Учреждение образования

«Гродненский государственный политехнический колледж»

Специальность: 2–40 01 01  «Программное    обеспечение       информационных технологий»

Специализация: 2 – 40 01 01 35 «Программное обеспечение обработки

экономической и деловой информации»

Предмет: «Основы алгоритмизации и программирования»

Группа: ПЗТ – 38

**КУРСОВОЙ ПРОЕКТ**

Тема: «Разработка игрового приложение «Кладоискатель»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Разработал Д. В. Лебедь

Руководитель проекта Т. И. Шатова

2023

Учреждение образования «Гродненский государственный политехнический колледж»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УТВЕРЖДАЮ | | | | | |
| Председатель цикловой комиссии | | | | | |
|  |  | В.Д. Орехво | | | |
| (подпись) |  | (инициалы, фамилия) | | | |
|  | | | 20 |  | г. |

**ЗАДАНИЕ**

**по курсовому проекту (курсовой работе)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Учащемуся(имся) | | | | Лебедю Даниилу Владимировичу | | | | | | | | | | | | | |
| .(фамилия, собственное имя, отчество (если таковое имеется) | | | | | | | | | | | | | | | | | |
| курса | 2 | | | | | учебной группы | | | ПЗТ-38 | | | | | | | | |
| по учебному предмету, модулю | | | | | | Основы алгоритмизации и программирования | | | | | | | | | | | |
| Тема курсового проекта (курсовой работы) | | | | | | Разработка игрового приложения “Кладоискатель” | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| Исходные данные | | | | | |  | | | | | | | | | | | |
| правила игры | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| **Содержание курсового проекта (курсовой работы)** | | | | | | | | | | | | | | | | | |
| **Пояснительная записка** | | | | | | | | | | | | | | | | | |
| Содержание разделов | | | | | | | | | | | | | | Срок выполнения | | | |
| Введение. 1 Анализ задачи | | | | | | | | | | | | | | 24.05.2023 | | | |
| 2 Проектирование задачи | | | | | | | | | | | | | | 24.05.2023 | | | |
| 3 Реализация | | | | | | | | | | | | | | 07.06.2023 | | | |
| 4 Тестирование | | | | | | | | | | | | | | 07.06.2023 | | | |
| 5 Применение | | | | | | | | | | | | | | 13.06.2023 | | | |
| Заключение. Список использованных источников. Приложения | | | | | | | | | | | | | | 20.06.2023 | | | |
| **Графическая (практическая) часть курсового проекта (курсовой работы)** | | | | | | | | | | | | | | | | | |
| Лист 1- | | Схема работы системы (А1) | | | | | | | | | | | | 14.06.2023 | | | |
| Лист 2- | |  | | | | | | | | | | | | | | | |
| Дата выдачи | | | 05 мая | | 20 | | 23 | г. | | | | | | |
| Срок сдачи | | | 23 июня | | 20 | | 23 | г. | | | | | | |
|  | | | | | |  | |  | | |  |  | | | | |
| Руководитель  курсового проекта (курсовой работы) | | | | | |  |  | | |  | | | Т. И. Шатова | | | | | |
|  | | | | | |  | (подпись) | | | | | | | |  | (инициалы, фамилия) | |
| Подпись учащегося | | | | | | |  | | | | |

**Содержание**

Введение 4

*Изм.*

*Кол*

*Лист №док*

*Подпись*

*Дата*

*Лист*

*3*

*КП 2-40 01 01.35.38.13.23 ПЗ*

*Разраб.*

*Лебедь*

*Пров.*

*Шатова*

*Н. контр.*

*Утв.*

*Разработка игрового приложения ”Кладоискатель”*

*Стадия*

*Листов*

*38*

*УО ГГПК*

1 Анализ задачи 6

1.1 Постановка задачи 6

1.2 Инструменты разработки 6

1.3 Требования к приложению 7

2 Проектирование задачи 8

2.1 Организация данных 8

2.2 Процессы 8

2.3 Описание внешнего пользовательского интерфейса 9

3 Реализация 10

3.1 Структура программы 10

3.1.1 Структура и описание процедур и функций пользователя 10

3.1.2 Описание использованных компонентов 14

3.2 Спецификация программы 16

4 Тестирование .18

5 Применение 20

5.1 Общие сведения о программном продукте 20

5.2 Инсталляция 20

5.3 Выполнения программы 20

5.3.1 Запуск программы 20

5.3.2 Инструкция по работе с программой 20

5.3.3 Использование системы справочной информации 24

Заключение .25

Список использованных источников 26

Приложение А Листинг программы 27

**Введение**

На курсовое проектирование была поставлена задача, разработать на тему: Игровое приложение «Кладоискатель».

Создаваемое приложение будет рассчитана на любого рода пользователей и может использоваться в любой момент по желанию пользователя.

Далее приведём 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. 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 создаётся данный курсовой проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачей.

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 чётко 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 чётко описаны функции пользователя и их структура. Здесь можно будет найти таблицы, в которых будет представлена полная аннотация процедур, функций, компонентов и файлов, используемых в данном проекте.

Четвёртый 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, степень соответствия проектных решений поставленной задаче и возможный модификации проекта.

В “Литературе” будет приведён список используемых при разработке источников.

В приложениях к пояснительной записке будет приведён листинг программы с необходимыми комментариями.

Схема работы системы будет представлена в графической части.

**1 Анализ задачи**

# 1.1 Постановка задачи

**Наименование задачи:** Разработка игрового приложения «Кладоискатель».

**Цель разработки:** создание игрового приложения «Кладоискатель», которое рассчитано на людей, интересующихся различными играми, мотивируя их к изучению такой сферы в IT, как разработка видеоигр.

**Назначение:** для развлечения и отдыха, развития скорости реакции, демонстрацииреализации игрового приложения с графическим представлением.

**Предметная область:** вданной предметной области существуют видеоигры с схожими особенностями и механиками. Например, графический дизайн был вдохновлён серией игр “Heroes of Might and Magic”, а концепция прохождения подземелья и сражения с монстрами была вдохновлена серией игр “Diablo”.

**Периодичность использования:** может использоваться в любой момент по желанию пользователя.

**Входная, выходная и постоянная информация:** к входной информации можно отнести текстовый файл, содержащий карту. К выходной информации можно отнести сохранения. К постоянной информации можно отнести изображения и звуки.

Разрабатываемый программный продукт позволит выполнить следующие действия:

* самостоятельно изменить или создать свою карту;
* осуществить игровую сессию;
* сыграть в мини-игры.

# 1.2 Инструменты разработки

Для разработки данного проекта выбрана среда Delphi (Delphi 11), так как это среда объектно-ориентированного программирования, относящаяся к классу RAD – (Rapid Application Development – «Средство быстрой разработки приложений»), реализованная на Object Pascal. Используется для разработки визуализированного представления программного обеспечения.

Также для разработки программы необходимы:

* HTML Help Workshop –редактор для создания html-страниц и справок;
* Google-браузер – нужен для нахождения информации и картинок персонажей игры, фоновых изображений;
* Word 2021 – нужен для написания пояснительной записки;
* Power Point 2021 – нужен для создания отчётной презентации;
* Aseprite – нужен для создания и редактирования пиксельных изображений;
* Smart Install Maker – нужен для создания установщика.

При разработке данного программного продукта был использован ноутбук Lenovo со следующими характеристиками:

* Процессор: AMD Ryzen 5;
* ОЗУ: 8Gb;
* Память: Samsung SSD 256Gb;
* ОС – Windows 10.

# 1.3 Требования к приложению

На этапе исследования предметной области был установленный целый ряд требований, предъявляемые к разрабатываемой задаче.

При моделировании форм следует учесть такие моменты:

* интуитивно понятное управление;
* небольшое количество информации на игровом экране;
* небольшие окна формы;
* визуально понятный и приятный глазу интерфейс;
* наличие информации о функционале игры;
* увлекательный игровой процесс;

Приложение имеет небольшие требования к системе и может работать на большинстве персональных компьютеров.

При разработке интерфейса приложения использованы преимущественного неяркие оттенки. Каждое окно проекта имеет ясную визуальную иерархию своих элементов. Разработан понятный пользователю интерфейс, фрагменты текста располагаются на экране так, чтобы пользователь не испытывал какого-либо дискомфорта в плане восприятия игровой информации, отображённой на экране. Для объектов (персонажа и монстров) подобраны размеры, удобные для управления и игры.

Ошибки программы вследствие некорректного взаимодействия пользователя при работе с программой через графический интерфейс не должны влиять на работоспособность.

Надёжность программы должна быть на высоком уровне, не допуская ситуаций программных ошибок и некорректной работы приложения.

**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 три вида данных.

Первым видом являются данные, которые будут введены разработчиком на этапе реализации задачи. Сюда можно отнести изображения персонажей и объектов, описание, исходные коды теории в разделах.

Вторым видом данных, используемых в программе, является вводимая пользователем информация. К входной информации можно отнести сохранение, которое пользователь выбирает при загрузке игры, выбранная игрокам сложность и нажатия клавиш при игровом процессе.

Третьим видом данных является результат программы – например, сообщения о победе или поражении игрока.

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 её задачей будет являться развлечение и развитие реакции пользователя, а также привлечения его внимания к индустрии видеоигр.

Для реализации задач будут использоваться процедуры. С помощью процедур, например, будет осуществляться заполнение таблиц изображениями, проводится игровой процесс в мини-играх.

Игровой процесс на главной карте будет осуществляться путём перемещения персонажа игрока нажатиями клавиш клавиатуры. Игровой процесс в мини-играх будет осуществляться при помощи мыши.

Карта, противники и характеристики игрока загружается из текстового файла.

На этапе проектирования было важным рассмотреть состав, внешний вид и структуру используемых форм. Они должен быть интуитивно понятными и приятными для глаз.

**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.

Структура навигации по проекту представлена на рисунке 1:



Рисунок 1 – Навигация между формами программы

## 3 Реализация

### 3.1 Структура программы

Данный курсовой проект содержит 8 модулей. Далее рассмотрим назначение каждого модуля:

1. Модуль uGame – отвечает за создание карты, передвижение, открытие сундуков начало боя;
2. Модуль uInv – отвечает за инвентарь игрока;
3. Модуль uOrkB – отвечает за бой с орком.
4. Модуль uShroomB – отвечает за бой с грибником;
5. Модуль uSnakeB – отвечает за бой с змеёй;
6. Модуль uStart – главное меню приложения;
7. Модуль uStats – отвечает за отображение характеристик игрока
8. Модуль uTitle – отвечает за отображение заставки.

### 3.1.1 Структура и описание процедур и функций пользователя

Описание разработанных процедур находятся в таблице 1.

Таблица 1 – Процедуры и функции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Имя процедуры (функции) | В каком модуле находится | За каким компонентом закреплена | Назначение |
| 1 | 2 | 3 | 4 | 5 |
| 1 | procedure TitleTimerTimer(Sender: TObject); | uTitle.pas | TitleTimer | Управляет плавным появление и исчезновением заставки, переходом на следующую форму |
| 2 | procedure BStartClick(Sender: TObject); | uStart.pas | BStart | Запускает новую игру |
| 3 | procedure BExitClick(Sender: TObject); | uStart.pas | BExit | Выходит из программы |
| 4 | procedure DiffRadioGroupClick(Sender: TObject); | uStart.pas | DiffRadioGroup | Делает кнопку начала новой игры активной |
| 5 | procedure SaveComboBoxSelect(Sender: TObject); | uStart.pas | SaveComboBox | Делает кнопку загрузки игры активной |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Продолжение таблицы 1 | | | |
| 1 | 2 | 3 | 4 | 5 |
| 6 | procedure FormShow(Sender: TObject); | uStart.pas | - | Заполняет список сохранений, рисует фон |
| 7 | procedure BLoadClick(Sender: TObject); | uStart.pas | BLoad | Загружает сохранённую игру |
| 8 | procedure FormClose(Sender: TObject; var Action: TCloseAction); | uGame.pas | - | Переходит на стартовое меню |
| 9 | procedure FormCreate(Sender: TObject); | uGame.pas | - | Загружает карту, противников, инвентарь и характеристики персонажа игрока из файла сохранения или файла с начальной картой |
| 10 | procedure N1Click(Sender: TObject); | uGame.pas | GameMainmenu.N1 | Переходит на инвентарь |
| 11 | procedure N2Click(Sender: TObject); | uGame.pas | GameMainmenu.N2 | Переходит на характеристики |
| 12 | procedure N3Click(Sender: TObject); | uGame.pas | GameMainmenu.N3 | Сохраняет игру в текстовый файл |
| 13 | procedure N4Click(Sender: TObject); | uGame.pas | GameMainmenu.N4 | Открывает справку |
| 14 | constructor Create(HP:integer; Dmg:integer;  PosX, PosY:integer); | uGame.pas | Player | Создаёт объект класса Игрок |
| 15 | procedure Move(x,y:integer;  SMap:TStringGrid); | uGame.pas | Player | Перемещает игрока, начинает сражение с противником, открывает сундук или добавляет предмет, перемещает камеру, заканчивает игру победой |
| 16 | constructor Create(HP,dmg:integer; kind:string; PosX, PosY:integer); | uGame.pas | Enemy | Создаёт объект класса Противник |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Продолжение таблицы 1 | | | |
| 1 | 2 | 3 | 4 | 5 |
| 17 | procedure KeyPress(var Key: Char); override; | uGame.pas | StringGrid | Читает нажатую клавишу и передаёт направление движения процедуре Move |
| 18 | procedure DrawCell(ACol, ARow: Integer; ARect: TRect; AState: TGridDrawState); override; | uGame.pas | StringGrid | Читает содержимое ячейки таблицы и рисует на соответствую картинку на ней |
| 19 | procedure Loaded; override; | uGame.pas | StringGrid | Находит и привязывает список картинок к таблице |
| 20 | ItemListBoxClick(Sender: TObject); | uInv.pas | ItemListBox | Вставляет описание и картинку предмета в зависимости от выбранного предмета |
| 21 | procedure FormShow(Sender: TObject); | uStats.pas | - | Заполняет таблицу характеристик и делает картинки видимыми при наличии предметов в инвентаре |
| 22 | procedure EButtonClick(Sender: TObject); | uStats.pas | EButton | Закрывает форму |
| 23 | procedure FormCreate(Sender: TObject); | uShroomB.pas | - | Заполняет таблицу грибов нулями, заполняет таблицу характеристик, включает таймер |
| 24 | procedure TurnTimerTimer(Sender: TObject); | uShroomB.pas | TurnTimer | Создаёт грибы на поле, взрывает существующий гриб, играет соответствующий звук, убирает взрыв и убивает игрока, если у него закончилось здоровье |
| 25 | procedure Loaded; override; | uShroomB.pas | StringGrid | Привязывает список картинок к таблице грибов |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Продолжение таблицы 1 | | | |
| 1 | 2 | 3 | 4 | 5 |
| 26 | procedure PlayGridSelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean); | uShroomB.pas | PlayGrid | Запрещает щелкать за край таблицы |
| 27 | procedure FormClose(Sender: TObject; var Action: TCloseAction); | uShroomB.pas | - | Выключает таймер |
| 28 | procedure DrawCell(ACol, ARow: Integer; ARect: TRect; AState: TGridDrawState); override; | uShroomB.pas | StringGrid | Читает содержимое ячеек таблицы и рисует соответствующие картинки. |
| 29 | procedure MouseDown(Button: TMouseButton; Shift: TShiftState;X, Y: Integer); override; | uShroomB.pas | StringGrid | Убирает грибы, на которые нажимает игрок, играет соответствующий звук, наносит противнику урон, заканчивает бой победой, если у противника закончилось здоровье |
| 30 | procedure FormCreate(Sender: TObject); | uSnakeB.pas | - | Включает таймер и заполняет таблицу характеристик |
| 31 | procedure SnakeTimerTimer(Sender: TObject); | uSnakeB.pas | - | Поочерёдно меняет картинку стрелки |
| 32 | procedure FormClose(Sender: TObject; var Action: TCloseAction); | uSnakeB.pas | - | Выключает таймер |
| 33 | procedure AttackButtonClick(Sender: TObject); | uSnakeB.pas | - | Наносит урон игроку и змее в зависимости от положения стрелки, заканчивает бой победой или поражением |
| 34 | procedure FormCreate(Sender: TObject); | uOrkBattle.pas | - | Заполняет таблицу характеристик, выключает таймеры |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Продолжение таблицы 1 | | | |
| 1 | 2 | 3 | 4 | 5 |
| 35 | procedure DrumsImageClick(Sender: TObject); | uOrkBattle.pas | DrumsImage | Включает таймер и считает количество нажатий по барабану, играет соответствующий звук |
| 36 | procedure DrumsTimerTimer(Sender: TObject); | uOrkBattle.pas | DrumsTimer | После того, как у игрока заканчивается время переводит форму в режим отдыха, наносит игроку и противнику урон, заканчивает игру победой или поражением, в зависимости от того, у кого быстрее закончится здоровье |
| 37 | procedure RestTimerTimer(Sender: TObject); | uOrkBattle.pas | RestTimer | После окончания фазы отдыха снова переводит компоненты в активное состояние, обнуляет счётчики |
| 38 | procedure FormClose(Sender: TObject; var Action: TCloseAction); | uOrkBattle.pas | - | Выключает таймеры |

### 3.1.2 Описание использованных компонентов

Описание использованных для разработки приложения компонентов приводится в таблице 2.

Таблица 2 – Использованные компоненты

|  |  |  |  |
| --- | --- | --- | --- |
| № | Компонент | На какой форме расположен | Назначение |
| 1 | 1 | 2 | 3 |
| 1 | TitleImage | Title | Картинка заставки |
| 2 | TitleTimer | Title | Повышает и понижает прозрачность формы |
| 3 | BackGroundImage | Start | Используется как фон для формы |
| 4 | BExit | Start | Используется для выхода из формы |
| 5 | DiffRadioGroup | Start | Используется для выбора сложности |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Продолжение таблицы 2 | | |
| 1 | 2 | 3 | 4 |
| 6 | StartTitleImage | Start | Отображает картинку с названием игры |
| 7 | BLoad | Start | Используется для загрузки сохранения |
| 8 | SaveComboBox | Start | Используется для выбора файла сохранения |
| 9 | StartTitleImage | Start | Позволяет получить доступ к устройствам ввода информации |
| 10 | GameMainMenu | Game | Позволяет открыть инвентарь, характеристики, справку и сохранить игру |
| 11 | SpriteImageList | Game | Содержит изображения, которыми заполняется карта |
| 12 | SMap | Game | Используется как игровое поле |
| 13 | ItemListBox | Inventory | Содержит список предметов |
| 14 | ItemImage | Inventory | Отображает изображение предмета |
| 15 | ItemsImageList | Inventory | Содержит изображения предметов |
| 16 | ItemMemo | Inventory | Содержит описание предмета |
| 17 | PImage | Stats | Отображает изображение игрока |
| 18 | StatGrid | Stats | Отображает характеристики игрока |
| 19 | SwordImage | Stats | Отображает меч |
| 20 | AmuletImage | Stats | Отображает Амулет |
| 21 | EButton | Stats | Закрывает форму |
| 22 | PImage | OrkBattle | Отображает изображение игрока |
| 23 | StatGrid | OrkBattle | Отображает характеристики |
| 24 | EImage | OrkBattle | Отображает изображение противника |
| 25 | DrumsImage | OrkBattle | Отображает изображение барабана |
| 26 | DrumsBar | OrkBattle | Отображает количество нажатий |
| 27 | DrumsTimer | OrkBattle | Считает время, отведённое для нажатий |
| 28 | TimeLabel | OrkBattle | Отображает время, отведённое для нажатий |
| 29 | RestTimer | OrkBattle | Считает время, отведённое для отдыха |
| 30 | CountLabel | OrkBattle | Отображает количество нажатий |
| 31 | PlayGrid | ShroomBattle | Используется как игровое поле, на котором появляются грибы |
| 32 | MushroomImageList | ShroomBattle | Содержит изображения грибов |
| 33 | EImage | ShroomBattle | Отображает изображение противника |
| 34 | PImage | ShroomBattle | Отображает изображение игрока |
| 35 | StatGrid | ShroomBattle | Отображает характеристики |
| 36 | TurnTimer | ShroomBattle | Считает время появления и взрыва грибов |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Продолжение таблицы 2 | | |
| 1 | 2 | 3 | 4 |
| 37 | StatGrid | SnakeBattle | Отображает характеристики |
| 38 | EImage | SnakeBattle | Отображает изображение противника |
| 39 | PImage | SnakeBattle | Отображает изображение игрока |
| 40 | AttackButton | SnakeBattle | Используется как кнопка удара |
| 41 | SnakeTimer | SnakeBattle | Используется для изменения положения стрелки |
| 42 | SnakeImage | SnakeBattle | Отображает изображение стрелки со змеёй |
| 43 | SnakeImageList | SnakeBattle | Содержит изображения стрелок со змеёй в разных положениях |

### 3.2 Спецификация программы

Точное название проекта и его состав приводится в Таблице 3.

Таблица 3 – Спецификация программы

|  |  |  |
| --- | --- | --- |
| № | Имя файла | Назначение |
| 1 | 2 | 3 |
| 1 | Кладоискатель.exe | Исполняемый файл проекта |
| 2 | Кладоискатель.dpr | Файл проекта, связывает все файлы, из которых состоит приложение |
| 3 | UGame.pas | Файл программного модуля для формы Game |
| 4 | UInv.pas | Файл программного модуля для формы Inventory |
| 5 | UOrkB.pas | Файл программного модуля для формы OrkBattle |
| 6 | UShroomB.pas | Файл программного модуля для формы ShroomBattle |
| 7 | USnakeB.pas | Файл программного модуля для формы SnakeBattle |
| 8 | UStart.pas | Файл программного модуля для формы Start |
| 9 | UStats.pas | Файл программного модуля для формы Stats |
| 10 | UTitle.pas | Файл программного модуля для формы Title |
| 11 | UGame.dfm | Форма Game |
| 12 | UInv.dfm | Форма Inventory |
| 13 | UOrkB.dfm | Форма OrkBattle |
| 14 | UShroomB.dfm | Форма ShroomBattle |
| 15 | USnakeB.dfm | Форма SnakeBattle |
| 16 | UStart.dfm | Форма Start |
| 17 | UStats.dfm | Форма Stats |

|  |  |  |
| --- | --- | --- |
|  | Продолжение таблицы 3 | |
| 1 | 2 | 3 |
| 18 | UTitle.dfm | Форма Title |
| 19 | Кладоискатель\_\_Icon.ico | Иконка проекта |
| 20 | ProjectHelp.chm | Файл справки, содержит инструкции к игре |
| 21 | OriginalMap.txt | Текстовый файл, содержащий оригинальную карту |
| 22 | BattleVictory.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 23 | Death1.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 24 | Death2.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 25 | Death3.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 26 | Death4.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 27 | GainItem.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 28 | Hit1.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 29 | Hit2.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 30 | Hit3.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 31 | MushroomClear1.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 32 | MushroomClear2.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 33 | MushroomClear3.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 34 | MushroomClear4.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 35 | MushroomExplode1.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 36 | MushroomExplode2.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 37 | MushroomExplode3.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 38 | Snake1.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 39 | Snake2.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |
| 40 | Victory.wav | Звуковой файл, проигрываемый в ходе выполнения проекта |

**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 | 2 | 3 | 4 | 5 |
| 1 | Проверка запуска приложения | Открытие заставки и стартового меню | Открытие заставки и стартового меню | Выполнено |
| 2 | Проверка начала новой игры | Начало игры | Начало игры | Выполнено |
| 3 | Проверка инвентаря | Отображение предмета, изображения и описания этого предмета | Отображение предмета, изображения и описания этого предмета | Выполнено |
| 4 | Проверка справки | Открытие справочного файла | Открытие справочного файла | Выполнено |
| 5 | Проверка характеристик | Отображение характеристик | Отображение характеристик | Выполнено |
| 6 | Проверка сохранения | Создание файла сохранения | Создание файла сохранения | Выполнено |
| 7 | Проверка загрузки | Загрузка файла сохранения | Загрузка файла сохранения | Выполнено |
| 8 | Проверка открытия сундука | Получение предмета и очков | Получение предмета и очков | Выполнено |
| 9 | Проверка боя с орком | Корректное начало, процесс и окончание мини-игры | Корректное начало, процесс и окончание мини-игры | Выполнено |
| 10 | Проверка боя с змеёй | Корректное начало, процесс и окончание мини-игры | Корректное начало, процесс и окончание мини-игры | Выполнено |
| 11 | Проверка боя с грибником | Корректное начало, процесс и окончание мини-игры | Корректное начало, процесс и окончание мини-игры | Выполнено |
| 13 | Проверка кнопки выхода | Полное закрытие всех форм приложения | Полное закрытие всех форм приложения | Выполнено |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Продолжение таблицы 4 | | | | |
| 1 | 2 | 3 | 4 | 5 |
| 12 | Проверка взаимодействия с дверью в случае получения 1000 очков | Вывод сообщения о победе, количество очков и окончание игры | Вывод сообщения о победе, количество очков и окончание игры | Выполнено |
| 13 | Проверка проигрывания звуков | Звуки проигрались | Звуки проигрались | Выполнено |

При разработке программного продукта было решено множество проблем, например, некорректно заполнялась карта, неправильно записывался и читался файл сохранения, не отображались изображения. После тестирования и исправления проблем пользователь не столкнётся с ими.

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

**5 Руководство пользователя**

# 5.1 Общие сведения о программном продукте

Цель данного проекта заключается в развлечении пользователя, развитии его реакции и привлечении к индустрии видеоигр.

Создаваемое игровое приложение будет рассчитано на любого рода пользователей и может использоваться в любой момент по желанию пользователя.

Быстродействие любой программы во многом зависит от характеристик выбранного персонального компьютера: рабочей частоты процессора, объема оперативной памяти и т.д. Несмотря на все реализованные в ней задачи, она легко запускается и функционирует на любых машинах.

Тестирование проводилось на разных классах ЭВМ и данное игровое приложение стабильно работало на любом персональном компьютере. Программа разработана на ноутбуке со следующими характеристиками:

− процессор AMD Ryzen 5 3500U with Vega mobile GFX, 2,1 ГГц, ядер:

4, логических процессоров: 8

− объем ОЗУ 8Гб;

− Графический процессор AMD Radeon Vega 8 Graphics;

− операционная система Windows 10.

# 5.2 Инсталляция

Для того, чтобы установить программу необходимо запустить файл Setup.exe. Появится окно установки приложения “Кладоискатель”.

Затем достаточно следовать приведенной инструкции установки приложения.

# 5.3 Выполнение программы

## 5.3.1 Запуск программы

Для запуска данного приложения необходимо запустить исполняемый файл Кладоискатель.exe, который находится в папке Кладоискатель по пути, выбранному при установке.

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

После запуска приложения и просмотра заставки пользователь попадает на экран стартового меню (рисунок 2).

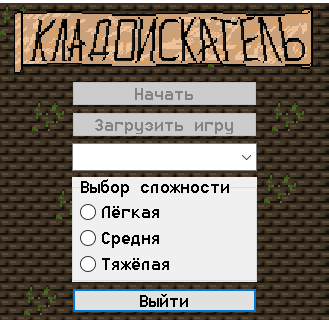


Рисунок 2 – Стартовое меню

На этом экране пользователь либо выбрать уровень сложности и нажать на кнопку “Начать”, либо выбрать файл сохранения и нажать на кнопку “Загрузить игру”. После этого пользователь попадает на главную игровую форму (либо начав игру сначала, либо продолжив своё сохранение) (рисунок 3).

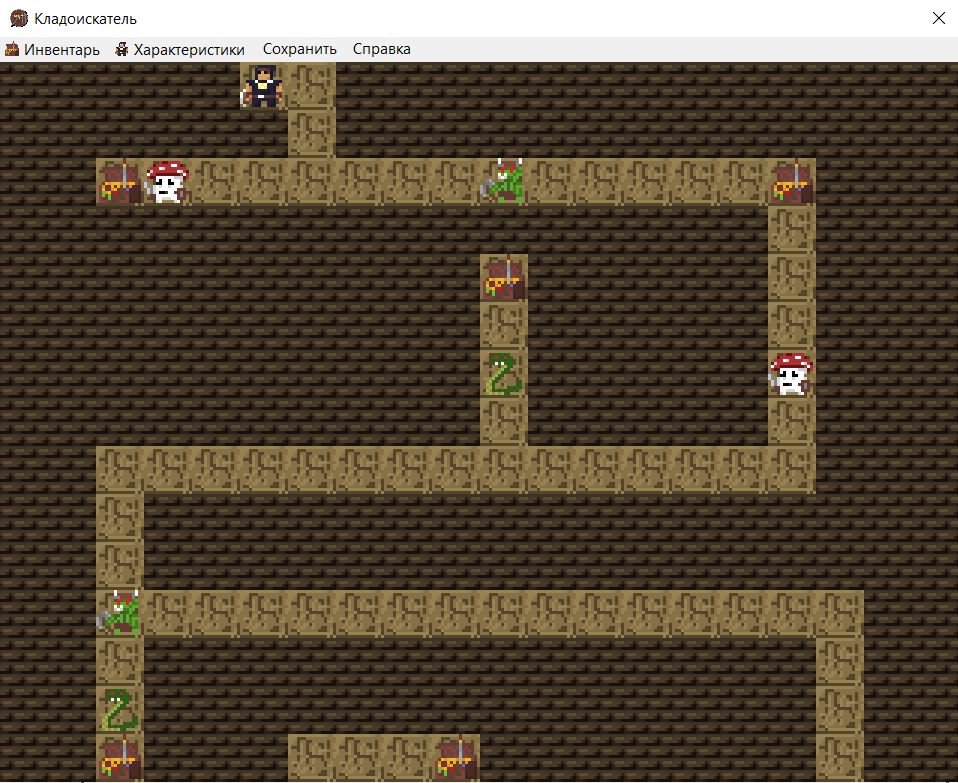


Рисунок 3 – Главная игровая форма

На этом этапе начинается сам игровой процесс. Пользователь может перемещать своего персонажа для того, чтобы взаимодействовать с объектами на карте. При взаимодействии с сундуком игрок получит очки и предмет, улучшающий его характеристики. Предмет попадает в инвентарь, который можно открыть, нажав кнопку “Инвентарь” на меню сверху (рисунок 4).



Рисунок 4 – Инвентарь

Так же игрок имеет 3 характеристики: здоровье, урон и очки, которые можно просмотреть, нажав на кнопку “Характеристики” на меню сверху (рисунок 5).

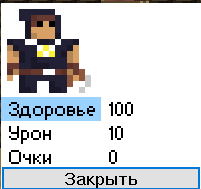


Рисунок 5 – Характеристики

Путь игрока преграждают монстры. При взаимодействии с ними начнётся одна из трёх мини-игр, в зависимости от типа монстра. Далее будет описана каждая мини-игра.

В битве с грибником игроку необходимо как можно быстрее нажимать на появляющиеся грибы. Если он успевает на них нажать, то противник получает урон, а если нет, то гриб взрывается и урон получает игрок (рисунок 6).



Рисунок 6 – Битва с грибником

В битве с орком игроку необходимо как можно быстрее нажимать на барабан. Таймер начинается только после первого удара. После окончания счета таймер игрок и противник получают урон, соответствующий количеству нажатий по барабану. После этого игроку дают несколько секунд на отдых. Бой представлен на рисунке 7.



Рисунок 7 – Битва с орком

В битве со змеёй игроку необходимо нажимать на кнопку удара тогда, когда стрелка указывает вверх. Чем ближе стрелка к верху, тем больше урона нанесёт игрок и тем меньше урона нанесёт противник и наоборот, если стрелка дальше от верха (рисунок 8).

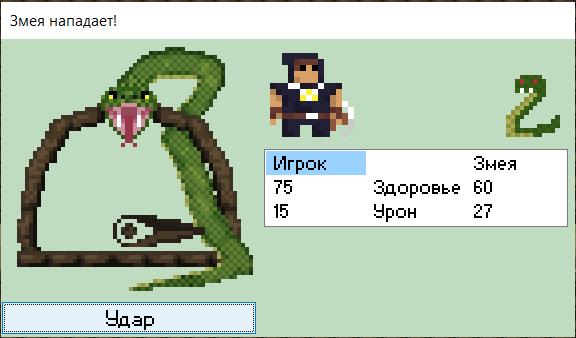


Рисунок 8 – Битва с змеёй

Если в ходе каждой из игр игрок теряет все здоровье, то он проигрывает и переносится обратно на стартовое меню. Если же здоровье заканчивается у противника, то игрок его побеждает и получает очки.

В конце уровня игрок найдёт дверь. Если при взаимодействии с ней у игрока не будет хотя бы 1000 очков, то он не сможет пройти. Если же у него есть 1000 очков, то он побеждает в игре и переносится обратно на стартовое меню.

Так же на меню сверху игрок может найти кнопки “Сохранение” и “Справка”, при нажатии по которым он сохранит свою игру или откроет справку соответственно.

## 5.3.3 Завершение работы с программой

Для завершения работы с программой игрок должен попасть в стартовое меню либо проиграв, либо победив, либо нажав на крестик на главной игровой форме. После этого игрок должен нажать на кнопку “Выход” на главной игровой форме (рисунок 9).

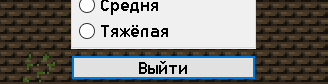


Рисунок 9 – Выход из приложения

# 5.4 Использование системы справочной информации

Справочную систему можно запустить с помощью пункта меню “Справка” на верху главной игровой формы (рисунок 10).

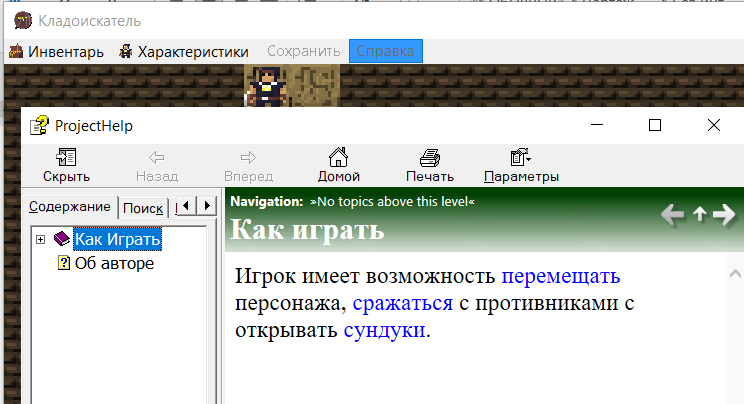


Рисунок 10 – Открытие справочной системы

**Заключение**

Для выполнение данного проекта была поставлена задача разработать игровое приложение “Кладоискатель”, целью которого является развлечение пользователя, развитие его реакции и привлечение его внимания к индустрии видеоигр.

При выполнении данного проекта возникали ошибки, все из которых были решены. Поставленную задачу можно считать выполненной успешно.

При разработке данного приложения использование сторонних материалов было минимальным, в результате чего все составляющие проекта, кроме звуков и компонентов Delphi, были разработаны самостоятельно.

Решения, принятые во время разработки проекта, полностью соответствуют поставленной задаче.

При разработке проекта были обнаружены нетрадиционные способы решения некоторых задач. Например, модификация стандартных классов Delphi вместо создания новых процедур.

Данный проект может быть модифицирован в дальнейшем. Например, добавлением самостоятельно созданных звуков и большего количества противников.

**Список использованных источников**

1. Форум для программистов StackOverflow.com [Электронный ресурс]. – Режим доступа: https://stackoverflow.com. – Дата доступа: 18.06.2023
2. Сайт с бесплатными звуками Zvukogram.com [Электронный ресурс]. – Режим доступа: https://zvukogram.com. – Дата доступа: 18.06.2023
3. Официальная википедия Embarcadero embarcadero.com [Электронный ресурс]. – Режим доступа: https://docwiki.embarcadero.com. – Дата доступа: 18.06.2023
4. Симонович С.В., Евсеев Г.А., Алексеев А.Г. Специальная информатика: Учебное пособие. - М.: АСТ-ПРЕСС: Инфорком-Пресс, 2001. - 480 с.

**Приложение А**

Листинг приложения

unit UTitle;

interface

uses

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

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

type

TTitle = class(TForm)

TitleImage: TImage;

TitleTimer: TTimer;

procedure TitleTimerTimer(Sender: TObject);

private

public

end;

var

Title: TTitle;

x: byte;

implementation

{$R \*.dfm}

uses UStart;

procedure TTitle.TitleTimerTimer(Sender: TObject);

//Процедура делает заставку менее прозрачной, когда она становится полностью непрозрачной начинается отсчёт, после которого форма становится более прозрачной

begin

if Title.AlphaBlendValue = 255 then x := x+1; //Отсчёт, если форма непрозрачная

if x = 0 then Title.AlphaBlendValue:= Title.AlphaBlendValue + 5 //Если отсчёта не было, делает форму менее прозрачной

else if x = 100 then Title.AlphaBlendValue:= Title.AlphaBlendValue - 5; //Если отсчёт закончился, делает форму более прозрачной

if Title.AlphaBlendValue = 0 then begin Application.CreateForm(TStart, Start); //Переход на следующую форму

Start.Show;

TitleTimer.enabled:=False;

Title.Hide; end;

end;

end.

unit UStart;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,

Vcl.Imaging.pngimage, System.ImageList, Vcl.ImgList;

type

TStart = class(TForm)

BStart: TButton;

BExit: TButton;

DiffRadioGroup: TRadioGroup;

StartTitleImage: TImage;

BLoad: TButton;

SaveComboBox: TComboBox;

BackgroundImage: TImage;

procedure BStartClick(Sender: TObject);

procedure BExitClick(Sender: TObject);

procedure DiffRadioGroupClick(Sender: TObject);

procedure SaveComboBoxSelect(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure BLoadClick(Sender: TObject);

private

public

end;

var

Start: TStart;

FName: string;

implementation

{$R \*.dfm}

uses UGame, UInv, UStats;

procedure TStart.BExitClick(Sender: TObject);

begin

Application.Terminate; //Закрывает проект

end;

procedure TStart.BLoadClick(Sender: TObject);

begin

FName := SaveComboBox.Items[SaveComboBox.Itemindex]; //Задание переменной, означающей название файла, из которого загружается карта

Application.CreateForm(TInventory, Inventory);

Application.CreateForm(TGame, Game);

Application.CreateForm(TStats, Stats);

Game.show;

Start.hide; //Переход на следующую форму

end;

procedure TStart.BStartClick(Sender: TObject);

begin

FName := 'OriginalMap.txt'; //Задание переменной, означающей название файла, из которого загружается карта

Application.CreateForm(TInventory, Inventory);

Application.CreateForm(TGame, Game);

Application.CreateForm(TStats, Stats);

Game.show;

Start.hide; //Переход на следующую форму

end;

procedure TStart.DiffRadioGroupClick(Sender: TObject);

begin

BStart.Enabled := True; //Активация кнопки

end;

procedure TStart.FormShow(Sender: TObject);

var k: integer;

var sr: TSearchRec;

begin

Canvas.Draw(0,0,BackgroundImage.Picture.Bitmap); //Рисует фон

SaveComboBox.Items.Clear; //Очищает списое сохранений

k := 0;

if FindFirst(ExtractFilePath(Application.ExeName)+'Saves/Save\_\*.txt', faAnyFile, sr)=0 then //ищем файлы сохранений в каталоге

repeat

SaveComboBox.Items.Add(sr.Name); //выводим список в ComboBox

until FindNext(sr)<>0;

FindClose(sr);

end;

procedure TStart.SaveComboBoxSelect(Sender: TObject);

begin

BLoad.Enabled := True; //Активирует кнопку загрузки

end;

end.

unit UGame;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus, Grids, System.ImageList,

Vcl.ImgList, Vcl.StdCtrls, mmsystem, ShellAPI;

type

//В этой форме происходит модификация класса StringGrid

TStringGrid = class(Grids.TStringGrid)

private

FImageList: TImageList;

protected

procedure KeyPress(var Key: Char); override;

procedure DrawCell(ACol, ARow: Integer; ARect: TRect; AState: TGridDrawState); override;

procedure Loaded; override;

end;

type

TGame = class(TForm)

GameMainMenu: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

SpriteImageList: TImageList;

SMap: TStringGrid;

N3: TMenuItem;

N4: TMenuItem;

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

procedure FormCreate(Sender: TObject);

procedure N1Click(Sender: TObject);

procedure N2Click(Sender: TObject);

procedure N3Click(Sender: TObject);

procedure N4Click(Sender: TObject);

private

public

end;

type

//Описание класса игрока

Player = class

public

HP: integer;

Dmg: integer;

PosX:integer;

PosY:integer;

constructor Create(HP:integer;Dmg:integer;PosX,PosY:integer);

procedure Move(x,y:integer;SMap:TStringGrid);

end;

type

//Описание класса противника

Enemy = class

public

HP: integer;

PosX:integer;

PosY:integer;

kind:string;

dmg:integer;

constructor Create(HP,dmg:integer;kind:string;PosX,PosY:integer);

end;

var

Game: TGame;

EMap: array of array of Enemy; //Карта врагов. Ячейки в ней либо пустые, либо содержат объект класса противника. Карта нужна для удобной записи и хранения всех врагов

p: Player;

epx: integer;

epy: integer;

DiffMod: byte; //Модификатор сложности, влияющий на здоровье и урон врагов, а так же на скорость мини-игр

Points: Cardinal; //Очки, необходимые для победы, которые даются за получение предметов и победу над врагами

implementation

{$R \*.dfm}

uses UStart,UInv, UOrkB, UShroomB, USnakeB, UStats;

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

//При закрытии формы открывает стартовое меню

begin

Start.show;

end;

constructor Player.Create(HP:integer;Dmg:integer;PosX,PosY:integer);

//Конструктор класса игрока

begin

Self.HP := HP;

Self.Dmg := Dmg;

Self.PosX := Posx;

Self.PosY := PosY;

end;

constructor Enemy.Create(HP,dmg:integer;kind:string;PosX,PosY:integer);

//Конструктор класса противника

begin

Self.HP := Hp;

Self.PosX := Posx;

Self.PosY := PosY;

Self.kind := kind;

Self.dmg := dmg;

end;

procedure Player.Move(x,y:integer;SMap:TStringGrid);

//Процедура движения игрока

const r = 7;

var i,k:byte;

begin

if ((self.PosX + x) < 0) or ((self.PosY + y) < 0) or ((self.PosX + x) > SMap.ColCount - 1) or ((self.PosY + y) > SMap.RowCount - 1) then Exit; //Проверяет, не пытается ли игрок выйти за пределы таблицы

//Если игрок двигается на клетку пути и там нет противника, то перемещает его

if (EMap[self.PosX + x,self.PosY + y] = Nil) and (SMap.Cells[self.PosX + x,self.PosY + y] = '1') then begin

self.PosX:= self.PosX + x;

self.PosY:= self.PosY + y;

//Перемещает положение таблицы так, что бы всегда был виден игрок, направление в каторое он движет и при это не выходило за границы таблицы

if ((self.PosX + x\*r) >= 0) and ((self.PosY + y\*r) >= 0) and ((self.PosX + x\*r) <= SMap.ColCount - 1) and ((self.PosY + y\*r) <= SMap.RowCount - 1) then begin

SMap.Row := self.Posy + y \* r;

SMap.Col := self.Posx + x \* r;

end; end

//Если игрок двигается на противника, то начинает соответствующий бой, запоминая при этом положение противника

else if EMap[self.PosX + x,self.PosY + y] <> Nil then begin

if EMap[self.PosX + x,self.PosY + y].kind = 'ork' then begin epx:= self.PosX + x;

epy:= self.PosY + y;

Application.CreateForm(TOrkBattle, OrkBattle);

OrkBattle.showmodal end else

if EMap[self.PosX + x,self.PosY + y].kind = 'snake' then begin epx:= self.PosX + x;

epy:= self.PosY + y;

Application.CreateForm(TSnakeBattle, SnakeBattle);

SnakeBattle.showmodal end else

if EMap[self.PosX + x,self.PosY + y].kind = 'shroom' then begin epx:= self.PosX + x;

epy:= self.PosY + y;

Application.CreateForm(TShroomBattle, ShroomBattle);

ShroomBattle.showmodal end;

end

//Если игрок двигается на сундук, то убриет сундук, дает игроку случайный предмет и выводит сообщение

else if SMap.Cells[self.PosX + x,self.PosY + y] = '2' then begin

SMap.Cells[self.PosX + x,self.PosY + y] := '1';

i := random(3);

k := 0;

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/GainItem.wav'),1,SND\_ASYNC); //Играет звук

//Определение, какой предмет получил игрок, добавление его в инвентарь и прибавление к характеристикам

if i = 0 then begin

for var j:integer := 0 to Inventory.ItemListBox.Items.Count - 1 do

//Если игрок подбирает уже второй меч, то не улучшает характеристики игрока

if Inventory.ItemListBox.Items[j] = 'Меч' then k:=1;

if k = 0 then p.Dmg := p.Dmg + 5;

Inventory.ItemListBox.Items.Add('Меч');

ShowMessage('Вы нашли меч!'+#13+#10+'Он стоит 150 очков');

Points := Points + 150;

end

else if i = 1 then begin

for var j:integer := 0 to Inventory.ItemListBox.Items.Count - 1 do

//Если игрок подбирает уже второй амулет, то не улучшает характеристики игрока

if Inventory.ItemListBox.Items[j] = 'Амулет' then k:=1;

if k = 0 then p.hp := p.hp + 50;

Inventory.ItemListBox.Items.Add('Амулет');

Points := Points + 100;

ShowMessage('Вы нашли амулет!'+#13+#10+'Он стоит 100 очков')

end

//При подборе зелья характеристики улучшаются каждый раз

else if i = 2 then begin

p.hp := p.hp + 20;

Inventory.ItemListBox.Items.Add('Зелье');

Points := Points + 50;

ShowMessage('Вы нашли зелье!'+#13+#10+'Он стоит 50 очков');

end;

end

//Если игрок двигается на выход, то выводит сообщение, запрещающее ему выйти если очков не хватает, или заканчивает игру победой если очков хватает

else if SMap.Cells[self.PosX + x,self.PosY + y] = '3' then begin

if Points < 1000 then ShowMessage('Вы не можете уйти с пустыми руками!'+#13+#10+'Для победы наберите хотя бы 1000 очков')

else begin

//Играет звук и выводит сообщение победы, показывая количество очков

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/Victory.wav'),1,SND\_ASYNC);

ShowMessage('Победа!'+#13+#10+'Вы набрали '+IntToStr(Points)+' очков');

Game.Close;

end;

end;

end;

procedure TGame.FormCreate(Sender: TObject);

var

f : textfile;

a,ik:char;

c,ec,px,py,eh,ex,ey,ed,ph,pd:Cardinal;

ek: string;

//Процедура читает файл оригинальной карты или файл сохранения и берёт карту, характеристики игрока и положения врагов из него

begin

assignfile(f,ExtractFilePath(Application.Exename)+'Saves/'+FName);

reset(f);

//Если файлом выбирается оригинальная карта, то сложность считывается из стартового меню, иначе она читается из файла

if FName = 'OriginalMap.txt' then

case Start.DiffRadioGroup.ItemIndex of

0: DiffMod:=1;

1: DiffMod:=2;

2: DiffMod:=3;

end

else readln(f,DiffMod);

//Чтение размера карты

read(f,c);

SMap.ColCount := c;

readln(f,c);

SMap.RowCount := c;

//Чтение ячеек карты

for var i:integer := 0 to SMap.RowCount -1 do begin

for var j:integer := 0 to SMap.ColCount -1 do begin

read(f, a);

SMap.Cells[j,i]:=a;

end;

readln(f)

end;

//Чтение характеристик игрока

read(f,px);

read(f,py);

read(f,ph);

read(f,pd);

readln(f,Points);

p := Player.Create(ph,pd,px,py);

//Чтение количества врагов и самих врагов, заполняя при этом карту врагов

readln(f, ec);

SetLength(EMap,SMap.ColCount,SMap.RowCount);

if ec > 0 then

for var k: integer := 0 to ec - 1 do begin

read(f,ex);

readln(f,ey);

readln(f,ek);

ed:=0;

eh:=0;

//Здоровье и урон врагов задается в зависимости от их типа и сложности

if ek = 'shroom' then begin ed := 5 \* DiffMod;

eh := 50\* DiffMod; end

else if ek = 'ork' then begin ed := 5\* DiffMod;

eh := 50\* DiffMod; end

else if ek = 'snake' then begin ed := 9\* DiffMod;

eh := 30\* DiffMod; end;

EMap[ex,ey] := Enemy.Create(eh,ed,ek,ex,ey);

end;

//Читает содержимое инвентаря, если оно присутствует

while not Eof(f) do begin

read(f,ik);

if ik = '0' then Inventory.ItemListBox.Items.Add('Амулет')

else if ik = '1' then Inventory.ItemListBox.Items.Add('Меч')

else if ik = '2' then Inventory.ItemListBox.Items.Add('Зелье');

end;

closefile(f);

end;

procedure TGame.N1Click(Sender: TObject);

//Открывает инвентарь

begin

Inventory.ShowModal;

end;

procedure TGame.N2Click(Sender: TObject);

//Открывает характеристики

begin

Stats.ShowModal;

end;

procedure TGame.N3Click(Sender: TObject);

var f: TextFile;

fk, ec:integer;

//Сохраняет игру, записывая её в файл Save\_\*.txt

begin

fk:=0;

//Определяет имя сохранения, поочерёдно перебирая номер и проверяя, не занят ли он

if not DirectoryExists(ExtractFilePath(Application.Exename)+'Saves') then CreateDir(ExtractFilePath(Application.Exename)+'Saves');

Repeat

inc(fk)

Until not FileExists(ExtractFilePath(Application.Exename)+'Saves/'+'Save\_'+inttostr(fk)+'.txt');

AssignFile(F,ExtractFilePath(Application.Exename)+'Saves/'+'Save\_'+inttostr(fk)+'.txt');

//Далее до конца процедуры записываются те же значения, что читает процедура FormCreate

Rewrite(f);

writeln(f,DiffMod);

write(f,SMap.ColCount);

write(f,' ');

writeln(F,SMap.RowCount);

ec := 0;

for var i:integer := 0 to SMap.RowCount -1 do begin

for var j:integer := 0 to SMap.ColCount -1 do begin

write(f,SMap.Cells[j,i]);

if EMap[j,i] <> Nil then inc(ec);

end;

writeln(f)

end;

write(f,p.PosX);

write(f,' ');

write(f,p.PosY);

write(f,' ');

write(f,p.HP);

write(f,' ');

write(f,p.Dmg);

write(f,' ');

writeln(f,Points);

writeln(f,ec);

for var i:integer := 0 to SMap.RowCount -1 do begin

for var j:integer := 0 to SMap.ColCount -1 do begin

if EMap[j,i] <> Nil then begin

write(f,j);

write(f,' ');

writeln(f,i);

writeln(f,EMap[j,i].kind);

end;

end;

end;

for var i:integer := 0 to Inventory.ItemListBox.Items.Count - 1 do

if Inventory.ItemListBox.Items[i] = 'Амулет' then write(f,'0 ')

else if Inventory.ItemListBox.Items[i] = 'Меч' then write(f,'1 ')

else if Inventory.ItemListBox.Items[i] = 'Зелье' then write(f,'2 ');

ShowMessage('Ваша игра была успешно сохранена как Save\_'+inttostr(fk)+'.txt');

CloseFile(f);

end;

procedure TGame.N4Click(Sender: TObject);

//Открывает файл справки

begin

ShellExecute(0,PChar('Open'),PChar(ExtractFilePath(Application.Exename)+'ProjectHelp.chm'),Nil,Nil,SW\_SHOW);

end;

procedure TStringGrid.Loaded;

var

cmp: TComponent;

//Метод класса StringGrid, привязывающий список картинок к таблице

begin

inherited;

cmp := Owner.FindComponent('SpriteImageList');

if Assigned(cmp) and (cmp is TImageList) then

FImageList := TImageList(cmp);

end;

procedure TStringGrid.DrawCell(ACol, ARow: Integer; ARect: TRect; AState: TGridDrawState);

var

s: string;

bmp: TBitmap;

xOff: Integer;

YOff: Integer;

//Метод класса StringGrid, который читает содержимое ячейки таблицы карты, таблицы противника и положение игрока и рисует соответствующую картинку на таблице

begin

inherited;

if not Assigned(FImageList) then //Проверяет, привязан ли список картинок к таблице

Exit;

s := Cells[ACol, ARow]; //Читает ячейки таблицы

Canvas.FillRect(ARect);

bmp := TBitmap.Create; //Создает переменную картинки

try

//Проверяет значение ячейки карты, значение ячейки таблицы противников и положение игрока и выбирает соответствующую картинку

if s='0' then FImageList.GetBitmap(0, bmp)

else if s='1' then FImageList.GetBitmap(1, bmp)

else if s='2' then FImageList.GetBitmap(6, bmp)

else if s='3' then FImageList.GetBitmap(7, bmp);

if (p.PosX = ACol) and (p.PosY = ARow) then

FImageList.GetBitmap(2, bmp)

else if EMap[ACol,ARow] <> Nil then

if EMap[ACol,ARow].kind = 'snake' then FImageList.GetBitmap(3, bmp)

else if EMap[ACol,ARow].kind = 'ork' then FImageList.GetBitmap(4, bmp)

else if EMap[ACol,ARow].kind = 'shroom' then FImageList.GetBitmap(5, bmp);

//Определяет положение нужное картнки и рисует её

xOff := ARect.Left + ((ARect.Right - ARect.Left) - bmp.Width) div 2;

YOff := ARect.Top + ((ARect.Bottom - ARect.Top) - bmp.Height) div 2;

Canvas.Draw(xOff, YOff, bmp);

finally

FreeAndNil(bmp); //Освобождает переменную картинки, даже если при выполнении try что-то пошло не так

end;

end;

procedure TStringGrid.KeyPress(var Key: Char);

//Метод класса StringGrid, который вызывает процедуру Move и передает ей

begin

//Ячейкам, на которых находится игрок до и после перемещения заново задаются значения клетки пути для дого, что бы вызвать процедуру DrawCell

Self.Cells[p.PosX,p.PosY] := '1';

if (Key = 'a') or (Key = 'A') or (Key = 'Ф') or (Key = 'ф') then p.Move(-1,0,Self);

if (Key = 'w') or (Key = 'W') or (Key = 'Ц') or (Key = 'ц') then p.Move(0,-1,Self);

if (Key = 'd') or (Key = 'D') or (Key = 'В') or (Key = 'в') then p.Move(1,0,Self);

if (Key = 's') or (Key = 'S') or (Key = 'Ы') or (Key = 'ы') then p.Move(0,1,Self);

Self.Cells[p.PosX,p.PosY] := '1';

end;

end.

unit UInv;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, System.ImageList,

Vcl.ImgList, Vcl.ExtCtrls, Vcl.Imaging.pngimage;

type

TInventory = class(TForm)

ItemListBox: TListBox;

ItemImage: TImage;

ItemsImageList: TImageList;

ItemMemo: TMemo;

procedure ItemListBoxClick(Sender: TObject);

private

public

end;

var

Inventory: TInventory;

implementation

{$R \*.dfm}

procedure TInventory.ItemListBoxClick(Sender: TObject);

//Процедура читает пункт меню, который выбрал игрок и выводит соответствующий текст и картинку

begin

if ItemListBox.ItemIndex > -1 then begin

if ItemListBox.Items[ItemListBox.ItemIndex] = 'Меч' then begin

ItemsImageList.GetIcon(0,ItemImage.Picture.Icon);

ItemMemo.Text := 'Позолоченный меч'+#13+#10+'Прибавляет 5 к урону игрока'+#13+#10+'Используется не больше одного'+#13+#10+'Стоит 150 очков';

end

else if ItemListBox.Items[ItemListBox.ItemIndex] = 'Амулет' then begin

ItemsImageList.GetIcon(1,ItemImage.Picture.Icon);

ItemMemo.Text := 'Магический амулет'+#13+#10+'Прибавляет 50 к здоровью игрока'+#13+#10+'Используется не больше одного'+#13+#10+'Стоит 100 очков';

end

else if ItemListBox.Items[ItemListBox.ItemIndex] = 'Зелье' then begin

ItemsImageList.GetIcon(2,ItemImage.Picture.Icon);

ItemMemo.Text := 'Лечебное зелье'+#13+#10+'Прибавляет 20 к здоровью игрока'+#13+#10+'Эффекты нескольких зелий складываются'+#13+#10+'Стоит 50 очков';

end;

end;

end;

end.

unit UStats;

interface

uses

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

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

Vcl.ExtCtrls, Vcl.Grids, System.ImageList, Vcl.ImgList;

type

TStats = class(TForm)

StatGrid: TStringGrid;

PImage: TImage;

EButton: TButton;

SwordImage: TImage;

AmuletImage: TImage;

procedure FormShow(Sender: TObject);

procedure EButtonClick(Sender: TObject);

private

public

end;

var

Stats: TStats;

implementation

{$R \*.dfm}

uses UGame, UInv;

procedure TStats.EButtonClick(Sender: TObject);

begin

Stats.Close; //Закрытие формы

end;

procedure TStats.FormShow(Sender: TObject);

begin

SwordImage.Visible := False;

AmuletImage.Visible := False;

for var j:integer := 0 to Inventory.ItemListBox.Items.Count - 1 do begin

//Если в инвентаре присутствует меч или амулет, то делает соответствующие картинки видимыми

if Inventory.ItemListBox.Items[j] = 'Меч' then SwordImage.Visible := True

else if Inventory.ItemListBox.Items[j] = 'Амулет' then AmuletImage.Visible := True;

end;

//Заполняет таблицу характеристик

StatGrid.Cells[1,0]:= inttostr(p.hp);

StatGrid.Cells[0,0]:= 'Здоровье';

StatGrid.Cells[1,1]:= inttostr(p.dmg);

StatGrid.Cells[0,1]:= 'Урон';

StatGrid.Cells[1,2]:= inttostr(Points);

StatGrid.Cells[0,2]:= 'Очки';

end;

end.

unit UOrkB;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls, Vcl.Grids,

Vcl.Imaging.pngimage, Vcl.ExtCtrls, Vcl.StdCtrls, mmsystem;

type

TOrkBattle = class(TForm)

PImage: TImage;

StatGrid: TStringGrid;

EImage: TImage;

DrumsImage: TImage;

DrumsBar: TProgressBar;

DrumsTimer: TTimer;

TimeLabel: TLabel;

RestTimer: TTimer;

CountLabel: TLabel;

procedure FormCreate(Sender: TObject);

procedure DrumsImageClick(Sender: TObject);

procedure DrumsTimerTimer(Sender: TObject);

procedure RestTimerTimer(Sender: TObject);

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

private

public

end;

var

OrkBattle: TOrkBattle;

x,y,t:Integer;

implementation

{$R \*.dfm}

uses UGame;

procedure TOrkBattle.DrumsImageClick(Sender: TObject);

//Процедура запускает таймер, ведёт счёт нажатий и играет соответствующий звук

begin

if DrumsTimer.Enabled = False then begin

DrumsTimer.Enabled := True;

DrumsBar.State:= pbsNormal;

end;

inc(x);

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/Drums'+inttostr(random(4)+1)+'.wav'),1,SND\_ASYNC);

DrumsBar.Position := x;

CountLabel.Caption := inttostr(x);

//Цвет надписи меняется от количества нажатий

if x <= 5 then CountLabel.Font.Color := clRed

else if x <= 10 then CountLabel.Font.Color := clYellow

else if x > 10 then CountLabel.Font.Color := clGreen;

if x = 15 then DrumsImage.Enabled := False;

end;

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

begin

//При закрытии формы выключает таймеры

DrumsTimer.Enabled := False;

RestTimer.Enabled := False;

end;

procedure TOrkBattle.FormCreate(Sender: TObject);

begin

//Таймеры поумолчанию выключены

DrumsBar.State:= pbsError;

DrumsTimer.Enabled := False;

RestTimer.Enabled := False;

//Устанавливает время для нажатий

t := 9 div DiffMod + 1;

TimeLabel.Caption := inttostr(t);

//Заполняет таблицу характеристик

StatGrid.Cells[0,0]:= 'Игрок';

StatGrid.Cells[1,0]:= '';

StatGrid.Cells[2,0]:= 'Орк';

StatGrid.Cells[0,1]:= inttostr(p.hp);

StatGrid.Cells[1,1]:= 'Здоровье';

StatGrid.Cells[2,1]:= inttostr(EMap[epx,epy].hp);

StatGrid.Cells[0,2]:= inttostr(p.dmg);

StatGrid.Cells[1,2]:= 'Урон';

StatGrid.Cells[2,2]:= inttostr(EMap[epx,epy].dmg);

end;

procedure TOrkBattle.RestTimerTimer(Sender: TObject);

//Процедура считает время, пока игра находится в режиме отдыха, затем опять активирует нужные компоненты и обнуляет значения

begin

inc(y);

TimeLabel.Caption := TimeLabel.Caption + '.';

if y = 4 then begin

DrumsImage.Enabled := True;

t := 9 div DiffMod + 1;

TimeLabel.Caption := inttostr(t);

DrumsBar.Position := 0;

x := 0;

y := 0;

RestTimer.Enabled := False;

CountLabel.Caption := '0';

DrumsBar.Position := 0;

end;

end;

procedure TOrkBattle.DrumsTimerTimer(Sender: TObject);

//Процедура уменьшает счётчик времени. Когда время кончается, наносит игроку и противнику урон, зависящий от количества нажатий и переходит в режим отдыха

begin

dec(t);

TimeLabel.Caption := inttostr(t);

if TimeLabel.Caption = '0' then begin

EMap[epx,epy].hp := EMap[epx,epy].hp - Round(10\*p.dmg\*x/15);

p.hp := p.hp - Round((EMap[epx,epy].dmg/(x/15))-EMap[epx,epy].dmg);

StatGrid.Cells[2,1]:= inttostr(EMap[epx,epy].hp);

StatGrid.Cells[0,1]:= inttostr(p.hp);

//Обнуляет счётчики

x := 0;

y := 0;

//Переход в режим отдыха

DrumsTimer.Enabled := False;

DrumsBar.State:= pbsError;

RestTimer.Enabled := True;

DrumsImage.Enabled := False;

TimeLabel.Caption := 'Отдых';

//Оканчивает игру поражением

if p.HP <= 0 then begin DrumsTimer.Enabled := False;

RestTimer.Enabled := False;

OrkBattle.Close;

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/Death'+inttostr(random(2)+1)+'.wav'),1,SND\_ASYNC);

ShowMessage('Вы погибли!');

Game.Close;

end

//Оканчивает сражение победой

else if EMap[epx,epy].hp <= 0 then begin EMap[epx,epy]:=Nil;

Game.SMap.Cells[epx,epy]:= '1';

DrumsTimer.Enabled := False;

RestTimer.Enabled := False;

Points := Points + 150;

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/BattleVictory.wav'),1,SND\_ASYNC);

ShowMessage('Вы выиграли!'+#13+#10+'Вы получили 150 очков');

OrkBattle.Close;

end;

end;

end;

end.

unit UShroomB;

interface

uses

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

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, System.ImageList,

Vcl.ImgList, Grids, Vcl.Imaging.pngimage, Vcl.StdCtrls, mmsystem;

type

//В этой форме происходит модификация класса StringGrid

TStringGrid = class(Grids.TStringGrid)

private

FImageList: TImageList;

protected

procedure DrawCell(ACol, ARow: Integer; ARect: TRect; AState: TGridDrawState); override;

procedure Loaded; override;

procedure MouseDown(Button: TMouseButton; Shift: TShiftState;X, Y: Integer); override;

end;

type

TShroomBattle = class(TForm)

PlayGrid: TStringGrid;

MushroomImageList: TImageList;

EImage: TImage;

PImage: TImage;

StatGrid: TStringGrid;

TurnTimer: TTimer;

procedure FormCreate(Sender: TObject);

procedure TurnTimerTimer(Sender: TObject);

procedure PlayGridSelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

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

private

public

end;

var

ShroomBattle: TShroomBattle;

x1,y1,x2,y2:integer;

implementation

{$R \*.dfm}

uses UGame;

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

//При закрытии формы выклчает таймер

begin

TurnTimer.enabled:= False;

end;

procedure TShroomBattle.FormCreate(Sender: TObject);

begin

TurnTimer.Interval := TurnTimer.Interval div DiffMod;

for var i:integer := 0 to PlayGrid.RowCount - 1 do begin

for var j:integer := 0 to PlayGrid.ColCount - 1 do begin

PlayGrid.Cells[j,i]:='0';

end;

end;

TurnTimer.enabled:= True; //Включает таймер

//Заполняет таблицу характеристик

StatGrid.Cells[0,0]:= 'Игрок';

StatGrid.Cells[1,0]:= '';

StatGrid.Cells[2,0]:= 'Грибник';

StatGrid.Cells[0,1]:= inttostr(p.hp);

StatGrid.Cells[1,1]:= 'Здоровье';

StatGrid.Cells[2,1]:= inttostr(EMap[epx,epy].hp);

StatGrid.Cells[0,2]:= inttostr(p.dmg);

StatGrid.Cells[1,2]:= 'Урон';

StatGrid.Cells[2,2]:= inttostr(EMap[epx,epy].dmg);

end;

procedure TShroomBattle.PlayGridSelectCell(Sender: TObject; ACol, ARow: Integer;var CanSelect: Boolean);

//Если в случаае какой-либо ошибки вылезет лишняя ячейка таблицы, то запрещает нажать на нее

begin

if (ACol < 0) or (ARow < 0) then CanSelect:=false;

end;

procedure TShroomBattle.TurnTimerTimer(Sender: TObject);

var x,y:integer;

//Процедура ставит гриб на случайной позиции, взрывает прошлый неубранный гриб (при этом нанося урон), и убирает взрыв

begin

repeat

x:= random(7);

y:= random(7);

until (PlayGrid.Cells[x,y] = '0');

PlayGrid.Cells[x,y] := inttostr(random(3)+1); //Ставит случайный гриб на случайную позицию

if (PlayGrid.Cells[x1,y1] <> '0') and (PlayGrid.Cells[x1,y1] <> '4') then begin //Взрывает неубранный гриб, нанося урон игроку и играя звук

PlayGrid.Cells[x1,y1] := '4';

p.HP := p.HP - EMap[epx,epy].dmg;

StatGrid.Cells[0,1]:= inttostr(p.hp);

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/MushroomExplode'+inttostr(random(3)+1)+'.wav'),1,SND\_ASYNC);

//Заканчивает игру поражением

if p.HP <= 0 then begin TurnTimer.enabled:= False;

ShroomBattle.Close;

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/Death'+inttostr(random(2)+1)+'.wav'),1,SND\_ASYNC);

ShowMessage('Вы погибли!');

Game.Close;

end

end;

if PlayGrid.Cells[x2,y2] = '4' then PlayGrid.Cells[x2,y2]:= '0';

x2 := x1; //Задает координаты позапрошлого гриба

y2 := y1;

x1 := x; //Задает координаты прошлого гриба

y1 := y;

end;

procedure TStringGrid.Loaded;

var

cmp: TComponent;

//Метод класса StringGrid, привязывающий к нему список картинок грибов

begin

inherited;

cmp := Owner.FindComponent('MushroomImageList');

if Assigned(cmp) and (cmp is TImageList) then

FImageList := TImageList(cmp);

end;

procedure TStringGrid.DrawCell(ACol, ARow: Integer; ARect: TRect; AState: TGridDrawState);

var

s: string;

bmp: TBitmap;

xOff: Integer;

YOff: Integer;

//Метод класса StringGrid, рисующий картинку на ячейке таблицы, соответсвующую зодержанию этой ячейки

begin

inherited;

//Заканчивает процедуру, если в случае какой-либо ошибки список картинок окажется не привязан и в стучае, если он применяется не к той таблице

if not Assigned(FImageList) then

Exit;

if self <> ShroomBattle.PlayGrid then

Exit;

s := Cells[ACol, ARow]; //читает содержимое ячейки

Canvas.FillRect(ARect);

bmp := TBitmap.Create; //Создает переменную картинки

try

//Определяет нужную картинку

if (s = '1') then

FImageList.GetBitmap(1, bmp)

else if (s = '2') then

FImageList.GetBitmap(2, bmp)

else if (s = '3') then

FImageList.GetBitmap(3, bmp)

else if (s = '0') then

FImageList.GetBitmap(0, bmp)

else if (s = '4') then

FImageList.GetBitmap(4, bmp);

//Определяет положение нужное картнки и рисует её

xOff := ARect.Left + ((ARect.Right - ARect.Left) - bmp.Width) div 2;

YOff := ARect.Top + ((ARect.Bottom - ARect.Top) - bmp.Height) div 2;

Canvas.Draw(xOff, YOff, bmp);

finally

FreeAndNil(bmp); //Освобождает переменную картинки, даже если при выполнении try что-то пошло не так

end;

end;

procedure TStringGrid.MouseDown(Button: TMouseButton; Shift: TShiftState;

X, Y: Integer);

var

C: Integer;

r: Integer;

//Метод класса StringGrid, который обрабатывает щелчки мышью по таблице и, если нажатие происходит по грибу, то убирает его и наносит противнику урон

begin

inherited;

MouseToCell(X, Y, C, r); //Определяет ячейку, на которую нажал игрок

if (c<0) or (r<0) then //Проверяет, не было ли в случае какой-либо ошибки выхода за границу таблицы и выходит из процедуры если было

Exit;

if (Button = mbleft) and (Self.Cells[c,r] <> '0') and (Self.Cells[c,r] <> '4') then begin //Проверяет, было ли нажатие по существующему грибу

//Убирает гриб, наносит урон противнику и играет звук

Self.Cells[c,r]:='0';

EMap[epx,epy].hp := EMap[epx,epy].hp - p.Dmg;

ShroomBattle.StatGrid.Cells[2,1]:= inttostr(EMap[epx,epy].hp);

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/MushroomClear'+inttostr(random(4)+1)+'.wav'),1,SND\_ASYNC);

//Заканчивает бой победой

if EMap[epx,epy].hp <= 0 then begin EMap[epx,epy]:=Nil;

Game.SMap.Cells[epx,epy]:= '1';

ShroomBattle.TurnTimer.enabled:= False;

Points := Points + 150;

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/BattleVictory.wav'),1,SND\_ASYNC);

ShowMessage('Вы выиграли!'+#13+#10+'Вы получили 150 очков');

ShroomBattle.Close;

end;

end;

end;

end.

unit USnakeB;

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.pngimage, Grids, Vcl.StdCtrls, Vcl.Samples.Gauges,

System.ImageList, Vcl.ImgList, mmsystem;

type

TSnakeBattle = class(TForm)

StatGrid: TStringGrid;

EImage: TImage;

PImage: TImage;

AttackButton: TButton;

SnakeTimer: TTimer;

SnakeImage: TImage;

SnakeImageList: TImageList;

procedure FormCreate(Sender: TObject);

procedure SnakeTimerTimer(Sender: TObject);

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

procedure AttackButtonClick(Sender: TObject);

private

public

end;

var

SnakeBattle: TSnakeBattle;

x,i:ShortInt;

implementation

{$R \*.dfm}

uses UGame;

procedure TSnakeBattle.SnakeTimerTimer(Sender: TObject);

//Меняет значение положения стрелки и устанавливает соответствующую картинку

begin

if x = 4 then i := -1;

if x = 0 then i := 1;

x := x + i;

SnakeImageList.GetIcon(x,SnakeImage.Picture.Icon)

end;

procedure TSnakeBattle.AttackButtonClick(Sender: TObject);

//При нажатии на кнопку удара наносит игроку и противнику урон, зависящий от положения стрелки и играет нужный звук

begin

if x = 2 then begin

EMap[epx,epy].hp := EMap[epx,epy].hp - p.Dmg;

SnakeBattle.StatGrid.Cells[2,1]:= inttostr(EMap[epx,epy].hp);

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/Hit'+inttostr(random(3)+1)+'.wav'),1,SND\_ASYNC);

end

else if (x = 1) or (x = 3) then begin

p.HP := p.HP - EMap[epx,epy].dmg div 3;

StatGrid.Cells[0,1]:= inttostr(p.hp);

EMap[epx,epy].hp := EMap[epx,epy].hp - p.Dmg div 2;

SnakeBattle.StatGrid.Cells[2,1]:= inttostr(EMap[epx,epy].hp);

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/Snake'+inttostr(random(2)+1)+'.wav'),1,SND\_ASYNC);

end

else if (x = 0) or (x = 4) then begin

p.HP := p.HP - EMap[epx,epy].dmg;

StatGrid.Cells[0,1]:= inttostr(p.hp);

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/Snake'+inttostr(random(2)+1)+'.wav'),1,SND\_ASYNC);

end;

//Заканчивает игру положением

if p.HP <= 0 then begin SnakeTimer.Enabled := False;

SnakeBattle.Close;

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/Death'+inttostr(random(2)+1)+'.wav'),1,SND\_ASYNC);

ShowMessage('Вы погибли!');

Game.Close;

end

//Заканчивает сражение победой

else if EMap[epx,epy].hp <= 0 then begin EMap[epx,epy]:=Nil;

Game.SMap.Cells[epx,epy]:= '1';

SnakeTimer.Enabled := False;

Points := Points + 200;

PlaySound(pchar(ExtractFilePath(Application.Exename)+'Sounds/BattleVictory.wav'),1,SND\_ASYNC);

ShowMessage('Вы выиграли!'+#13+#10+'Вы получили 200 очков');

SnakeBattle.Close;

end;

end;

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

//При закрытии формы выключает таймер

begin

SnakeTimer.Enabled := False;

end;

procedure TSnakeBattle.FormCreate(Sender: TObject);

begin

//Включает таймер и ставит интервал

SnakeTimer.Enabled := True;

SnakeTimer.Interval:= SnakeTimer.Interval div DiffMod;

//Заполняет таблицу характеристик

StatGrid.Cells[0,0]:= 'Игрок';

StatGrid.Cells[1,0]:= '';

StatGrid.Cells[2,0]:= 'Змея';

StatGrid.Cells[0,1]:= inttostr(p.hp);

StatGrid.Cells[1,1]:= 'Здоровье';

StatGrid.Cells[2,1]:= inttostr(EMap[epx,epy].hp);

StatGrid.Cells[0,2]:= inttostr(p.dmg);

StatGrid.Cells[1,2]:= 'Урон';

StatGrid.Cells[2,2]:= inttostr(EMap[epx,epy].dmg);

end;

end.