 35 Setup.exe | Файл для установки приложения |

**4 Тестирование**

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения испытания реализации программы было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме.

Отчёт о результатах тестирования предоставлен в таблице 4.

Таблица 4 – Отчёт результатах тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тест | Ожидаемый результат | Физический результат | Результат  тестирования |
| 1 | Проверка нажатия кнопки «Играть» | Переход в игровое меню | Переход в игровое меню | Выполнено |
| 2 | Проверка нажатия кнопки «Закрыть» | Закрытие всего проекта | Закрытие всего проекта | Выполнено |

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в процедурах.

**5 Применение**

**5.1 Общие сведения о программном продукте**

Цель данного программного продукта в обеспечении приятного время препровождения за игрой «Марио».

Приложение рассчитано на различных типов пользователей, которые имеют разные интересы и цели. Данный игра является достаточно старой, но не престает иметь свою популярность.

Быстродействие программы зависит от характеристик персонального компьютера, а именно, рабочей частоты процессора, объема оперативной памяти и т.д. Но стоит отметить, что данный программный продукт легко запускается и функционирует на относительно современных машинах.

**5.2 Инсталляция**

Для того, чтобы установить программу необходимо запустить файл Setup.exe. Появится окно установки приложения «mario».

Затем следует выполнять указанные инструкции установки приложения.

**5.3 Выполнение программы**

**5.3.1 Запуск программы**

Для запуска программы требуется дважды щелкнуть левой кнопки мыши на ярлыке или исполнительном файле с названием «mario.exe».

По подготовленным тестам будет осуществляться функциональное и полное тестирование программного продукта. Отчет о результатах тестирования будет представлен в 4 разделе пояснительной записки.

**5.3.2 Инструкции по работе с программой**

Первое, что видит пользователь при запуске программы – это загрузочный экран(рисунок 2)**.**

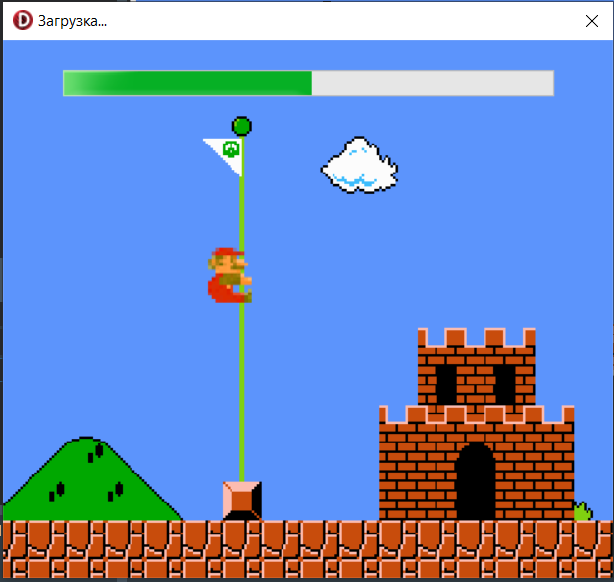
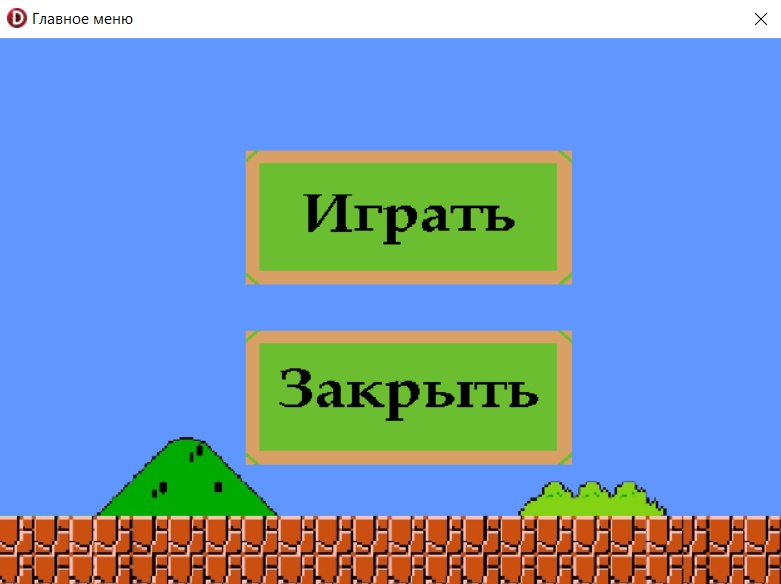
 После загрузки пользователь попадает в главное меню, в котором имеется выбор между двумя кнопками «Играть», «Закрыть»(рисунок 3).

Рисунок 2 – Загрузочный экран

Рисунок 3 – Главное меню

На форме главного меню пользователь может нажать на одну из кнопок «Играть», «Закрыть». При нажатии кнопки «Закрыть» проект закрывается. При нажатии кнопки «Играть» запускается сама игра(рисунок 4).

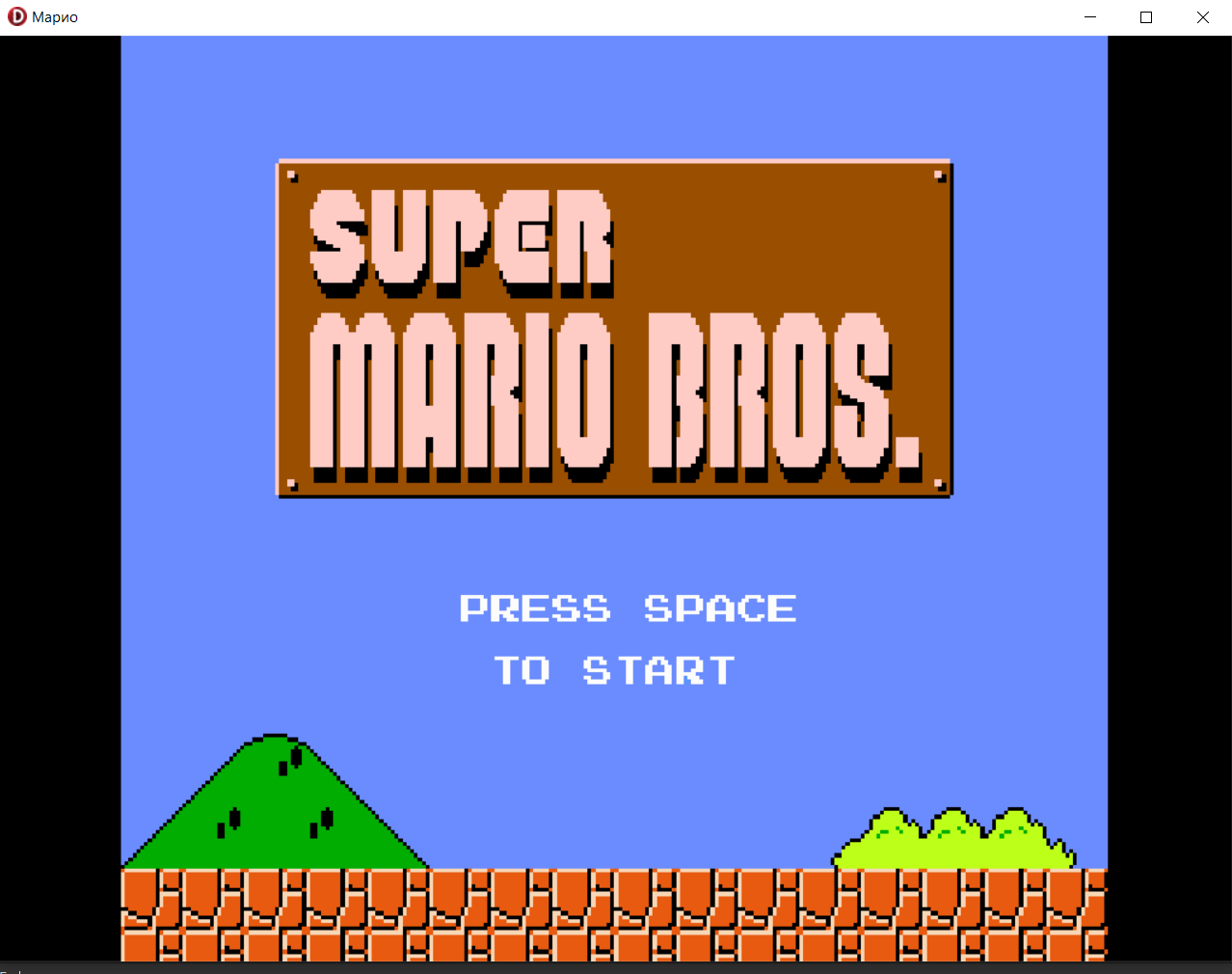
Рисунок 4 – Игровой процесс

Рисунок 4 - Форма настроек

**5.3.3 Завершение работы с программой**

Завершить программу можно несколькими способами. Первый способ заключается в нажатии крестика в правом верхнем углу главной формы. Второй способ – это нажатие кнопки «Закрыть» в главном меню.

**5.4 Использование системы справочной информации**

Справочную систему можно запустить с помощью пункта меню Справка, которая находится в панели инструментов главной формы по пути «Справка» - «Справка проекта».

**Заключение**

В наше время существует множество современных игр, но иногда появляется желание сыграть в старого «Марио».

Разработанное игровое приложение «Марио» предоставит пользователям возможность сыграть в того самого «Марио».

Задача была выполнена с помощью среды разработки Delphi 12. Для разработки программы использовались различные инструменты и средства, такие как Help + Manual 9 для создания справок, Word 2022 для написания документации, PowerPoint 2023 для создания отчётной презентации, Smart Install Maker для создания инсталлятора. Это позволило создать полноценное десктопное приложение.

В ходе разработки программы были найдены следующие нетрадиционные способы решения задачи: использование черно белой картинки как коллайдера, после чего была наложена цветная картинка.

Программа может быть модифицирована для добавления новых функций и материалов, а также для адаптации к другим платформам и устройствам. Она может быть расширена для включения дополнительных функций, так как писалась с ориентиром на гибкость и комфортную разработку.

**Список использованных источников**

1. Марио - Википедия [Электронный ресурс] – Режим доступа:https: //ru.wikipedia.org/wiki/Марио– Дата доступа: 19.06.2024.

2. Базовые компоненты Delphi [Электронный ресурс] – Режим доступа https://www.youtube.com/results?search\_query=Базовые+компоненты+Delphi+ – Дата доступа: 21.06.2024.

**Приложение А**

**Листинг**

unit Unit1;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.MPlayer,

Vcl.Imaging.jpeg, Vcl.ExtCtrls, Vcl.Buttons, Vcl.Imaging.pngimage;

type

TForm1 = class(TForm)

MediaPlayer1: TMediaPlayer;

Image1: TImage;

Image2: TImage;

Image3: TImage;

SpeedButton1: TSpeedButton;

SpeedButton2: TSpeedButton;

procedure FormCreate(Sender: TObject);

procedure MediaPlayer1Notify(Sender: TObject);

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

procedure FormActivate(Sender: TObject);

procedure SpeedButton1Click(Sender: TObject);

procedure SpeedButton2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

uses Main, Unit2;

procedure TForm1.FormActivate(Sender: TObject);

begin

ScreenForm.Close;

end;

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

begin

Form2.CLose;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

MediaPlayer1.FileName := 'mario.mp3';

MediaPlayer1.Open;

MediaPlayer1.Play;

end;

procedure TForm1.MediaPlayer1Notify(Sender: TObject);

begin

with MediaPlayer1 do

if NotifyValue = nvSuccessful then

begin

Notify := True;

Play;

end;

end;

procedure TForm1.SpeedButton1Click(Sender: TObject);

begin

ScreenForm.Show;

Form1.Hide;

end;

procedure TForm1.SpeedButton2Click(Sender: TObject);

begin

Form1.Close;

Form2.Close;

ScreenForm.Close;

end;

end.

unit Unit2;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.ComCtrls,Vcl.Imaging.GIFImg;

type

TForm2 = class(TForm)

ProgressBar1: TProgressBar;

Timer1: TTimer;

Image1: TImage;

procedure Timer1Timer(Sender: TObject);

procedure FormActivate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

implementation

{$R \*.dfm}

uses Unit1;

procedure TForm2.FormActivate(Sender: TObject);

begin

( Image1.Picture.Graphic as TGIFImage ).Animate := True;// gets it goin'

( Image1.Picture.Graphic as TGIFImage ).AnimationSpeed:= 100;// adjust your speed

Form2.DoubleBuffered := True;

end;

procedure TForm2.Timer1Timer(Sender: TObject);

begin

ProgressBar1.Position:=ProgressBar1.Position+10;

if ProgressBar1.Position=100 then

begin

Timer1.Enabled:=False;

Form1.Show;

Form2.Hide;

end;

end;

end.

unit Main;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

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

type

TScreenForm = class(TForm)

RenderTimer: TTimer;

Display: TPaintBox;

procedure DisplayPaint(Sender: TObject);

procedure RenderTimerTimer(Sender: TObject);

procedure FormResize(Sender: TObject);

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

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

FBackBuffer: TBitmap;

FLevel: TBitmap;

FStaticCollision: TBitmap;

FDynamicCollision: array[Boolean] of TBitmap;

//first dimension is for different characters

//second dimension represents the states:

//0 = stand, 1 = walk, 2 = jump/InAir

FSprites: array[0..13] of array[0..2] of TBitmap;

FFlippedSprites: array[0..13] of array[0..2] of TBitmap;

FCamera\_X: Integer;

FCamera\_Y: Integer;

FNextCameraX: Integer;

FNextCameraY: Integer;

FScreenWidth: Integer;

FScreenHeight: Integer;

FFrameCounter: Integer;

FCurrentDC: Boolean;

FTargetDimensions: TRect;

FNumActiveScreens: Integer;

FOldDimension: TRect;

FOldStyle: TFormBorderStyle;

FImageFolder: string;

procedure LoadSprites;

procedure LoadSprite(AIndex: Integer; AStates: array of string);

procedure DrawRect(ATarget, ASource: TCanvas; AX, AY: Integer; ASRect: TRect);

public

{ Public declarations }

constructor Create(AOwner: TComponent); override;

procedure ToggleFullScreen;

end;

var

ScreenForm: TScreenForm;

implementation

uses

IOUtils,

PNGImage,

Level\_1\_1.Entities,

SyncObjs,

Math, Unit1, Unit2;

const

CSpriteWidth = 16;

CSpriteHeight = 16;

CHalfSpriteWidth = 8;

CHalfSpriteHeight = 8;

CNoDynCollision = clWhite;

CResolutionX = 256;

CResolutionY = 240;

{$R \*.dfm}

{ TScreenForm }

constructor TScreenForm.Create(AOwner: TComponent);

begin

inherited;

if TDirectory.Exists('Images') then

FImageFolder := 'Images'

else

FImageFolder := '..\..\Images';

FBackBuffer := TBitmap.Create();

FBackBuffer.SetSize(256, 240);

FBackBuffer.PixelFormat := pf32bit;

FLevel := TBitmap.Create();

FLevel.LoadFromFile(TPath.Combine(FImageFolder, 'Level-1-1.bmp'));

FLevel.PixelFormat := pf32bit;

FStaticCollision := TBitmap.Create();

FStaticCollision.LoadFromFile(TPath.Combine(FImageFolder, 'Level-1-1-Collision.bmp'));

FStaticCollision.PixelFormat := pf32bit;

FDynamicCollision[False] := TBitmap.Create();

FDynamicCollision[False].SetSize(FStaticCollision.Width, FStaticCollision.Height);

FDynamicCollision[False].PixelFormat := pf32bit;

FDynamicCollision[False].Canvas.Brush.Color := CNoDynCollision;

FDynamicCollision[False].Canvas.Pen.Color := CNoDynCollision;

FDynamicCollision[False].Canvas.FillRect(FDynamicCollision[False].Canvas.ClipRect);

FDynamicCollision[True] := TBitmap.Create();

FDynamicCollision[True].SetSize(FStaticCollision.Width, FStaticCollision.Height);

FDynamicCollision[True].PixelFormat := pf32bit;

FDynamicCollision[True].Canvas.Brush.Color := CNoDynCollision;

FDynamicCollision[True].Canvas.Pen.Color := CNoDynCollision;

FDynamicCollision[True].Canvas.FillRect(FDynamicCollision[True].Canvas.ClipRect);

FNumActiveScreens := 1;

ClientHeight := 768;//FBackBuffer.Height \* FNumActiveScreens;

ClientWidth := 1024;//FBackBuffer.Width;

LoadSprites();

FScreenWidth := FBackBuffer.Width;

FScreenHeight := FBackBuffer.Height;

end;

procedure TScreenForm.DisplayPaint(Sender: TObject);

var

LSecondTarget: TRect;

begin

Display.Canvas.CopyMode := SRCCOPY;

Display.Canvas.StretchDraw(FTargetDimensions, FBackBuffer);

if FNumActiveScreens > 1 then

begin

LSecondTarget.Top := FTargetDimensions.Height;

LSecondTarget.Bottom := LSecondTarget.Top + LSecondTarget.Top;

LSecondTarget.Left := FTargetDimensions.Left;

LSecondTarget.Right := LSecondTarget.Left + FTargetDimensions.Width;

Display.Canvas.CopyRect(LSecondTarget,

FStaticCollision.Canvas, Rect(FCamera\_X, FCamera\_Y, FCamera\_X + FBackBuffer.Width, FBackBuffer.Height));

Display.Canvas.CopyMode := SRCINVERT;

Display.Canvas.CopyRect(LSecondTarget,

FDynamicCollision[not FCurrentDC].Canvas, Rect(FCamera\_X, FCamera\_Y, FCamera\_X + FBackBuffer.Width, FBackBuffer.Height));

end;

end;

procedure TScreenForm.DrawRect(ATarget, ASource: TCanvas; AX, AY: Integer;

ASRect: TRect);

var

LBlend: BLENDFUNCTION;

begin

LBlend.BlendOp := AC\_SRC\_OVER;

LBlend.BlendFlags := 0;

LBlend.SourceConstantAlpha := 255;

LBlend.AlphaFormat := AC\_SRC\_ALPHA;

Winapi.Windows.AlphaBlend(ATarget.Handle, AX, AY, ASRect.Width, ASRect.Height,

ASource.Handle, ASRect.Left, ASRect.Top, ASRect.Width, ASRect.Height, LBlend)

end;

procedure TScreenForm.FormCreate(Sender: TObject);

begin

if ScreenForm.visible then

begin

form1.Close;

form2.close;

end;

end;

procedure TScreenForm.FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if (Key = VK\_RETURN) and (Shift = [ssAlt]) then

ToggleFullScreen();

if Key = VK\_F1 then

begin

if FNumActiveScreens = 1 then

FNumActiveScreens := 2

else

FNumActiveScreens := 1;

FormResize(Self);

end;

end;

procedure TScreenForm.FormResize(Sender: TObject);

var

LWidth, LHeight: Integer;

begin

if (ClientHeight div FNumActiveScreens) < ClientWidth then

begin

LHeight := ClientHeight div FNumActiveScreens;

LWidth := Trunc(LHeight / CResolutionY \* CResolutionX);

end

else

begin

LWidth := ClientWidth;

LHeight := Trunc(LWidth / CResolutionX \* CResolutionY);

end;

FTargetDimensions.Left := (ClientWidth - LWidth) div 2;

FTargetDimensions.Top := 0;

FTargetDimensions.Right := FTargetDimensions.Left + LWidth;

FTargetDimensions.Bottom := FTargetDimensions.Top + LHeight;

end;

procedure TScreenForm.ToggleFullScreen;

begin

if BorderStyle <> bsNone then

begin

FOldStyle := BorderStyle;

FOldDimension.Location := Point(Left, Top);

FOldDimension.Size := TSize.Create(ClientWidth, ClientHeight);

BorderStyle := bsNone;

Left := 0;

Top := 0;

Width := Screen.Width;

Height := Screen.Height;

end

else

begin

BorderStyle := FOldStyle;

Left := FOldDimension.Left;

Top := FOldDimension.Top;

ClientWidth := FOldDimension.Width;

ClientHeight := FOldDimension.Height;

end;

end;

procedure TScreenForm.LoadSprite(AIndex: Integer; AStates: array of string);

var

LTemp: TPngImage;

i, LState: Integer;

begin

LTemp := TPngImage.Create();

try

LState := 0;

for i := LState to High(FSprites[0]) do

begin

if (i = LState) and (AStates[LState] <> '') then

begin

LTemp.LoadFromFile(TPath.Combine(FImageFolder, AStates[LState]));

end;

FSprites[AIndex, i] := TBitmap.Create();

FSprites[AIndex, i].Assign(LTemp);

FFlippedSprites[AIndex][i] := TBitmap.Create();

FFlippedSprites[AIndex][i].SetSize(LTemp.Width, LTemp.Height);

FFlippedSprites[AIndex][i].PixelFormat := pf32bit;

FFlippedSprites[AIndex][i].Canvas.CopyRect(Rect(FSprites[AIndex][i].Width - 1, 0, -1, FSprites[AIndex][i].Height), FSprites[AIndex][i].Canvas, FSprites[AIndex][i].Canvas.ClipRect);

if (LState < High(AStates)) then

Inc(LState);

end;

finally

LTemp.Free;

end;

end;

procedure TScreenForm.LoadSprites;

begin

LoadSprite(0,['Mario\_Stand.png', 'Mario\_Walk.png', 'Mario\_Jump.png']);

LoadSprite(1, ['Mario\_Dead.png']);

LoadSprite(2, ['Brick.png']);

LoadSprite(3, ['Gumba\_Walk.png']);

LoadSprite(4, ['Gumba\_Dead.png']);

LoadSprite(5, ['ItemBlock.png']);

LoadSprite(6, ['ItemBlock\_Empty.png']);

LoadSprite(7, ['Coin\_Spinning.png']);

LoadSprite(8, ['Koopa\_Walk.png']);

LoadSprite(9, ['Koopa\_Shell.png']);

LoadSprite(10, ['GameOver.png']);

LoadSprite(11, ['StartScreen.png']);

LoadSprite(12, ['Finish.png']);

LoadSprite(13, ['Null.png']);

end;

const

CCameraDeadZone = 80;

CActivityBorder = 40;

CStand = 0;

CWalk = 1;

CJump = 2;

CGravity = 0.3;

CEdgeIdent = 2;

CDeadZoneMin = 240;

CDeadZoneMax = 260;

procedure TScreenForm.RenderTimerTimer(Sender: TObject);

var

i: Integer;

LDummy: Boolean;

LTopEdgeLeft, LTopEdgeRight, LLeftEdgeTop, LLeftEdgeBottom, LRightEdgeTop, LRightEdgeBottom, LBottomEdgeLeft, LBottomEdgeRight: Integer;

LX, LY: Integer;

LRect: TRect;

LSprite: TBitmap;

LSpriteState, LSpriteFrame, LFrameCount, LFrameWidth: Integer;

LDynLeft, LDynRight, LDynTop, LDynDown, LTargetEnt, LNextEntity: Integer;

LBBLeft, LBBRight, LBBTop, LBBBottom: Integer;

begin

//reset dynamic collisionmask

FDynamicCollision[not FCurrentDC].Canvas.Brush.Color := CNoDynCollision;

FDynamicCollision[not FCurrentDC].Canvas.Pen.Color := CNoDynCollision;

FDynamicCollision[not FCurrentDC].Canvas.FillRect(FDynamicCollision[not FCurrentDC].Canvas.ClipRect);

//Draw Level

FCamera\_X := FNextCameraX;

FCamera\_Y := FNextCameraY;

FBackBuffer.Canvas.Draw(-FCamera\_X, -FCamera\_Y, FLevel);

for i := Low(GEntity) to High(GEntity) do

begin

//the following giant IF is executed per Entity. It evaluates if the Entity must be updated/painted

//and executes it's logik at the same run!

//execute only if active and has input or is near to the camera

if (GEntity[i].Active <> 0) and (GEntity[i].Input or (bfAlwaysUpdate in GEntity[i].BehaviorFlags) or (((GEntity[i].X - FCamera\_X) >= -CActivityBorder) and ((GEntity[i].X - FCamera\_X) < FScreenWidth + CActivityBorder))) and Boolean(Trunc(

//reset some values

TInterlocked.Exchange(LTargetEnt, CNoDynCollision)

//get input

+ Integer(GEntity[i].Input and Boolean(Trunc(TInterlocked.Exchange(GEntity[i].Vel\_X, 2\*((GetAsyncKeyState(VK\_RIGHT) shr 31 and 1) - (GetAsyncKeyState(VK\_LEFT) shr 31 and 1))))))

+ Integer(GEntity[i].Input and (GEntity[i].DownBlocked <> 0) and Boolean(Trunc(TInterlocked.Exchange(GEntity[i].Vel\_Y, 6.5 \* (GetAsyncKeyState(VK\_SPACE) shr 31 and 1)))))

//apply velocity

+ Integer( (((GEntity[i].LeftBlocked = 0) and (GEntity[i].Vel\_X < 0)) or ((GEntity[i].RightBlocked = 0) and (GEntity[i].Vel\_X > 0))) and (TInterlocked.Exchange(GEntity[i].X, GEntity[i].X + GEntity[i].Vel\_X) \* 0 = 0))

//apply gravity

+ TInterlocked.Exchange(GEntity[i].Vel\_Y, Max(-5, GEntity[i].Vel\_Y - CGravity \* GEntity[i].Gravity))

+ Integer((((GEntity[i].DownBlocked = 0) and (GEntity[i].Vel\_Y < 0)) or ((GENtity[i].UpBlocked = 0) and (GEntity[i].Vel\_Y > 0))) and

Boolean(Trunc(

TInterlocked.Exchange(GEntity[i].Y, GEntity[i].Y - GEntity[i].Vel\_Y)

+ TInterlocked.Exchange(GEntity[i].InAirTimer, GEntity[i].InAirTimer + 1)

))

)

//////Collisiooncode...loooong!

//Trunc current position to pixels

+ TInterlocked.Exchange(LX, Trunc(GEntity[i].X))

+ TInterlocked.Exchange(LY, Trunc(GEntity[i].Y))

+ TInterlocked.Exchange(LBBLeft, LX - GEntity[i].bbWidth div 2)

+ TInterlocked.Exchange(LBBRight, LX + GEntity[i].bbWidth div 2 - 1)

+ TInterlocked.Exchange(LBBTop, LY - GEntity[i].bbHeight div 2)

+ TInterlocked.Exchange(LBBBottom, LY + GEntity[i].bbHeight div 2 - 1)

//do checks only if we are alive and do not ignore it!

+ Integer(not (bfIgnoreCollision in GEntity[i].BehaviorFlags) and (GEntity[i].Live > 0)

and Boolean(Trunc(

//collision checks

//static collision

//get collision values of edges of BoundingBox

//TopEdge

+ TInterlocked.Exchange(LTopEdgeLeft, FStaticCollision.Canvas.Pixels[LBBLeft + CEdgeIdent, LBBTop])

+ TInterlocked.Exchange(LTopEdgeRight, FStaticCollision.Canvas.Pixels[LBBRight - CEdgeIdent, LBBTop])

//LeftEdge

+ TInterlocked.Exchange(LLeftEdgeTop, FStaticCollision.Canvas.Pixels[LBBLeft, LBBTop + CEdgeIdent])

+ TInterlocked.Exchange(LLeftEdgeBottom, FStaticCollision.Canvas.Pixels[LBBLeft, LBBBottom - CEdgeIdent])

//RightEdge

+ TInterlocked.Exchange(LRightEdgeTop, FStaticCollision.Canvas.Pixels[LBBRight, LBBTop + CEdgeIdent])

+ TInterlocked.Exchange(LRightEdgeBottom, FStaticCollision.Canvas.Pixels[LBBRight, LBBBottom - CEdgeIdent])

//BottomEdge

+ TInterlocked.Exchange(LBottomEdgeLeft, FStaticCollision.Canvas.Pixels[LBBLeft + CEdgeIdent, LBBBottom])

+ TInterlocked.Exchange(LBottomEdgeRight, FStaticCollision.Canvas.Pixels[LBBRight - CEdgeIdent, LBBBottom])

//combine 2 corners each edge to determine if walking into the given direction is possible

+ TInterlocked.Exchange(GEntity[i].LeftBlocked, LLeftEdgeTop + LLeftEdgeBottom)

+ TInterlocked.Exchange(GEntity[i].RightBlocked, LRightEdgeTop + LRightEdgeBottom)

+ TInterlocked.Exchange(GEntity[i].UpBlocked, LTopEdgeLeft + LTopEdgeRight)

+ TInterlocked.Exchange(GEntity[i].DownBlocked, LBottomEdgeLeft + LBottomEdgeRight)

//Dynamic Collision

//get collision values of edges of BoundingBox

//TopEdge

+ TInterlocked.Exchange(LTopEdgeLeft, FDynamicCollision[FCurrentDC].Canvas.Pixels[LBBLeft + CEdgeIdent, LBBTop])

+ TInterlocked.Exchange(LTopEdgeRight, FDynamicCollision[FCurrentDC].Canvas.Pixels[LBBRight - CEdgeIdent, LBBTop])

//LeftEdge

+ TInterlocked.Exchange(LLeftEdgeTop, FDynamicCollision[FCurrentDC].Canvas.Pixels[LBBLeft, LBBTop + CEdgeIdent])

+ TInterlocked.Exchange(LLeftEdgeBottom, FDynamicCollision[FCurrentDC].Canvas.Pixels[LBBLeft, LBBBottom - CEdgeIdent])

//RightEdge

+ TInterlocked.Exchange(LRightEdgeTop, FDynamicCollision[FCurrentDC].Canvas.Pixels[LBBRight, LBBTop + CEdgeIdent])

+ TInterlocked.Exchange(LRightEdgeBottom, FDynamicCollision[FCurrentDC].Canvas.Pixels[LBBRight, LBBBottom - CEdgeIdent])

//BottomEdge

+ TInterlocked.Exchange(LBottomEdgeLeft, FDynamicCollision[FCurrentDC].Canvas.Pixels[LBBLeft + CEdgeIdent, LBBBottom])

+ TInterlocked.Exchange(LBottomEdgeRight, FDynamicCollision[FCurrentDC].Canvas.Pixels[LBBRight - CEdgeIdent, LBBBottom])

//combine 2 corners each edge to determine if walking into the given direction is possible

+ TInterlocked.Exchange(LDynLeft, IfThen((LLeftEdgeTop <> CNoDynCollision) and (LLeftEdgeTop <> i), LLeftEdgeTop, LLeftEdgeBottom))

+ TInterlocked.Exchange(LDynRight, IfThen((LRightEdgeTop <> CNoDynCollision) and (LRightEdgeTop <> i), LRightEdgeTop, LRightEdgeBottom))

+ TInterlocked.Exchange(LDynTop, IfThen((LTopEdgeLeft <> CNoDynCollision) and (LTopEdgeLeft <> i), LTopEdgeLeft, LTopEdgeRight))

+ TInterlocked.Exchange(LDynDown, IfThen((LBottomEdgeLeft <> CNoDynCollision) and (LBottomEdgeLeft <> i), LBottomEdgeLeft, LBottomEdgeRight))

// //if collided and not collided with self, add to entity collision state

+ TInterlocked.Exchange(GEntity[i].LeftBlocked, GEntity[i].LeftBlocked + IfThen((LDynLeft <> CNoDynCollision) and (LDynLeft <> i), LDynLeft, 0))

+ TInterlocked.Exchange(GEntity[i].RightBlocked, GEntity[i].RightBlocked + IfThen((LDynRight <> CNoDynCollision) and (LDynRight <> i), LDynRight, 0))

+ TInterlocked.Exchange(GEntity[i].UpBlocked, GEntity[i].UpBlocked + IfThen((LDynTop <> CNoDynCollision) and (LDynTop <> i), LDynTop, 0))

+ TInterlocked.Exchange(GEntity[i].DownBlocked, GEntity[i].DownBlocked + IfThen((LDynDown <> CNoDynCollision) and (LDynDown <> i), LDynDown, 0))

// //determine entity we collided with if exists

+ TInterlocked.Exchange(LTargetEnt, IfThen((LDynRight <> i) and (LDynRight <> CNoDynCollision), LDynRight, CNoDynCollision))

+ TInterlocked.Exchange(LTargetEnt, IfThen((LDynLeft <> i) and (LDynLeft <> CNoDynCollision), LDynLeft, LTargetEnt))

+ TInterlocked.Exchange(LTargetEnt, IfThen((LDynDown <> i) and (LDynDown <> CNoDynCollision), LDynDown, LTargetEnt))

+ TInterlocked.Exchange(LTargetEnt, IfThen((LDynTop <> i) and (LDynTop <> CNoDynCollision), LDynTop, LTargetEnt))

)))

//end of collisioncode!!!

//positional correction in case we are inside an obstacle

//do not correct Horizontal position if fallspeed was higher then walkspeed and we collided with floor to avoid being pushed of blocks we jumped on

//wen just a few pixels on the blocks

+ Integer(((Abs(GEntity[i].Vel\_X) >= Abs(GEntity[i].Vel\_Y)) or (GEntity[i].DownBlocked = 0)) and Boolean(

Integer( (GEntity[i].LeftBlocked <> 0) and Boolean(Trunc(TInterlocked.Exchange(GEntity[i].X, GEntity[i].X + 1))))

+ Integer( (GEntity[i].RightBlocked <> 0) and Boolean(Trunc(TInterlocked.Exchange(GEntity[i].X, GEntity[i].X - 1))))

))

+ Integer( (GEntity[i].UpBlocked <> 0) and Boolean(Trunc(TInterlocked.Exchange(GEntity[i].Y, GEntity[i].Y + 1))))

+ Integer( (GEntity[i].DownBlocked <> 0) and Boolean(Trunc(TInterlocked.Exchange(GEntity[i].Y, LY - 1))))

//bounce/reset velocity when colliding to reset falling momentum

+ Integer( (((GEntity[i].LeftBlocked <> 0) and (GEntity[i].Vel\_X < 0)) or ((GEntity[i].RightBlocked <> 0) and (GEntity[i].Vel\_X > 0)))

and Boolean(Trunc(

TInterlocked.Exchange(GEntity[i].Vel\_X, GEntity[i].Vel\_X \* -GEntity[i].BounceX)

)))

+ Integer( (((GEntity[i].UpBlocked <> 0) and (GEntity[i].Vel\_Y > 0)) or ((GEntity[i].DownBlocked <> 0) and (GEntity[i].Vel\_Y < 0)))

and Boolean(Trunc(

TInterlocked.Exchange(GEntity[i].Vel\_Y, GEntity[i].Vel\_Y \* -GEntity[i].BounceY))

+ TInterlocked.Exchange(GEntity[i].InAirTimer, 0)

)

)

+ TInterlocked.Increment(GEntity[i].LastColliderTime)

//if we collided with another entity(which we didn collide with in the last frame), which is alive and not in our team, attack and take damage!

+ Integer((LTargetEnt <> CNoDynCollision) and (GEntity[LTargetEnt].Active <> 0) and (GEntity[LTargetEnt].Team <> GEntity[i].Team)

and (((GEntity[i].LastCollider <> LTargetEnt) and (GEntity[LTargetEnt].LastCollider <> i))

or ((GEntity[i].LastColliderTime > 5) and (GEntity[LTargetEnt].LastColliderTime > 5))

)

and Boolean(Trunc(

+ Integer((LTargetEnt = LDynTop) and (GEntity[LTargetEnt].Live > 0) and Boolean(Trunc(

TInterlocked.Add(GEntity[LTargetEnt].Live, -GEntity[i].DamageTop\*GEntity[LTargetEnt].VulnerableBottom)

+ TInterlocked.Add(GEntity[i].Live, -GEntity[LTargetEnt].DamageBottom\*GEntity[i].VulnerableTop)

)))

+ Integer((LTargetEnt = LDynDown) and (GEntity[LTargetEnt].Live > 0) and Boolean(Trunc(

TInterlocked.Add(GEntity[LTargetEnt].Live, -GEntity[i].DamageBottom\*GEntity[LTargetEnt].VulnerableTop)

+ TInterlocked.Add(GEntity[i].Live, -GEntity[LTargetEnt].DamageTop\*GEntity[i].VulnerableBottom)

)))

+ Integer((LTargetEnt = LDynLeft) and (GEntity[LTargetEnt].Live > 0) and Boolean(Trunc(

TInterlocked.Add(GEntity[LTargetEnt].Live, -GEntity[i].DamageLeft\*GEntity[LTargetEnt].VulnerableRight)

+ TInterlocked.Add(GEntity[i].Live, -GEntity[LTargetEnt].DamageRight\*GEntity[i].VulnerableLeft)

)))

+ Integer((LTargetEnt = LDynRight) and (GEntity[LTargetEnt].Live > 0) and Boolean(Trunc(

TInterlocked.Add(GEntity[LTargetEnt].Live, -GEntity[i].DamageRight\*GEntity[LTargetEnt].VulnerableLeft)

+ TInterlocked.Add(GEntity[i].Live, -GEntity[LTargetEnt].DamageLeft\*GEntity[i].VulnerableRight)

)))

//store us as lastcollider in targetent so it does not interact with us on next frame

+ TInterlocked.Exchange(GEntity[LTargetEnt].LastCollider, i)

+ TInterlocked.Exchange(GEntity[LTargetEnt].LastColliderTime, 0)

+ TInterlocked.Exchange(GEntity[i].LastCollider, LTargetEnt)

+ TInterlocked.Exchange(GEntity[i].LastColliderTime, 0)

)))

//check DeadZone

+ Integer((GEntity[i].Y > CDeadZoneMin) and (GEntity[i].Y < CDeadZoneMax) and Boolean(

+ TInterlocked.Exchange(GEntity[i].Live, 0)

))

//camera movement

+ Integer((bfCameraFollows in GENtity[i].BehaviorFlags) and Boolean(

+ TInterlocked.Exchange(FNextCameraY, Trunc(GEntity[i].Y) div CDeadZoneMax \* FScreenHeight)

+ Integer((GEntity[i].X - FCamera\_X > FScreenWidth-CCameraDeadZone) and Boolean(TInterlocked.Exchange(FNextCameraX, Min(FLevel.Width - FScreenWidth, Trunc(FCamera\_X + (GEntity[i].X - FCamera\_X - FScreenWidth + CCameraDeadZone))))))

+ Integer((GEntity[i].X - FCamera\_X < CCameraDeadZone) and Boolean(TInterlocked.Exchange(FNextCameraX, Max(0, Trunc(FCamera\_X + (GEntity[i].X - FCamera\_X - CCameraDeadZone))))))

))

+ Integer((bfCameraCenters in GEntity[i].BehaviorFlags) and Boolean(

TInterlocked.Exchange(FNextCameraX, Trunc(GEntity[i].X - FScreenWidth div 2))

+ TInterlocked.Exchange(FNextCameraY, Trunc(GEntity[i].Y - FScreenHeight div 2))

))

+ TInterlocked.Add(GEntity[i].LiveTime, -1)

//check if we died and take actions if this is true

+ Integer(((GEntity[i].Live < 1) or (GEntity[i].LiveTime = 0)) and Boolean(Trunc(

Integer((GEntity[i].LiveTime = 0) and Boolean(TInterlocked.Exchange(LNextEntity, GEntity[i].ReplaceOnTimeOut)))

+ Integer((GEntity[i].Live < 1) and Boolean(TInterlocked.Exchange(LNextEntity, GEntity[i].ReplaceOnDead)))

//deactivate us so we are removed next frame

+ TInterlocked.Exchange(GEntity[i].Active, 0)

//replace with entity if specified

+ Integer((LNextEntity > 0) and Boolean(Trunc(

+ TInterlocked.Exchange(LNextEntity, i + LNextEntity)

+ TInterlocked.Exchange(GEntity[LNextEntity].Active, 1)

+ TInterlocked.Exchange(GEntity[LNextEntity].LastCollider, GEntity[i].LastCollider)

+ Integer((sfCopyPosition in GEntity[LNextEntity].SpawnFlags) and Boolean(Trunc(

+ TInterlocked.Exchange(GEntity[LNextEntity].X, GEntity[i].X)

+ TInterlocked.Exchange(GEntity[LNextEntity].Y, GEntity[i].Y)

)))

)))

)))

//check our current state(standing, walking, Jumping/InAir)

+ TInterlocked.Exchange(LSpriteState, CStand)

+ Integer((GEntity[i].Vel\_X <> 0) and Boolean(Trunc(TInterlocked.Exchange(LSpriteState, CWalk))))

+ Integer(((GEntity[i].DownBlocked = 0) and (GEntity[i].InAirTimer > 1)) and Boolean(Trunc(TInterlocked.Exchange(LSpriteState, CJump))))

//check if we need to use a flipped version of our image or the normal one

+ Integer((not GEntity[i].NoDirection) and (GEntity[i].InAirTimer <= 1) and (GEntity[i].Vel\_X <> 0)

and (Boolean((GEntity[i].Vel\_X < 0) and Boolean(Trunc(

+ TInterlocked.Exchange(GEntity[i].Orientation, -1) \* 0 - 1

)))

or Boolean(

TInterlocked.Exchange(GEntity[i].Orientation, 1)

))

)

+ Integer(((GEntity[i].Orientation < 0) and

Boolean(Trunc(

Integer(TInterlocked.Exchange<TBitmap>(LSprite, FFlippedSprites[GEntity[i].Sprite][LSpriteState])) \* 0 - 1

))) or

Boolean(Trunc(

Integer(TInterlocked.Exchange<TBitmap>(LSprite, FSprites[GEntity[i].Sprite][LSpriteState]))

))

)

//Calculate current Frame of sprite to display

+ Integer(((GEntity[i].Frames > 0) and Boolean(

TInterlocked.Exchange(LFrameCount, GEntity[i].Frames) \* 0 - 1))

or

Boolean(TInterlocked.Exchange(LFrameCount, (LSprite.Width div CSpriteWidth)))

)

+ TInterlocked.Exchange(LFrameWidth, LSprite.Width div LFrameCount)

+ TInterlocked.Exchange(LSpriteFrame, FFrameCounter div (50 div 16) mod LFrameCount)

//Translate EntityPosition

+ TInterlocked.Exchange(LX, Trunc(GEntity[i].X - LFrameWidth div 2 - FCamera\_X))

+ TInterlocked.Exchange(LY, Trunc(GEntity[i].Y - LSprite.Height div 2 - FCamera\_Y))

//recalculate the BB area

+ TInterlocked.Exchange(LBBLeft, Trunc(GEntity[i].X) - GEntity[i].bbWidth div 2)

+ TInterlocked.Exchange(LBBRight, Trunc(GEntity[i].X) + GEntity[i].bbWidth div 2)

+ TInterlocked.Exchange(LBBTop, Trunc(GEntity[i].Y) - GEntity[i].bbHeight div 2 )

+ TInterlocked.Exchange(LBBBottom, Trunc(GEntity[i].Y) + GEntity[i].bbHeight div 2)

)\*0-1)

then

begin

//Draw Entity

FDynamicCollision[not FCurrentDC].Canvas.Brush.Color := TColor(i);

FDynamicCollision[not FCurrentDC].Canvas.Pen.Color := TColor(i);

FDynamicCollision[not FCurrentDC].Canvas.Rectangle(LBBLeft, LBBTop, LBBRight, LBBBottom);

// //debugstuff

// FDynamicCollision[not FCurrentDC].Canvas.Pixels[LBBLeft + CEdgeIdent, LBBTop] := clFuchsia;

// FDynamicCollision[FCurrentDC].Canvas.Pixels[LBBRight - CEdgeIdent - 1, LBBTop] := clFuchsia;

// //LeftEdge

// FDynamicCollision[not FCurrentDC].Canvas.Pixels[LBBLeft, LBBTop + CEdgeIdent]:= clFuchsia;

// FDynamicCollision[not FCurrentDC].Canvas.Pixels[LBBLeft, LBBBottom - 1 - CEdgeIdent]:= clFuchsia;

// //RightEdge

// FDynamicCollision[not FCurrentDC].Canvas.Pixels[LBBRight - 1, LBBTop + CEdgeIdent]:= clFuchsia;

// FDynamicCollision[not FCurrentDC].Canvas.Pixels[LBBRight - 1, LBBBottom - CEdgeIdent - 1]:= clFuchsia;

// //BottomEdge

// FDynamicCollision[not FCurrentDC].Canvas.Pixels[LBBLeft + CEdgeIdent, LBBBottom - 1]:= clFuchsia;

// FDynamicCollision[not FCurrentDC].Canvas.Pixels[LBBRight - 1 - CEdgeIdent, LBBBottom - 1]:= clFuchsia;

// //debugstuff

DrawRect(FBackBUffer.Canvas, LSprite.Canvas, LX, LY,

Rect(LSpriteFrame\*LFrameWidth, 0, LSpriteFrame\*LFrameWidth + LFrameWidth, LSprite.Height));

end;

end;

Display.Repaint();

FCurrentDC := not FCurrentDC;

Inc(FFrameCounter);

end;

end.;

begin

Save;

end;

end.

unit Loading;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls, Vcl.ExtCtrls;

type

TForm3 = class(TForm)

ProgressBar1: TProgressBar;

Timer1: TTimer;

Image1: TImage;

procedure FormCreate(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

private

{ Private declarations }

public

await:integer;

end;

var

load: TForm3;

implementation

uses Main;

// активация формы при конце загрузки

procedure TForm3.Timer1Timer(Sender: TObject);

begin

ProgressBar1.StepBy(1);

if ProgressBar1.Position = ProgressBar1.Max then begin

await := await - 1;

if await = 0 then begin

Timer1.Enabled := False;

hide;

mf.Show();

end;

end;

end;

{$R \*.dfm}

// установка значений по умолчанию для загрузки и запуск таймера

procedure TForm3.FormCreate(Sender: TObject);

begin

await := 40;

ProgressBar1.Max := 100;

ProgressBar1.Position := 0;

Timer1.Enabled := True;

end;

end.