Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования(ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

ИГРОВОЕ ПРОГРАММНОЕ СРЕДСТВО

БГУИР КР 1-40 01 01 106 ПЗ

Студент: гр. 351001

Зиновенко К.А.

Руководитель:

асс. Фадеева Е.Е.

Минск 2024

**СОДЕРЖАНИЕ**

Содержание…..........................................................................................................4

Введение…...............................................................................................................5

1. Анализ требований, литературных источников и формирование требова-ний к проектируемому программному средству…..............................................6
   1. Обзор известных решений…….…..................................................................6
   2. Назначение разработки………........................................................................9
   3. Состав выполняемых функций………….......................................................9
   4. Входные данные...............................................................................................9
   5. Выходные данные……....................................................................................9
2. Анализ требований к программному средству и разработка функциональ-ных требований……………………………………………..................................10
   1. Описание функциональности системы………………................................10
   2. Спецификация функциональных требований……………….....................11
3. Проектирование программного средства……………………………………14
   1. Основной алгоритм программы……………………………………………14
   2. Главное меню………………………………………………………………..14
   3. Легенда управления………………………………………………………...14
   4. Таблица лидеров…………………………………………………………….14
   5. Авторизация…………………………………………………………………14
   6. Начало игры…………………………………………………………………15
   7. Начало уровня……………………………………………………………….15
   8. Конец уровня………………………………………………………………..15
   9. Обновление информационной панели…………………………………….15
   10. Перемещение танков………………………………………………………15
   11. Перемещение снарядов……………………………………………………15
   12. Разрушение объектов……………………………………………………...15
   13. Возрождение танков……………………………………………………….15
   14. Выход из программы………………………………………………………15
4. Создание программного средства……………………………………………16
   1. Проектирование модулей программного средства……………………….16
   2. Разработка модулей програмнного средства……………………………...16
5. Тестирование, проверка работоспособности и анализ полученных резуль-татов………………………………………………………………………………20
   1. Тестирование ПС……………………………………………………………20
   2. Анализ полученных результатов…………………………………………..21
6. Руководство по установке и эксплуатации………………………………….22

Заключение……………………………………………………………………….26

Список используемой литературы……………………………………………...27

Приложение А……………………………………………………………………28

Приложение Б……………………………………………………………………36

**ВВЕДЕНИЕ**

Индустрия разработки игр стала набирать популярность в начале 00-х и до сих пор укрепляет свои позиции. С развитием технологий качество игр становилось все лучше и лучше, росли ожидания игроков, требования к компьютерному железу пользователей. Все больше и больше денег входило в оборот данной сферы. Существуют очень разнообразные жанры игр: шутеры, симуляторы, аркады, образовательные игры, хорроры, квесты и т.д. Но игры всех этих жанров объединяет одна важная черта, а именно: эмоции, опыт, захватывающие сюжеты и многое другое. Для кого-то игры - это лишь увлечение, завязанное на интересе к какой-либо теме. Для других игры - способ заработать большие деньги, ведь, как мы знаем, крупные проекты пользуются успехом среди игроков даже через десятки лет после выпуска. Для справки: небезызвестный проект GTA 5 имел бюджет 137млн.$, при этом игра собрала сумму с продаж копий свыше 1млрд.$ через 3 дня после мирового релиза.

Цель моей курсовой работы - разработать игру, популярную на консолях Денди в середине 90-х, ставшую приятным воспоминанием для взрослых старше 25-30 лет.

Для реализации данного проекта я проанализирую возможные варианты реализации игровых процессов и предложу свою версию игры, написанную на языке программирования Delphi 11.3. Я опишу необходимые алгоритмы, составлю их схемы, проанализирую процесс взаимодействия человека и игры и разработаю соответствующее программное средство.

1. **АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ**

**1.1. Обзор известных решений**

Существует множество модификаций игры, выбранной мной в качестве образца, однако рассмотрена будет версия игры 1985-го года, которая была разработана под консоли серии Денди. В настоящее время исходный код изучаемой игры представлен лишь в виде двоичного кода, поэтому, в связи с затруднениями перевода двоичного кода на человекоподобный язык, выводы будут составлены исходя из наблюдений во время изучения игры и анализа полученных результатов.

Суть игры заключается в том, чтобы защищать свою базу от волн вражеских танков. Цикл игры продолжается в течение 70-ти волн, во время которых игрок должен не допустить уничтожение своей базы, а так же не потерять имеющиеся жизни. Игра содержит 35 различных карт. С каждой новой волной танки противника становятся прочнее и быстрее. После прохождения 35-ти волн, 36 начинается с 1 уровня, однако противники становятся намного сильнее, чем на первых 35-ти волнах. В конечном итоге после прохождения 70-ти волн игрок видит концовку, хотя назвать ее таковой нельзя. В оригинале на экран выводятся имена разработчиков и сообщение с благодарностью за пройденные уровни. Можно сделать вывод, что игра принадлежит к жанру аркады, в которой результат измеряется заработанными очками за то или иное действие.

Карта представляет собой матрицу размером 13х13 объектов. Объекты - это также матрицы, но уже размером 2х2 и состоящие из более мелких частей. Объекты игры можно разделить на 2 группы: статические и динамические. К статическим объектам относятся: кирпичные стены, железные стены, кусты, ледяная поверхность, водная поверхность, а так же база игрока. Опишем подробно каждый из них.

Кирпичная стена - объект, имеющий базовую прочность, т.е. разрушается за 1 попадание танкового снаряда. Разрушается лишь четверть объект, который образуется из объектов данного типа. При попадании в стык разделения объекта будут уничтожены сразу две четверти. Однако, при наличии у танка 3-ёх звёзд улучшения, будет разрушено сразу половина объекта стены с той стороны, с которой было совершено попадание(вне зависимости от точки попадания).

Железная стена - объект, имеющий самую большую прочность в игре среди статических объектов. Разрушить четверть данного объекта можно только при наличии у танка 3-ёх звёзд улучшения.

Кусты - объект, не имеющий прочность. Его свойство заключается в том, что он понижает видимость танков и летящих снарядов, а так же звёзд улучшения танков.

Ледяная поверхность - объект, не имеющий прочности. Его свойство заключается в том, что он понижает управляемость(в обычной ситуации инструкция на поворот танка в сторону обрабатывается сразу, однако на данном объекте поворот танка занимает некоторое время).

Водная поверхность - объект, не имеющий прочности. Его свойство заключается в том, что по нему невозможно передвижение танков, однако возможно ведение огня.

База игрока - главный объект карты и игры. Он единственный из статических объектов, который является единой неделимой частью. Разрушается от одного попадания танка(как вражеского, так и танка игрока).

Следующая группа объектов - динамические объекты. К ним относятся танки, улучшающие звёзды для танков, а так же выпущенные снаряды на момент их существования.

Танки - главные динамические объекты в игре. Они не делятся на части, так же, как и база игрока. Танки можно разделить на две группы: танки игрока(в зависимости от имеющейся улучшающей звезды) и танки противника.

Танки противника - боты, управляемые алгоритмом, который будет рассмотрен далее. Танки противника делятся на 4 группы:

1. Обычный танк - танк, имеющий 100 очков прочности, обладает стандартной скоростью, стандартной скоростью ведения огня и уроном в 100 единиц. За его уничтожение игрок получает 100 очков.
2. Бронетранспортёр - танк, имеющий 100 очков прочности, обладает повышенной скоростью, стандартной скоростью ведения огня и уроном в 100 единиц. За его уничтожение игрок получает 200 очков.
3. Скорострельный танк - танк, имеющий 100 очков прочности, обладает стандартной скоростью, повышенной скоростью ведения огня и уроном в 100 единиц. За его уничтожение игрок получает 300 очков.
4. Тяжёлый танк - танк, имеющий 400 очков прочности, обладает стандартной скоростью, повышенной скоростью ведения огня и уроном в 100 единиц. За его уничтожение игрок получает 400 очков.

Всего за волну появляется 20 вражеских танков. 4-тый, 11-ый и 18-ый танки особенные. За их уничтожение на карту для игрока выпадают бонусы, которые игрок может забрать до появления следующего танка с вышеопределенным номером. Данные танки так же отмечены особым цветом. Одновременно на карте не может находится более 4-ых танков противника.

Танк игрока - объект, управляемый игроком. В зависимости от звезд улучшения имеется 4 варианта танка:

1. 0 звёзд улучшения - малый танк, имеющий 100 очков прочности, стандартную скорость, стандартную скорость ведения огня и урон в 100 единиц.
2. 1 звезда улучшения - лёгкий танк, имеющий 200 очков прочности, стандартную скорость, стандартную скорость ведения огня, повышенную скорость полёта снаряда(данный параметр учитывается при наличии 1-ой улучшающей звезды и более) и урон в 100 единиц.
3. 2 звезды улучшения - средний танк, имеющий 300 очков прочности, стандартную скорость, стандартную скорость ведения огня(так же только для этого типа танка доступен режим стрельбы очередями(данный параметр учитывается при наличии 2-ух улучшающих звезды и более)) и урон в 100 единиц.
4. 3 звезды улучшения - тяжёлый танк, имеющий 400 очков прочности, стандартную скорость, стандартную скорость ведения огня и урон в 100 единиц, а так же способность пробивать железные стены, которая доступна лишь при имении 3-ёх звёзд улучшения

Летящий снаряд - объект, существующий до тех пор, пока не попадёт в какой либо другой объект. При попадании в такой же снаряд они взаимно уничтожают друг друга.

Отдельным объектом являются бонусы - объекты, при подборе которых игрок получает 500 очков.

При переходе на следующий уровень игрок получает 1000 очков.

Преимущества такой задумки:

1. События на карте происходят очень быстро, что развивает скорость реакции и скорость принятия решений.
2. Динамичность и разнообразие уровней завлекают играть больше, а повышение сложности стимулирует к повышению навыков игрока, что, соответственно, повышает время пользователя, проведённое в игре.

Недостатки:

1. Сложность игры так же может оттолкнуть пользователя, уменьшив его время, проведённое в игре.

Отдельно будет рассмотрен алгоритм реализации искусственного интеллекта в игре. Танки ботов двигаются в случайном направлении, а выстрел производят в случайный момент времени.

Преимущества данного алгоритма:

1. Поведение ботов становится непредсказуемым, что затрудняет задачу уничтожения для игрока.
2. Простая реализация.

Недостатки:

1. В редких случаях боты могут застрять в объектах карты, что не позволит им добраться до базы игрока.

**1.2. Назначение разработки**

Целью разработки данного игрового программного средства является создание игры, которая будет завлекать игрока своей динамичностью, разнообразием карт и соревновательным тоном, чтобы обогнать других игроков в таблице рекордов.

**1.3. Состав выполняемых функций**

1. Отображение меню:
   1. Отображение главного меню
   2. Отображение легенды управления

1.3) Отображение таблицы лидеров

1. Отображение и обновление карты
2. Отображение и обновление положения игрока и противников
3. Отображение и обновление положения снарядов танков
4. Система взаимодействия снарядов и танков
5. Взаимодействие с объектами карты
6. Система перехода с уровня на уровень и начисление очков
7. Система авторизации пользователей

**1.4. Входные данные**

Главное меню, файлы с картами, картинками и данными пользователей.

**1.5. Выходные данные**

Полноценная игра-аркада с системой пользователей и таблицей лидеров.

**2.АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**

**2.1. Описание функциональности системы**

Функционал разрабатываемой системы должен предоставлять пользователю возможность удобной, интуитивно-понятной и захватывающей работы с программным средством.

При запуске программы пользователю выводится главное меню, где он может просмотреть информацию о создателе, легенду управления, таблицу лидеров, окно авторизации или начать игру. В случае выбора окна авторизации пользователь вводит данные от профиля и авторизовывается, либо создаёт новый аккаунт.

При авторизации программа подгружает список пользователей, который содержит информацию о каждом из них, а именно: никнейм, пароль и счет. При вводе существующего никнейма пользователю необходимо ввести верный пароль для успешной авторизации. Если пользователь введет несуществующий никнейм, то программа авторизует нового пользователя.

При просмотре таблицы лидеров программа подгружает список пользователей, сортирует его по убыванию очков, и выводит на экран формы.

Когда пользователь авторизован и нажимает кнопку начала игры, то начинается игровой цикл из пяти уровней. Уровни и картинки объектов находятся в файлах игры.

На каждом уровне находится по четыре противника, каждый из них имеет в запасе пять жизней. Игрок имеет по три жизни на каждый уровень. За уничтожение противника игроку начисляется сто очков. При переходе на следующий уровень игроку начисляется тысяча очков, а при полном прохождении игры игроку начисляется пять тысяч очков. После завершения игрового цикла или выхода из игрового процесса накопленные очки начисляются на профиль игрока.

При завершении работы с программой пользователь нажимает крестик, и программа закрывается.

Интерфейс разработан интуитивно-понятным, с минимальным функционалом, которого хватает для комфортного взаимодействия.

**2.2. Спецификация функциональных требований**

Функции программного средства должны быть реализованы в соответствии со следующими требованиями:

* Функция главного меню

Функция должна отобрать главное меню ПС и корректно обрабатывать взаимодействие с элементами меню.

* Функция отображения легенды управления

Функция должна выводить на экран элементы, которыми осуществляется управление в игровом цикле.

* Функция отображения таблицы лидеров

Функция должна выводить на экран таблицу лидеров, отсортированную по убыванию. Вверху – игроки с наибольшим количеством очков.

* Функция авторизации

Функция принимает на вход никнейм и пароль, а результатом является одно из трех возможных событий: создание нового профиля, авторизация в существующий профиль, сообщение о вводе неправильного пароля.

* Функция начала игры

Функция запускает игровой цикл.

* Функция обновления карты

Функция обновляет карту в соответствии с текущей информацией.

* Функция перемещения игрока и противников

Функция отслеживает перемещение игрока и противников, отрисовывает их на карте, а так же проверяет на валидность перемещения(выход за границу, проезд через статические объекты, которые не предполагают проезда через их текстуру, столкновение с другими танками).

* Функция перемещения и уничтожения снарядов

Функция отслеживает перемещение снарядов, отрисовывает их на карте, а так же проверяет на валидность перемещения(выход за границу, пролет через статические объекты, которые не предполагают проезда через их текстуру, столкновение с другими танками). При попадании в статический объект или танк они уничтожаются(кроме блока стали).

* Функция взаимодействия динамических объектов(танков и снарядов) со статическими объектами(кирпич, сталь, вода, куст, база игрока)

Функция обрабатывает следующие события:

1. Проезд танка через кирпич, сталь, воду(проезд невозможен)
2. Проезда танка через куст(проезд возможен, куст скрывает танк на карте)
3. Пролет снаряда через кирпич, воду(кирпич при попадании разрушается, через воду пролет возможен)
4. Уничтожение базы игрока(игровой цикл оканчивается поражением игрока)

* Функция взаимодействия динамических объектов

Функция обрабатывает следующие события:

1. Уничтожение танка(при попадании снаряда танк уничтожается и теряет жизнь. При потере игроком трех выделенных на уровень жизней игровой цикл оканчивается поражением игрока)
2. Уничтожение снаряда(при попадании снаряда в любой объект он уничтожается).

* Функция перехода на следующий уровень

Функция должна осуществлять переход на следующий уровень(при прохождении всех уровней игры цикл оканчивается победой игрока).

* Функция начисления очков

Функция ведет подсчет очков, заработанных игроком, и записывает их в соот. профиль. При уничт. противника игрок получает сто очков, при прох. уровня – тысячу, при прохождении игры – пять тысяч очков соот.

* Функция возрождения

Функция должна проверять наличие жизней у уничтоженного танка, и если в запасе еще есть жизни, возрождать танк в его точке возрождения.

* Функция задания направления движения ИИ

Функция должна задавать направление вражеских танков(ИИ) независимо друг от друга.

* Функция загрузки карты из файла

Функция должна корректно загружать текущую игровую карту.

* Функция начала\конца уровня

Функция должна корректно создавать\уничтожать объекты карты, проверять уровень на то, пройден ли он.

* Функция считывания клавиш управления

Функция должна считывать нажимаемые пользователем клавиши, игнорируя те, которые не представлены в легенде управления и которые не оказывают влияния на игровой процесс.

* Функция инициализации файлов игры

Функция должна корректно загружать и подготавливать к использованию необходимые файлы игры.

* Функция закрытия программы

Функция должна очищать загруженные файлы, объекты, формы и списки, и корректно совершать выход.

**3.ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

Схемы алгоритмов представлены в приложении А. Ниже представлены описания алгоритмов программного средства.

**3.1 Основной алгоритм программы(рисунок 1.1)**

Основным алгоритмом программы является игровой цикл, который представляет собой прохождение всех уровней игры. Программа отрисовывает карту, читает данные игрока и ИИ, обрабатывает их, изменяет параметры соот. объектов. Также идет проверка на завершение уровня, и на завершение игрового цикла.

**3.2 Главное меню(рисунок 1.2)**

Алгоритм отображает на главной форме средства взаимодействия с функциями меню, а так же выполняет обработку событий, которые вызываются нажатием соответствующей кнопки.

**3.3 Легенда управления**

При нажатии кнопки легенды управления на экран выводится сообщение с объяснением клавиш и их назначения.

**3.4 Таблица лидеров(рисунок 1.3)**

При нажатии кнопки таблицы лидеров открывается форма с таблицей. Перед выводом пользователей список предварительно сортируется по убыванию по критерию набранных очков.

**3.5 Авторизация(рисунок 1.4)**

При нажатии кнопки авторизации открывается окно формы с двумя полями: никнейм и пароль. При подтверждении ввода пользователь либо создает новый профиль и авторизуется в нем, либо авторизуется в существующем профиле при вводе правильного пароля, либо сообщение о неверном пароле.

**3.6 Начало игры**

При нажатии кнопки начала игры запускается процедура начала уровня.

**3.7** **Начало уровня(рисунок 1.5)**

Алгоритм загружает карту текущего уровня, далее выводит ее на экран и запускает игровой цикл.

**3.8 Конец уровня(рисунок 1.6)**

Алгоритм уничтожает все динамические объекты, и проверяет как закончился уровень. Если уровень пройден, то совершается переход на следующий уровень, иначе выводится сообщение о поражении. При прохождении последнего уровня выводится сообщение о прохождении игры.

**3.9 Обновление информационной панели**

Алгоритм периодически обновляет данные информационной панели.

**3.10 Перемещение танков(рисунок 1.7)**

Алгоритм обрабатывает движение танков в соответствии с направлением движения, а так же проверяет возможность движения в соответствии с картой.

**3.11 Перемещение снарядов(рисунок 1.8)**

Алгоритм обрабатывает движение снарядов в соответствии с направлением, проверяет снаряд на попадание

**3.12 Разрушение объектов(рисунок 1.9)**

Алгоритм проверяет снаряды на попадание в объекты карты, и в зависимости от результатов проверки происходит уничтожение тех или иных объектов, в которые попал снаряд.

**3.13 Возрождение танка(рисунок 1.10)**

Алгоритм проверяет наличие жизней у уничтоженного танка, и если их больше 0, то возрождает танк. Если у игра 0 жизней, игровой цикл останав.

**3.15 Выход из программы**

Алгоритм очищает все подгруженные файлы, и производит выход из программы.

**4.СОЗДАНИЕ ПРОГРАММНОГО СРЕДСВА**

**4.1 Проектирование модулей программного средства**

Программное средство разрабатывается на основе спроектированных алгоритмов(см. раздел 3) и спецификации функциональных требований(см. раздел 2). ПС имеет следующие модули:

1. UAuthentication – модуль авторизации
2. UEnemyShells – модуль вражеских снарядов
3. UEnemyTanks – модуль вражеских танков
4. UFileRouthine – модуль взаимодействия с файлами, которые содержат информацию о пользователях
5. UGameInterface – модуль обработки игрового цикла
6. UGameMap – модуль отрисовки и работы с файлами карты и игры
7. ULeadersTable – модуль с таблицей лидеров
8. UMainMenu – модуль главного меню
9. UShellType – модуль снаряда игрока
10. UTTankType – модуль танка игрока

Код программного средства представлен в приложении Б.

**4.2 Разработка модулей программного средства**

Таблица 1 – Описание используемых подпрограмм

|  |  |  |
| --- | --- | --- |
| 1.Имя и тип подпрограммы | 2.Назначение подпрограммы | 3.Параметры и их назначение |
| procedure TMainMenu.FormActivate(Sender: TObject); | Запуск программы | Sender – объект, запускающий процедуру |
| procedure TMainMenu.FormClose(Sender: TObject; var Action: TCloseAction); | Закрытие программы | Sender – объект, запускающий процедуру |
| procedure TMainMenu.FormCloseQuery(Sender: TObject; var CanClose: Boolean); | Подтверждение закрытия программы | Sender – объект, запускающий процедуру |
| procedure TMainMenu.GameNameClick(Sender: TObject); | Информация о разработчике | Sender – объект, запускающий процедуру |
| procedure TMainMenu.StartGameClick(Sender: TObject); | Начало игрового цикла | Sender – объект, запускающий процедуру |
| 1 | 2 | 3 |
| procedure TMainMenu.AuthenticationClick(Sender: TObject); | Запуск авторизационного окна | Sender – объект, запускающий процедуру |
| procedure TMainMenu.LeadsTableClick(Sender: TObject); | Запуск таблицы лидеров | Sender – объект, запускающий процедуру |
| procedure TMainMenu.ControlsDescClick(Sender: TObject); | Легенда управления | Sender – объект, запускающий процедуру |
| procedure TAuth.ConfirmClick(Sender: TObject); | Подтверждение авторизации | Sender – объект, запускающий процедуру |
| procedure TAuth.FormClose(Sender: TObject; var Action: TCloseAction); | Выход из окна авторизации | Sender – объект, запускающий процедуру |
| procedure TLeadersTable.FormActivate(Sender: TObject); | Вывод таблицы лидеров | Sender – объект, запускающий процедуру |
| procedure TLeadersTable.FormClose(Sender: TObject; var Action: TCloseAction); | Выход из таблицы лидеров | Sender – объект, запускающий процедуру |
| procedure TGameInterface.FormActivate(Sender: TObject); | Инициализация игровых параметров | Sender – объект, запускающий процедуру |
| procedure TGameInterface.FormClose(Sender: TObject; var Action: TCloseAction); | Выход из игрового окна | Sender – объект, запускающий процедуру |
| procedure TGameInterface.FormKeyDown(Sender: TObject; var Key: Word;  Shift: TShiftState); | Считывание нажатий клавиш пользователя | Sender – объект, запускающий процедуру  Key – код клавиши |
| procedure TGameInterface.LevelInitTimer(Sender: TObject); | Запуск уровня | Sender – объект, запускающий процедуру |
| procedure TGameInterface.UpdaInfoPanel(Sender: TObject); | Обновление информационной панели | Sender – объект, запускающий процедуру |
| 1 | 2 | 3 |
| procedure TGameInterface.StartLevel(Sender: TObject; path: string); | Инициализация значений для текущего уровня | Sender – объект, запускающий процедуру  Path – путь к файлу с игровой картой |
| procedure TGameInterface.EndGame(Sender: TObject; win: boolean); | Конец уровня | Sender – объект, запускающий процедуру  Win – состояние прохождения уровня |
| procedure TGameInterface.DExpBigBase(Sender: TObject); | Уничтожение базы | Sender – объект, запускающий процедуру |
| procedure TGameInterface.TankMovement(Sender: TObject); | Отслеживание перемещения танка | Sender – объект, запускающий процедуру |
| procedure TGameInterface.RespawnTimer(Sender: TObject); | Возрождение танка | Sender – объект, запускающий процедуру |
| procedure TGameInterface.ShellMovement(Sender: TObject); | Отслеживание перемещения снаряда | Sender – объект, запускающий процедуру |
| procedure TGameInterface.DExpSmall(Sender: TObject); | Разрушение объекта | Sender – объект, запускающий процедуру |
| procedure TGameInterface.ExpBig(Sender: TObject); | Уничтожение танка | Sender – объект, запускающий процедуру |
| procedure DrawBackGround(Screen: TImage); | Отрисовка фона | Screen – игровой экран |
| procedure StaticObjImgArrInit(var StaticObjImg: TStaticObjImgArr); | Инициализация файлов статических объектов | StaticObjImg – массив статических объектов |
| procedure LoadMapFromFile(Screen: TImage; path: string); | Загрузка карты из файла | Screen – игровой экран  Path- путь к файлу карты |
| procedure PlayerTankImgArrInit(); | Инициализация файлов игрока | - |
| 1 | 2 | 3 |
| Procedure TankShoot | Выстрел танка | - |
| Procedure TankMove | Передвижение танка | Screen – игровой экран |
| Procedure CreateTank | Создание танка | X, Y – положение на карте |
| procedure ExpImgInit(); | Инициализация файлов взрыва | - |
| Procedure CreateShell(direction, x, y: integer); | Создание снаряда | X, Y – положение на карте  Direction - направление |
| Procedure MoveShell(screen: TImage); | Передвижение снаряда | Screen – игровой экран |
| procedure EnemyTankImgArrInit(); | Инициализация файлов ИИ | - |
| procedure AddUser(); | Добавление пользователя | - |
| procedure CheckUser(str1, str2: string); | Проверка авторизации | Str1 – ник  Str2 - пароль |
| procedure ReadUsersData(); | Чтение данных из файла | - |
| procedure SaveData(); | Сохранение данных в файл | - |

**5.ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ**

**5.1 Тестирование ПС**

Таблица 2 – Тестирование программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| 1.№ | 2.Условия тестирования | 3.Ожидаемый результат | 4.Полученный результат |
| 1 | Запуск программы | Запуск программы без ошибок | Тест пройден |
| 2 | Отображение таблицы лидеров | Вывод на экран таблицы лидеров | Тест пройден |
| 3 | Отображение легенды управления | Вывод на экран легенды управления | Тест пройден |
| 4 | Авторизация несущ. пользователя | Создание нового профиля | Тест пройден |
| 5 | Авторизация сущ. пользователя | Пользователь авторизован | Тест пройден |
| 6 | Запуск игрового цикла | Игра запущена | Тест пройден |
| 7 | Обработка игрового цикла | Функции игрового цикла работают без ошибок | Тест пройден |
| 8 | Загрузка файлов игры | Файлы успешно подгружены | Тест пройден |
| 9 | Прохождение уровня | Уровень пройден, новый уровень запущен | Тест пройден |
| 10 | Поражение в игре | Вывод экрана поражения | Тест пройден |
| 11 | Победа в игре | Вывод экрана победы | Тест пройден |
| 12 | Начисление очков | Очки начисляются на счет игрока | Тест пройден |
| 13 | Разрушение объектов | Объект разрушен, информация обновлена | Тест пройден |
| 14 | Обработка снарядов | Функции обработки снарядов работают без ошибок | Тест пройден |
| 15 | Обработка карты | Обновление текущей информации о карте | Тест пройден |
| 1 | 2 | 3 | 4 |
| 16 | Обработка танков | Функции обработки танков работают без ошибок | Тест пройден |
| 17 | Уничтожение базы | База уничтожена, поражение в игре | Тест пройден |
| 18 | Выход из игрового цикла | Отображение главного меню | Тест пройден |
| 19 | Работа с файлами пользователей | Чтение и запись данных работают корректно | Тест пройден |
| 20 | Выход из программы | Подгруженные файлы очищены, программа успешно закрыта | Тест пройден |

**5.2 Анализ полученных результатов**

Программа работает корректно, все взаимодействия соответствуют функциональным требованиям. Единственная проблема в том, что на маломощных ПК игра может работать с некоторыми торможениями(ввиду многопоточности для оптимальном работы необходим многоядерный процессор).

**6.РУКОВОДСТВО ПО УСТАНОВКЕ И ЭКСПЛУАТАЦИИ**

Для того, чтобы установить ПС, необходимо скачать Zip-папку “BattleCityRemake”, извлечь из нее все файл и запустить файл BattleCity.Exe(для удобства можно создать ярлык)(рисунок 2.1 и 2.2).

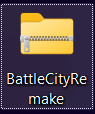
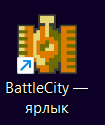
**** 

Рисунок 2.1 – Zip-Файл игры Рисунок 2.2 – Ярлык игры

При запуске программы нас встречает главное меню(рисунок 2.3).



Рисунок 2.3 – Главное меню

При нажатии кнопки “Легенда управления” на экран выводится легенда управления (рисунок 2.4).

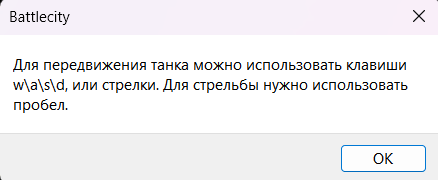


Рисунок 2.4 – Легенда управления

При нажатии на кнопку “Таблица лидеров” на экран выводится форма с таблицей лидеров(рисунок 2.5)

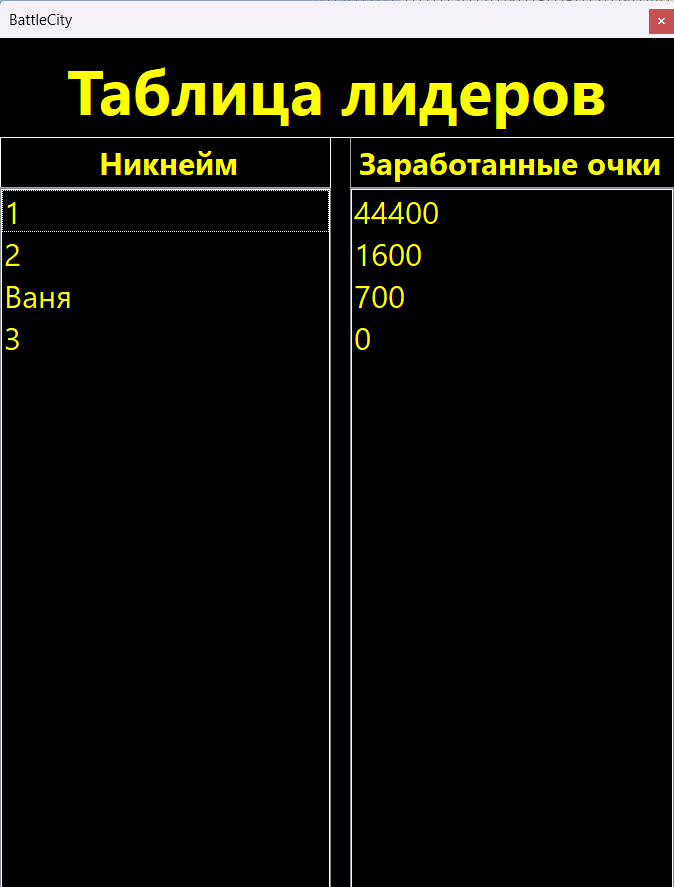


Рисунок 2.5 – Таблица лидеров.

При нажатии кнопки “Авторизация” на экран выводится форма с окном авторизации(рисунок 2.6).

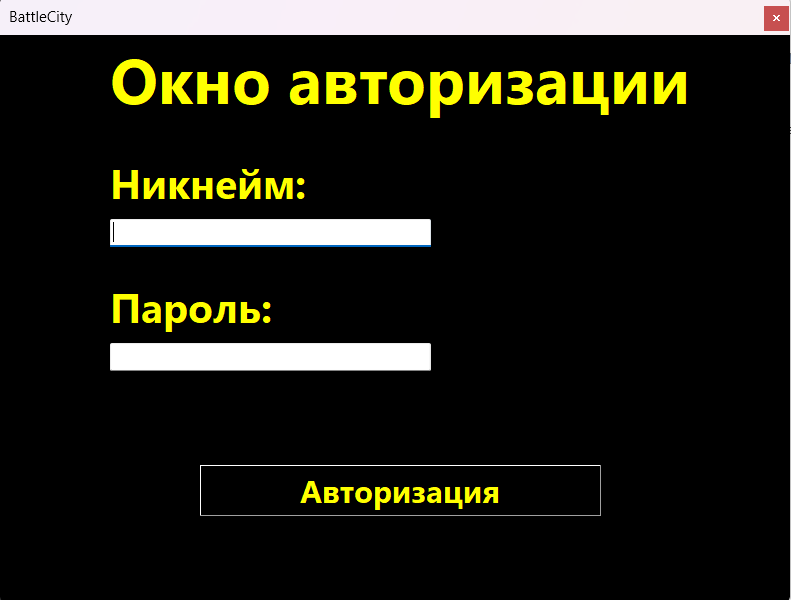


Рисунок 2.6 – Окно авторизации

При авторизации пользователя выводится соответствующее сообщение(рисунок 2.7).

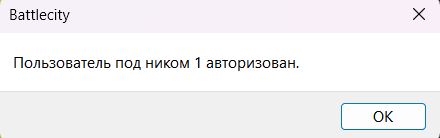


Рисунок 2.7 – Сообщение об авторизации пользователя

При нажатии кнопки “Начать игру” запускается игровой цикл и выводится форма с игрой(рисунок 2.8).

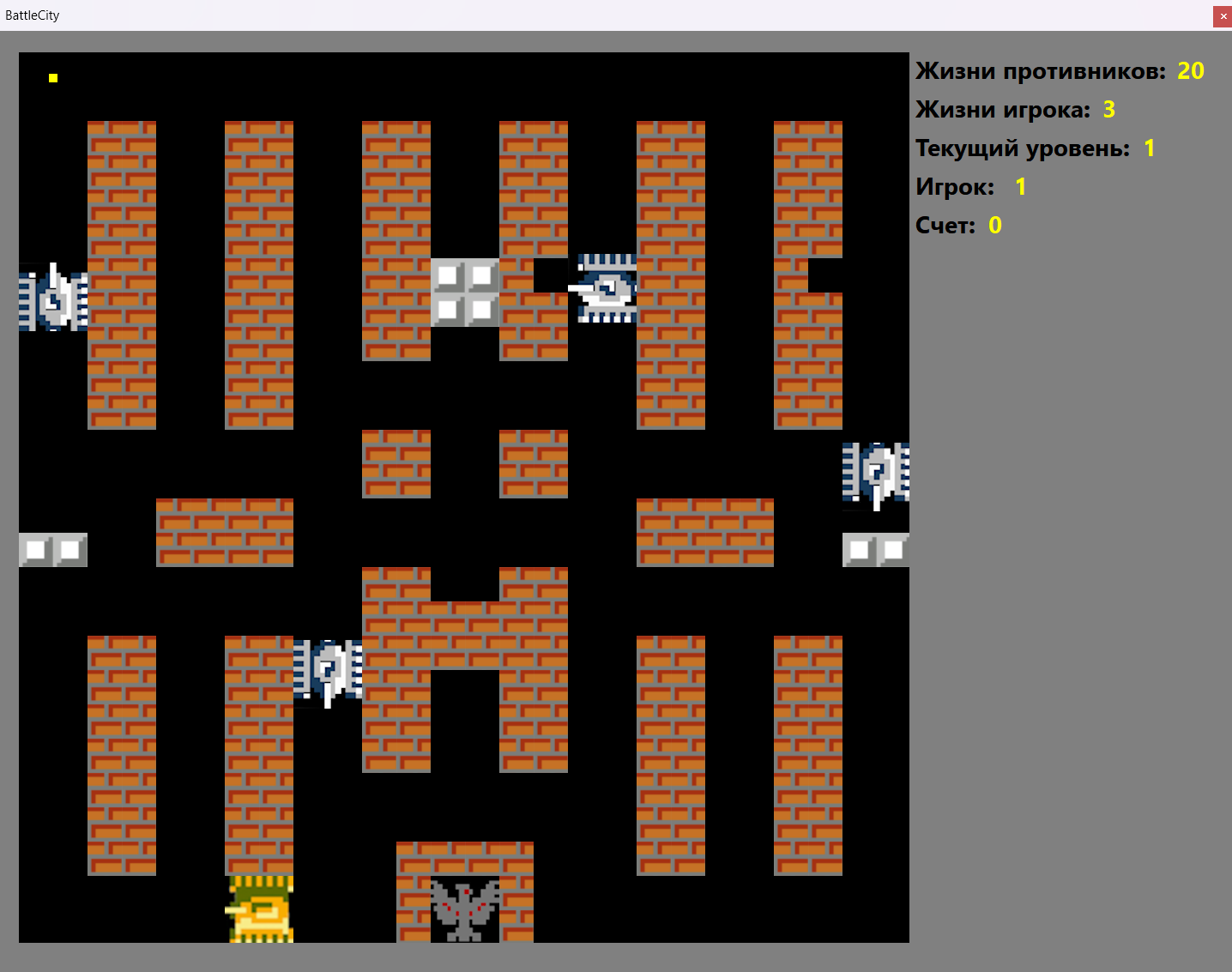


Рисунок 2.8 – Окно игры

Выход из программы осуществляется на крестик. При нажатии крестика на экране появляется диалоговое окно выхода(рисунок 2.9).

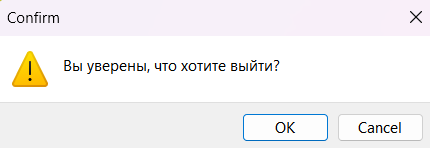


Рисунок 2.9 – Диалоговое окно выхода

**ЗАКЛЮЧЕНИЕ**

При разработке данного программного средства были изучены следующие компоненты:

1. Возможность обработки многих объектов одновременно(многопоточность)
2. Реализация графического интерфейса при помощи класса TCanvas
3. Взаимодействие со многими компонентами VCL Forms
4. Разбиение всего кода на отдельные части в целях лучшей читаемости и структурированности
5. Работа с графическими, текстовыми файлами.
6. Изучение и применение на практике принципов ООП
7. Было разработано вполне производительное ПС.

Был проведен анализ существующих решений, создание собственного решения и сравнение с имеющимися аналогами.

Разработанное ПС имеет перспективу в разработке, т.к. в нем реализованы не все функции оригинальной игры, а также возможны авторские разработки.

Автор надеется, что разработанное им программное средство понравится как молодому поколению, так и более старшему. Людям, которые хотели бы вновь окунуться в любимую игру детства.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

[1]BattleCity – статья на Wikipedia: <https://ru.wikipedia.org/wiki/Battle_City>

[2]BattleCity – сайт-Фандом: <https://vertigosn.ru/solutions/dendy/battlecity>

[3] Вирт Н. Алгоритмы и структуры данных. – М.: Мир, 1989 – 360 с.

[4] Серебряная Л. В. Структуры и алгоритмы обработки данных: учеб.-метод. пособие / Л.В. Серебряная, И.М. Марина. –Минск : БГУИР, 2013. – 51c.

[5] Фаронов В.В. Турбо Паскаль 7.0. Начальный курс. Учеб. Пособие. – М.: Ноллидж, 2009.

[6]Документация к среде разработки Delphi 7. [Встроенный справочный ресурс]. (Дата обращения 02.05.2024).

[7] Конспекты лекций Д. Е. Оношко по дисциплине ОАиП за 2023 и 2024.

**ПРИЛОЖЕНИЕ А**

**Блок-схемы алгоритмов**



Рисунок 1.1 – Основная схема алгоритма



Рисунок 1.2 – Главное меню



Рисунок 1.3 – Таблица лидеров



Рисунок 1.4 – Авторизация



Рисунок 1.5 – Начало уровня



Рисунок 1.6 – Конец уровня



Рисунок 1.7 – Перемещение танков



Рисунок 1.8 – Перемещение снарядов



Рисунок 1.9 – Разрушение объектов



Рисунок 1.10 – Возрождение танков

**ПРИЛОЖЕНИЕ В**

**Код программы**

unit UMainMenu;

interface

uses

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

System.Classes, Vcl.Graphics,

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

UFileRouthine;

type

TMainMenu = class(TForm)

MenuPanel: TPanel;

CaptionPanel: TPanel;

GameName: TLabel;

StartGameButton: TPanel;

StartGameCaption: TLabel;

ControlsDescription: TPanel;

LeadsTable: TPanel;

Authentication: TPanel;

procedure StartGameCaptionClick(Sender: TObject);

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

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

procedure GameNameClick(Sender: TObject);

procedure ControlsDescriptionClick(Sender: TObject);

procedure LeadsTableClick(Sender: TObject);

procedure AuthenticationClick(Sender: TObject);

procedure FormActivate(Sender: TObject);

end;

var

MainMenu: TMainMenu;

isAuthorised: Boolean;

isReaded: Boolean = false;

implementation

{$R \*.dfm}

uses

UGameInterface, ULeadersTable, UAuthentication;

procedure TMainMenu.FormActivate(Sender: TObject);

begin

if not isReaded then

begin

ReadUsersData();

isReaded := true;

end;

end;

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

begin

SaveData();

MainMenu.Destroy;

end;

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

begin

if MessageDlg('Вы уверены, что хотите выйти?', mtConfirmation,

[mbOk, mbCancel], 0) = mrCancel then

CanClose := false;

end;

procedure TMainMenu.GameNameClick(Sender: TObject);

begin

ShowMessage

('Программное средство было разработано студентом первого курса ПИ 2023/2024 - Кириллом Зиновенко.');

end;

procedure TMainMenu.StartGameCaptionClick(Sender: TObject);

begin

if isAuthorised then

begin

GameInterface := TGameInterface.Create(Owner);

// MainMenu.Hide;

end

else

ShowMessage('Для начала игры необходимо авторизоваться.');

end;

procedure TMainMenu.AuthenticationClick(Sender: TObject);

begin

Auth := TAuth.Create(Owner);

// MainMenu.Hide;

end;

procedure TMainMenu.LeadsTableClick(Sender: TObject);

begin

LeadersTable := TLeadersTable.Create(Owner);

// MainMenu.Hide;

end;

procedure TMainMenu.ControlsDescriptionClick(Sender: TObject);

begin

ShowMessage

('Для передвижения танка можно использовать клавиши w\a\s\d, или стрелки. Для стрельбы нужно использовать пробел.')

end;

end.

unit ULeadersTable;

interface

uses

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

System.Classes, Vcl.Graphics,

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

type

TLeadersTable = class(TForm)

LeadsDespription: TPanel;

NameDesc: TPanel;

ScoreDesc: TPanel;

BuffPanel: TPanel;

ListBox1: TListBox;

ListBox2: TListBox;

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

procedure FormActivate(Sender: TObject);

end;

var

LeadersTable: TLeadersTable;

implementation

uses

UMainMenu;

{$R \*.dfm}

procedure TLeadersTable.FormActivate(Sender: TObject);

var

i, j: integer;

temp: TListData;

begin

for i := 0 to length(header) - 2 do

for j := i + 1 to length(header) - 1 do

if header[j].Score > header[i].Score then

begin

temp := header[j];

header[j] := header[i];

header[i] := temp;

end;

for i := 0 to length(header) - 1 do

begin

listbox1.Items.Add(header[i].Name);

listbox2.Items.Add(IntToStr(header[i].Score));

end;

end;

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

begin

listbox1.Clear;

listbox2.Clear;

self.Destroy;

MainMenu.Show;

end;

end.

unit UAuthentication;

interface

uses

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

System.Classes, Vcl.Graphics,

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

UFileRouthine;

type

TAuth = class(TForm)

AuthPanel: TPanel;

AuthDesc: TLabel;

NickNameDesc: TLabel;

PassWordDesc: TLabel;

NickNameEnter: TEdit;

PasswordEnter: TEdit;

Confirm: TPanel;

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

procedure ConfirmClick(Sender: TObject);

end;

var

Auth: TAuth;

implementation

uses

UMainMenu;

{$R \*.dfm}

procedure TAuth.ConfirmClick(Sender: TObject);

var

NickNameStr, PassWordStr: string;

begin

NickNameStr := NickNameEnter.Text;

PassWordStr := PasswordEnter.Text;

CheckUser(NickNameStr, PassWordStr)

end;

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

begin

self.Destroy;

Mainmenu.Show;

end;

end.

unit UGameInterface;

interface

uses

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

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, UGameMap, UTTankType,

UShellType, UEnemyTanks, UEnemyShells, UMainMenu, UFileRouthine,

System.Actions, Vcl.ActnList, Vcl.StdCtrls;

type

TGameInterface = class(TForm)

PaddingPanelUp: TPanel;

PaddingPanelLeft: TPanel;

PaddingPanelBottom: TPanel;

GameInfo: TPanel;

GameScreen: TImage;

PlayerTankMovement: TTimer;

PlayerShellMovement: TTimer;

DeleteExpSmallPlayer1: TTimer;

DeleteExpSmallPlayer2: TTimer;

DeleteExpBigBase: TTimer;

EnemyTank1Movement: TTimer;

EnemyTank1SetDirection: TTimer;

EnemyTank2Movement: TTimer;

EnemyTank2SetDirection: TTimer;

EnemyTank3Movement: TTimer;

EnemyTank4Movement: TTimer;

EnemyTank3SetDirection: TTimer;

EnemyTank4SetDirection: TTimer;

DeleteExpBigPlayer: TTimer;

DeleteExpBigEnemy1: TTimer;

DeleteExpBigEnemy2: TTimer;

DeleteExpBigEnemy3: TTimer;

DeleteExpBigEnemy4: TTimer;

Enemy1ShellMovement: TTimer;

DeleteExpSmallEnemy11: TTimer;

DeleteExpSmallEnemy12: TTimer;

Enemy2ShellMovement: TTimer;

DeleteExpSmallEnemy21: TTimer;

DeleteExpSmallEnemy22: TTimer;

Enemy3ShellMovement: TTimer;

DeleteExpSmallEnemy31: TTimer;

DeleteExpSmallEnemy32: TTimer;

Enemy4ShellMovement: TTimer;

DeleteExpSmallEnemy41: TTimer;

DeleteExpSmallEnemy42: TTimer;

Enemy1Shoot: TTimer;

Enemy2Shoot: TTimer;

Enemy3Shoot: TTimer;

Enemy4Shoot: TTimer;

PlayerRespawn: TTimer;

Enemy1Respawn: TTimer;

Enemy2Respawn: TTimer;

Enemy3Respawn: TTimer;

Enemy4Respawn: TTimer;

LevelInit: TTimer;

UpdateInfoPanel: TTimer;

EnemyLivesDespription: TLabel;

EnemyLives: TLabel;

PlayerLivesDescription: TLabel;

PlayerLives: TLabel;

GameCompleted: TLabel;

CurrentLevelDescription: TLabel;

CurrentlevelInfo: TLabel;

PlayerNameDescription: TLabel;

ScoreDescription: TLabel;

PlayerName: TLabel;

Score: TLabel;

WinLabel: TLabel;

procedure EndGame(Sender: TObject; win: boolean);

procedure StartLevel(Sender: TObject; path: string);

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

procedure PlayerTankMovementTimer(Sender: TObject);

procedure PlayerShellMovementTimer(Sender: TObject);

procedure DeleteExpSmallPlayer1Timer(Sender: TObject);

procedure DeleteExpSmallPlayer2Timer(Sender: TObject);

procedure DeleteExpBigBaseTimer(Sender: TObject);

procedure EnemyTank1SetDirectionTimer(Sender: TObject);

procedure EnemyTank1MovementTimer(Sender: TObject);

procedure EnemyTank2MovementTimer(Sender: TObject);

procedure EnemyTank2SetDirectionTimer(Sender: TObject);

procedure EnemyTank3MovementTimer(Sender: TObject);

procedure EnemyTank3SetDirectionTimer(Sender: TObject);

procedure EnemyTank4MovementTimer(Sender: TObject);

procedure EnemyTank4SetDirectionTimer(Sender: TObject);

procedure DeleteExpBigPlayerTimer(Sender: TObject);

procedure DeleteExpBigEnemy1Timer(Sender: TObject);

procedure DeleteExpBigEnemy2Timer(Sender: TObject);

procedure DeleteExpBigEnemy3Timer(Sender: TObject);

procedure DeleteExpBigEnemy4Timer(Sender: TObject);

procedure Enemy1ShellMovementTimer(Sender: TObject);

procedure Enemy2ShellMovementTimer(Sender: TObject);

procedure Enemy3ShellMovementTimer(Sender: TObject);

procedure Enemy4ShellMovementTimer(Sender: TObject);

procedure DeleteExpSmallEnemy11Timer(Sender: TObject);

procedure DeleteExpSmallEnemy12Timer(Sender: TObject);

procedure DeleteExpSmallEnemy21Timer(Sender: TObject);

procedure DeleteExpSmallEnemy22Timer(Sender: TObject);

procedure DeleteExpSmallEnemy31Timer(Sender: TObject);

procedure DeleteExpSmallEnemy32Timer(Sender: TObject);

procedure DeleteExpSmallEnemy41Timer(Sender: TObject);

procedure DeleteExpSmallEnemy42Timer(Sender: TObject);

procedure Enemy1ShootTimer(Sender: TObject);

procedure Enemy2ShootTimer(Sender: TObject);

procedure Enemy3ShootTimer(Sender: TObject);

procedure Enemy4ShootTimer(Sender: TObject);

procedure PlayerRespawnTimer(Sender: TObject);

procedure Enemy1RespawnTimer(Sender: TObject);

procedure Enemy2RespawnTimer(Sender: TObject);

procedure Enemy3RespawnTimer(Sender: TObject);

procedure Enemy4RespawnTimer(Sender: TObject);

procedure UpdateInfoPanelTimer(Sender: TObject);

procedure LevelInitTimer(Sender: TObject);

procedure FormActivate(Sender: TObject);

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

end;

TParam = record

line, column, k: integer;

end;

TDeletePlayerArr = array [1 .. 4] of TParam;

TEnemyArr = array [1 .. 4] of TDeletePlayerArr;

TLivesArr = array [0 .. 4] of integer;

var

GameInterface: TGameInterface;

DeletePlayer: TDeletePlayerArr;

DeleteEnemy: TEnemyArr;

path: string;

pressedKeyCode, kBase, EnemyTankNum, currentLevel, PlayerScore: integer;

lives: TLivesArr;

implementation

{$R \*.dfm}

procedure TGameInterface.FormActivate(Sender: TObject);

begin

randomize;

self.GameScreen.Canvas.Brush.Color := clblack;

self.GameScreen.Canvas.Pen.Color := clblack;

self.GameScreen.Height := MapSize;

self.GameScreen.Width := MapSize;

StaticObjImgArrInit(staticobjimg);

PlayerTankImgArrInit();

EnemyTankImgArrInit();

ExpImgInit();

EnemyTankNum := 1;

currentLevel := 1;

PlayerScore := 0;

path := '..\maps\level' + IntToStr(currentLevel) + '.txt';

GameScreen.Canvas.Rectangle(0, 0, MapSize, MapSize);

self.GameCompleted.Caption := 'Уровень ' + IntToStr(currentLevel);

self.GameCompleted.Visible := true;

self.LevelInit.Enabled := true;

end;

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

var

i: integer;

begin

PlayerTankMovement.Enabled := false;

PlayerShellMovement.Enabled := false;

DeleteExpSmallPlayer1.Enabled := false;

DeleteExpSmallPlayer2.Enabled := false;

DeleteExpBigBase.Enabled := false;

EnemyTank1Movement.Enabled := false;

EnemyTank1SetDirection.Enabled := false;

EnemyTank2Movement.Enabled := false;

EnemyTank2SetDirection.Enabled := false;

EnemyTank3Movement.Enabled := false;

EnemyTank4Movement.Enabled := false;

EnemyTank3SetDirection.Enabled := false;

EnemyTank4SetDirection.Enabled := false;

DeleteExpBigPlayer.Enabled := false;

DeleteExpBigEnemy1.Enabled := false;

DeleteExpBigEnemy2.Enabled := false;

DeleteExpBigEnemy3.Enabled := false;

DeleteExpBigEnemy4.Enabled := false;

Enemy1ShellMovement.Enabled := false;

DeleteExpSmallEnemy11.Enabled := false;

DeleteExpSmallEnemy12.Enabled := false;

Enemy2ShellMovement.Enabled := false;

DeleteExpSmallEnemy21.Enabled := false;

DeleteExpSmallEnemy22.Enabled := false;

Enemy3ShellMovement.Enabled := false;

DeleteExpSmallEnemy31.Enabled := false;

DeleteExpSmallEnemy32.Enabled := false;

Enemy4ShellMovement.Enabled := false;

DeleteExpSmallEnemy41.Enabled := false;

DeleteExpSmallEnemy42.Enabled := false;

Enemy1Shoot.Enabled := false;

Enemy2Shoot.Enabled := false;

Enemy3Shoot.Enabled := false;

Enemy4Shoot.Enabled := false;

PlayerRespawn.Enabled := false;

Enemy1Respawn.Enabled := false;

Enemy2Respawn.Enabled := false;

Enemy3Respawn.Enabled := false;

Enemy4Respawn.Enabled := false;

PlayerShell.direction := -1;

for i := 1 to 4 do

EnemyShells[i].direction := -1;

SetLength(waterObj, 0);

SetLength(steelObj, 0);

SetLength(forestObj, 0);

inc(header[current].Score, PlayerScore);

self.Destroy;

MainMenu.Show;

end;

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

Shift: TShiftState);

begin

pressedKeyCode := Key;

end;

procedure TGameInterface.LevelInitTimer(Sender: TObject);

begin

StartLevel(Sender, path);

self.GameCompleted.Visible := false;

self.LevelInit.Enabled := false;

end;

procedure TGameInterface.UpdateInfoPanelTimer(Sender: TObject);

var

i, sum: integer;

Show: string;

begin

sum := 0;

for i := 1 to 4 do

sum := sum + lives[i];

Show := IntToStr(sum);

if sum = 0 then

EndGame(Sender, true);

self.EnemyLives.Caption := Show;

Show := IntToStr(lives[0]);

self.PlayerLives.Caption := Show;

Show := IntToStr(currentLevel);

self.CurrentlevelInfo.Caption := Show;

Show := header[current].Name;

self.PlayerName.Caption := Show;

Show := IntToStr(PlayerScore);

self.Score.Caption := Show;

end;

procedure TGameInterface.StartLevel(Sender: TObject; path: string);

var

i: integer;

begin

CreatePlayerTank(320, 960);

for i := 1 to 4 do

begin

CreateEnemyTank(EnemyCoordsSpawn[i].X, EnemyCoordsSpawn[i].Y, i);

CreateEnemyShell(EnemyTanks[i].direction, EnemyTanks[i].DP[0].X, EnemyTanks[i].DP[0].Y, i);

end;

DrawBackGround(GameInterface.GameScreen);

LoadMapFromFile(GameInterface.GameScreen, path);

self.UpdateInfoPanel.Enabled := true;

self.PlayerTankMovement.Enabled := true;

self.EnemyTank1Movement.Enabled := true;

self.EnemyTank1SetDirection.Enabled := true;

self.Enemy1Shoot.Enabled := true;

self.EnemyTank2Movement.Enabled := true;

self.EnemyTank2SetDirection.Enabled := true;

self.Enemy2Shoot.Enabled := true;

self.EnemyTank3Movement.Enabled := true;

self.EnemyTank3SetDirection.Enabled := true;

self.Enemy3Shoot.Enabled := true;

self.EnemyTank4Movement.Enabled := true;

self.EnemyTank4SetDirection.Enabled := true;

self.Enemy4Shoot.Enabled := true;

lives[0] := 3;

for i := 1 to 4 do

lives[i] := 5;

end;

procedure TGameInterface.EndGame(Sender: TObject; win: boolean);

var

i: integer;

begin

self.UpdateInfoPanel.Enabled := false;

self.PlayerTankMovement.Enabled := false;

self.PlayerShellMovement.Enabled := false;

self.EnemyTank1Movement.Enabled := false;

self.EnemyTank1SetDirection.Enabled := false;

self.Enemy1ShellMovement.Enabled := false;

self.Enemy1Shoot.Enabled := false;

self.EnemyTank2Movement.Enabled := false;

self.EnemyTank2SetDirection.Enabled := false;

self.Enemy2ShellMovement.Enabled := false;

self.Enemy2Shoot.Enabled := false;

self.EnemyTank3Movement.Enabled := false;

self.EnemyTank3SetDirection.Enabled := false;

self.Enemy3ShellMovement.Enabled := false;

self.Enemy3Shoot.Enabled := false;

self.EnemyTank4Movement.Enabled := false;

self.EnemyTank4SetDirection.Enabled := false;

self.Enemy4ShellMovement.Enabled := false;

self.Enemy4Shoot.Enabled := false;

PlayerShell.direction := -1;

for i := 1 to 4 do

EnemyShells[i].direction := -1;

if win then

begin

inc(currentLevel);

if currentLevel = 6 then

begin

inc(PlayerScore, 6000);

GameScreen.Canvas.Rectangle(0, 0, MapSize, MapSize);

self.WinLabel.Caption := 'Спасибо за прохождение игры!';

self.WinLabel.Visible := true;

end

else

begin

SetLength(waterObj, 0);

SetLength(forestObj, 0);

SetLength(steelObj, 0);

path := '..\maps\level' + IntToStr(currentLevel) + '.txt';

GameScreen.Canvas.Rectangle(0, 0, MapSize, MapSize);

self.GameCompleted.Caption := 'Уровень ' + IntToStr(currentLevel);

inc(PlayerScore, 1000);

self.GameCompleted.Visible := true;

self.LevelInit.Enabled := true;

end;

end

else

begin

GameScreen.Canvas.Rectangle(0, 0, MapSize, MapSize);

self.WinLabel.Caption := 'Повезет в следующий раз...';

self.WinLabel.Visible := true;

end;

end;

/// //////

procedure TGameInterface.DeleteExpBigBaseTimer(Sender: TObject);

begin

if kBase = 1 then

begin

GameScreen.Canvas.Rectangle(ColumnBase \* subobjlen, LineBase \* subobjlen,

ColumnBase \* subobjlen + objlen, LineBase \* subobjlen + objlen);

kBase := 0;

GameScreen.Canvas.Draw(ColumnBase \* subobjlen, LineBase \* subobjlen,

staticobjimg[2]);

self.DeleteExpBigBase.Enabled := false;

EndGame(Sender, false);

end

else

inc(kBase);

end;

/// ////////

procedure TGameInterface.PlayerTankMovementTimer(Sender: TObject);

begin

MovePlayer(pressedKeyCode, GameInterface.GameScreen);

ShootPlayer(pressedKeyCode);

pressedKeyCode := -100;

end;

procedure TGameInterface.PlayerRespawnTimer(Sender: TObject);

begin

if PlayerTank.isDestroyed and (lives[0] > 0) then

begin

dec(lives[0]);

if lives[0] > 0 then

begin

CreatePlayerTank(320, 960);

self.PlayerTankMovement.Enabled := true;

self.PlayerRespawn.Enabled := false;

end

else

EndGame(Sender, false);

end;

end;

procedure TGameInterface.PlayerShellMovementTimer(Sender: TObject);

begin

MovePlayerShell(GameInterface.GameScreen);

end;

procedure TGameInterface.DeleteExpSmallPlayer1Timer(Sender: TObject);

begin

if DeletePlayer[1].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeletePlayer[1].column \* subobjlen,

DeletePlayer[1].line \* subobjlen, DeletePlayer[1].column \* subobjlen +

subobjlen, DeletePlayer[1].line \* subobjlen + subobjlen);

DeletePlayer[1].k := 0;

self.DeleteExpSmallPlayer1.Enabled := false;

end

else

inc(DeletePlayer[1].k);

end;

procedure TGameInterface.DeleteExpSmallPlayer2Timer(Sender: TObject);

begin

if DeletePlayer[2].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeletePlayer[2].column \* subobjlen,

DeletePlayer[2].line \* subobjlen, DeletePlayer[2].column \* subobjlen +

subobjlen, DeletePlayer[2].line \* subobjlen + subobjlen);

DeletePlayer[2].k := 0;

self.DeleteExpSmallPlayer2.Enabled := false;

end

else

inc(DeletePlayer[2].k);

end;

procedure TGameInterface.DeleteExpBigPlayerTimer(Sender: TObject);

begin

if DeletePlayer[4].k = 1 then

begin

GameScreen.Canvas.Rectangle(PlayerTank.DP[0].X, PlayerTank.DP[0].Y,

PlayerTank.DP[3].X + 1, PlayerTank.DP[3].Y + 1);

DeletePlayer[4].k := 0;

self.DeleteExpBigPlayer.Enabled := false;

end

else

inc(DeletePlayer[4].k);

end;

/// ////////

procedure TGameInterface.EnemyTank1SetDirectionTimer(Sender: TObject);

begin

EnemyTanks[1].direction := random(4);

end;

procedure TGameInterface.EnemyTank1MovementTimer(Sender: TObject);

begin

MoveEnemyTank(GameInterface.GameScreen, 1);

end;

procedure TGameInterface.Enemy1RespawnTimer(Sender: TObject);

begin

if EnemyTanks[1].isDestroyed and (lives[1] > 0) then

begin

inc(PlayerScore, 100);

dec(lives[1]);

if lives[1] > 0 then

begin

CreateEnemyTank(EnemyCoordsSpawn[1].X, EnemyCoordsSpawn[1].Y, 1);

GameInterface.EnemyTank1Movement.Enabled := true;

GameInterface.Enemy1Shoot.Enabled := true;

GameInterface.EnemyTank1SetDirection.Enabled := true;

EnemyTanks[1].isDestroyed := false;

self.Enemy1Respawn.Enabled := false;

end

end;

end;

procedure TGameInterface.Enemy1ShellMovementTimer(Sender: TObject);

begin

MoveEnemyShell1(GameInterface.GameScreen, 1);

end;

procedure TGameInterface.Enemy1ShootTimer(Sender: TObject);

begin

ShootEnemyTank1(1);

end;

procedure TGameInterface.DeleteExpSmallEnemy11Timer(Sender: TObject);

begin

if DeleteEnemy[1][1].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeleteEnemy[1][1].column \* subobjlen,

DeleteEnemy[1][1].line \* subobjlen, DeleteEnemy[1][1].column \* subobjlen +

subobjlen, DeleteEnemy[1][1].line \* subobjlen + subobjlen);

DeleteEnemy[1][1].k := 0;

self.DeleteExpSmallEnemy11.Enabled := false;

end

else

inc(DeleteEnemy[1][1].k);

end;

procedure TGameInterface.DeleteExpSmallEnemy12Timer(Sender: TObject);

begin

if DeleteEnemy[1][2].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeleteEnemy[1][2].column \* subobjlen,

DeleteEnemy[1][2].line \* subobjlen, DeleteEnemy[1][2].column \* subobjlen +

subobjlen, DeleteEnemy[1][2].line \* subobjlen + subobjlen);

DeleteEnemy[1][2].k := 0;

self.DeleteExpSmallEnemy12.Enabled := false;

end

else

inc(DeleteEnemy[1][2].k);

end;

procedure TGameInterface.DeleteExpBigEnemy1Timer(Sender: TObject);

begin

if DeleteEnemy[1][4].k = 1 then

begin

GameScreen.Canvas.Rectangle(EnemyTanks[1].DP[0].X, EnemyTanks[1].DP[0].Y,

EnemyTanks[1].DP[3].X + 1, EnemyTanks[1].DP[3].Y + 1);

DeleteEnemy[1][4].k := 0;

self.DeleteExpBigEnemy1.Enabled := false;

end

else

inc(DeleteEnemy[1][4].k);

end;

/// ///////////

procedure TGameInterface.EnemyTank2MovementTimer(Sender: TObject);

begin

MoveEnemyTank(GameInterface.GameScreen, 2);

end;

procedure TGameInterface.EnemyTank2SetDirectionTimer(Sender: TObject);

begin

EnemyTanks[2].direction := random(4);

end;

procedure TGameInterface.Enemy2RespawnTimer(Sender: TObject);

begin

if EnemyTanks[2].isDestroyed and (lives[2] > 0) then

begin

inc(PlayerScore, 100);

dec(lives[2]);

if lives[2] > 0 then

begin

CreateEnemyTank(EnemyCoordsSpawn[2].X, EnemyCoordsSpawn[2].Y, 2);

GameInterface.EnemyTank2Movement.Enabled := true;

GameInterface.Enemy2Shoot.Enabled := true;

GameInterface.EnemyTank2SetDirection.Enabled := true;

EnemyTanks[2].isDestroyed := false;

self.Enemy2Respawn.Enabled := false;

end

end;

end;

procedure TGameInterface.Enemy2ShellMovementTimer(Sender: TObject);

begin

MoveEnemyShell2(GameInterface.GameScreen, 2);

end;

procedure TGameInterface.Enemy2ShootTimer(Sender: TObject);

begin

ShootEnemyTank2(2);

end;

procedure TGameInterface.DeleteExpSmallEnemy21Timer(Sender: TObject);

begin

if DeleteEnemy[2][1].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeleteEnemy[2][1].column \* subobjlen,

DeleteEnemy[2][1].line \* subobjlen, DeleteEnemy[2][1].column \* subobjlen +

subobjlen, DeleteEnemy[2][1].line \* subobjlen + subobjlen);

DeleteEnemy[2][1].k := 0;

self.DeleteExpSmallEnemy21.Enabled := false;

end

else

inc(DeleteEnemy[2][1].k);

end;

procedure TGameInterface.DeleteExpSmallEnemy22Timer(Sender: TObject);

begin

if DeleteEnemy[2][2].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeleteEnemy[2][2].column \* subobjlen,

DeleteEnemy[2][2].line \* subobjlen, DeleteEnemy[2][2].column \* subobjlen +

subobjlen, DeleteEnemy[2][2].line \* subobjlen + subobjlen);

DeleteEnemy[2][2].k := 0;

self.DeleteExpSmallEnemy22.Enabled := false;

end

else

inc(DeleteEnemy[2][2].k);

end;

procedure TGameInterface.DeleteExpBigEnemy2Timer(Sender: TObject);

begin

if DeleteEnemy[2][4].k = 1 then

begin

GameScreen.Canvas.Rectangle(EnemyTanks[2].DP[0].X, EnemyTanks[2].DP[0].Y,

EnemyTanks[2].DP[3].X + 1, EnemyTanks[2].DP[3].Y + 1);

DeleteEnemy[2][4].k := 0;

self.DeleteExpBigEnemy2.Enabled := false;

end

else

inc(DeleteEnemy[2][4].k);

end;

/// /////////

procedure TGameInterface.EnemyTank3MovementTimer(Sender: TObject);

begin

MoveEnemyTank(GameInterface.GameScreen, 3);

end;

procedure TGameInterface.EnemyTank3SetDirectionTimer(Sender: TObject);

begin

EnemyTanks[3].direction := random(4);

end;

procedure TGameInterface.Enemy3RespawnTimer(Sender: TObject);

begin

if EnemyTanks[3].isDestroyed and (lives[3] > 0) then

begin

inc(PlayerScore, 100);

dec(lives[3]);

if lives[3] > 0 then

begin

CreateEnemyTank(EnemyCoordsSpawn[3].X, EnemyCoordsSpawn[3].Y, 3);

GameInterface.EnemyTank3Movement.Enabled := true;

GameInterface.Enemy3Shoot.Enabled := true;

GameInterface.EnemyTank3SetDirection.Enabled := true;

EnemyTanks[3].isDestroyed := false;

self.Enemy3Respawn.Enabled := false;

end

end;

end;

procedure TGameInterface.Enemy3ShellMovementTimer(Sender: TObject);

begin

MoveEnemyShell3(GameInterface.GameScreen, 3);

end;

procedure TGameInterface.Enemy3ShootTimer(Sender: TObject);

begin

ShootEnemyTank3(3);

end;

procedure TGameInterface.DeleteExpSmallEnemy31Timer(Sender: TObject);

begin

if DeleteEnemy[3][1].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeleteEnemy[3][1].column \* subobjlen,

DeleteEnemy[3][1].line \* subobjlen, DeleteEnemy[3][1].column \* subobjlen +

subobjlen, DeleteEnemy[3][1].line \* subobjlen + subobjlen);

DeleteEnemy[3][1].k := 0;

self.DeleteExpSmallEnemy31.Enabled := false;

end

else

inc(DeleteEnemy[3][1].k);

end;

procedure TGameInterface.DeleteExpSmallEnemy32Timer(Sender: TObject);

begin

if DeleteEnemy[3][2].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeleteEnemy[3][2].column \* subobjlen,

DeleteEnemy[3][2].line \* subobjlen, DeleteEnemy[3][2].column \* subobjlen +

subobjlen, DeleteEnemy[3][2].line \* subobjlen + subobjlen);

DeleteEnemy[3][2].k := 0;

self.DeleteExpSmallEnemy32.Enabled := false;

end

else

inc(DeleteEnemy[3][2].k);

end;

procedure TGameInterface.DeleteExpBigEnemy3Timer(Sender: TObject);

begin

if DeleteEnemy[3][4].k = 1 then

begin

GameScreen.Canvas.Rectangle(EnemyTanks[3].DP[0].X, EnemyTanks[3].DP[0].Y,

EnemyTanks[3].DP[3].X + 1, EnemyTanks[3].DP[3].Y + 1);

DeleteEnemy[3][4].k := 0;

self.DeleteExpBigEnemy3.Enabled := false;

end

else

inc(DeleteEnemy[3][4].k);

end;

/// ////////

procedure TGameInterface.EnemyTank4MovementTimer(Sender: TObject);

begin

MoveEnemyTank(GameInterface.GameScreen, 4);

end;

procedure TGameInterface.EnemyTank4SetDirectionTimer(Sender: TObject);

begin

EnemyTanks[4].direction := random(4);

end;

procedure TGameInterface.Enemy4RespawnTimer(Sender: TObject);

begin

if EnemyTanks[4].isDestroyed and (lives[4] > 0) then

begin

inc(PlayerScore, 100);

dec(lives[4]);

if lives[4] > 0 then

begin

CreateEnemyTank(EnemyCoordsSpawn[4].X, EnemyCoordsSpawn[4].Y, 4);

GameInterface.EnemyTank4Movement.Enabled := true;

GameInterface.Enemy4Shoot.Enabled := true;

GameInterface.EnemyTank4SetDirection.Enabled := true;

EnemyTanks[4].isDestroyed := false;

self.Enemy4Respawn.Enabled := false;

end

end;

end;

procedure TGameInterface.Enemy4ShellMovementTimer(Sender: TObject);

begin

MoveEnemyShell4(GameInterface.GameScreen, 4);

end;

procedure TGameInterface.Enemy4ShootTimer(Sender: TObject);

begin

ShootEnemyTank4(4);

end;

procedure TGameInterface.DeleteExpSmallEnemy41Timer(Sender: TObject);

begin

if DeleteEnemy[4][1].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeleteEnemy[4][1].column \* subobjlen,

DeleteEnemy[4][1].line \* subobjlen, DeleteEnemy[4][1].column \* subobjlen +

subobjlen, DeleteEnemy[4][1].line \* subobjlen + subobjlen);

DeleteEnemy[4][1].k := 0;

self.DeleteExpSmallEnemy41.Enabled := false;

end

else

inc(DeleteEnemy[4][1].k);

end;

procedure TGameInterface.DeleteExpSmallEnemy42Timer(Sender: TObject);

begin

if DeleteEnemy[4][2].k = 1 then

begin

GameScreen.Canvas.Rectangle(DeleteEnemy[4][2].column \* subobjlen,

DeleteEnemy[4][2].line \* subobjlen, DeleteEnemy[4][2].column \* subobjlen +

subobjlen, DeleteEnemy[4][2].line \* subobjlen + subobjlen);

DeleteEnemy[4][2].k := 0;

self.DeleteExpSmallEnemy42.Enabled := false;

end

else

inc(DeleteEnemy[4][2].k);

end;

procedure TGameInterface.DeleteExpBigEnemy4Timer(Sender: TObject);

begin

if DeleteEnemy[4][4].k = 1 then

begin

GameScreen.Canvas.Rectangle(EnemyTanks[4].DP[0].X, EnemyTanks[4].DP[0].Y,

EnemyTanks[4].DP[3].X + 1, EnemyTanks[4].DP[3].Y + 1);

DeleteEnemy[4][4].k := 0;

self.DeleteExpBigEnemy4.Enabled := false;

end

else

inc(DeleteEnemy[4][4].k);

end;

/// ///////////

end.

unit UGameMap;

interface

uses

Vcl.Graphics, Vcl.ExtCtrls;

const

MapSize = 1040;

ObjLen = 80;

SubObjLen = 40;

ObjCnt = 13;

SubObjCnt = 2;

StaticObjImgCnt = 7;

LineBase = 24;

ColumnBase = 12;

type

TCoords = record

X, Y: integer;

end;

TForestObjArr = array of TCoords;

TWaterObjArr = array of TCoords;

TSteelObjArr = array of TCoords;

TTankPxMap = array [0 .. MapSize - 1, 0 .. MapSize - 1] of integer;

TPxMap = array [0 .. MapSize - 1, 0 .. MapSize - 1] of integer;

TMapFileMtx = array [1 .. ObjCnt \* SubObjCnt] of string;

TStaticObjImgArr = array [0 .. StaticObjImgCnt] of TBitMap;

procedure DrawBackGround(Screen: TImage);

procedure StaticObjImgArrInit(var StaticObjImg: TStaticObjImgArr);

procedure LoadMapFromFile(Screen: TImage; path: string);

var

PxMap: TPxMap;

TankPxMap: TTankPxMap;

MapFileMtx: TMapFileMtx;

StaticObjImg: TStaticObjImgArr;

ForestObj: TForestObjArr;

WaterObj: TWaterObjArr;

SteelObj: TSteelObjArr;

implementation

uses

UGameInterface, UTTankType, UShellType, UEnemyTanks, UEnemyShells;

procedure DrawBackGround(Screen: TImage);

begin

Screen.Canvas.Rectangle(0, 0, MapSize, MapSize);

end;

procedure StaticObjImgArrInit(var StaticObjImg: TStaticObjImgArr);

var

i: integer;

begin

for i := 0 to StaticObjImgCnt do

StaticObjImg[i] := TBitMap.Create;

StaticObjImg[0] := nil;

StaticObjImg[1].LoadFromFile('..\icons\StaticObjects\brick.bmp');

StaticObjImg[2].LoadFromFile('..\icons\StaticObjects\die.bmp');

StaticObjImg[3].LoadFromFile('..\icons\StaticObjects\eagle.bmp');

StaticObjImg[4].LoadFromFile('..\icons\StaticObjects\forest.bmp');

StaticObjImg[5].LoadFromFile('..\icons\StaticObjects\ice.bmp');

StaticObjImg[6].LoadFromFile('..\icons\StaticObjects\steel.bmp');

StaticObjImg[7].LoadFromFile('..\icons\StaticObjects\water.bmp');

SetLength(ForestObj, 0);

SetLength(WaterObj, 0);

SetLength(SteelObj, 0);

end;

procedure LoadMapFromFile(Screen: TImage; path: string);

var

MapFile: TextFile;

i, j, id, k, l, posX, posY: integer;

MapImgToDraw: TBitMap;

begin

posX := -1;

posY := -1;

AssignFile(MapFile, path);

Reset(MapFile);

for i := 1 to ObjCnt \* SubObjCnt do

readln(MapFile, MapFileMtx[i]);

CloseFile(MapFile);

id := 0;

for i := 1 to ObjCnt \* SubObjCnt do

for j := 1 to ObjCnt \* SubObjCnt do

begin

case MapFileMtx[j][i] of

'z':

id := 0;

'b':

id := 1;

'd':

id := 2;

'e':

begin

id := 3;

posX := i;

posY := j;

end;

'f':

id := 4;

's':

id := 6;

'w':

id := 7;

end;

MapImgToDraw := StaticObjImg[id];

Screen.Canvas.Draw((i - 1) \* SubObjLen, (j - 1) \* SubObjLen,

MapImgToDraw);

for k := (j - 1) \* SubObjLen to (j - 1) \* SubObjLen + SubObjLen - 1 do

for l := (i - 1) \* SubObjLen to (i - 1) \* SubObjLen + SubObjLen - 1 do

PxMap[k][l] := id;

case MapFileMtx[j][i] of

'f':

begin

SetLength(ForestObj, length(ForestObj) + 1);

ForestObj[length(ForestObj) - 1].X := (i - 1) \* SubObjLen;

ForestObj[length(ForestObj) - 1].Y := (j - 1) \* SubObjLen;

end;

'w':

begin

SetLength(WaterObj, length(WaterObj) + 1);

WaterObj[length(WaterObj) - 1].X := (i - 1) \* SubObjLen;

WaterObj[length(WaterObj) - 1].Y := (j - 1) \* SubObjLen;

end;

's':

begin

SetLength(SteelObj, length(SteelObj) + 1);

SteelObj[length(SteelObj) - 1].X := (i - 1) \* SubObjLen;

SteelObj[length(SteelObj) - 1].Y := (j - 1) \* SubObjLen;

end;

end;

end;

for i := (posX - 1) \* SubObjLen to (posX - 1) \* SubObjLen + ObjLen - 1 do

for j := (posY - 1) \* SubObjLen to (posY - 1) \* SubObjLen + ObjLen - 1 do

PxMap[j][i] := 3;

end;

end.

unit UTTankType;

interface

uses

Vcl.Graphics, Vcl.ExtCtrls, System.Classes;

const

DirImgCnt = 4;

DimPntCnt = 4;

TankSize = 80;

TankSpeed = 10;

type

TCoords = record

X, Y: integer;

end;

TDimPntArr = array [0 .. DirImgCnt - 1] of TCoords;

TDirImgArr = array [0 .. DirImgCnt - 1] of TBitMap;

TTankr = record

direction: integer;

DP: TDimPntArr;

isShotMade, isDestroyed: boolean;

end;

procedure CreatePlayerTank(X, Y: integer);

procedure MovePlayer(key: integer; screen: TImage);

procedure ShootPlayer(key: integer);

procedure PlayerTankImgArrInit();

var

PlayerTankImg: TDirImgArr;

PlayerTank: TTankr;

implementation

uses

UGameInterface, UGameMap, UShellType, UEnemyTanks, UEnemyShells;

procedure CreatePlayerTank(X, Y: integer);

var

i, j: integer;

begin

PlayerTank.direction := 0;

PlayerTank.isDestroyed := false;

PlayerTank.isShotMade := false;

PlayerTank.DP[0].X := X;

PlayerTank.DP[0].Y := Y;

PlayerTank.DP[1].X := X + TankSize - 1;

PlayerTank.DP[1].Y := Y;

PlayerTank.DP[2].X := X;

PlayerTank.DP[2].Y := Y + TankSize - 1;

PlayerTank.DP[3].X := X + TankSize - 1;

PlayerTank.DP[3].Y := Y + TankSize - 1;

for i := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

for j := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

TankPxMap[i][j] := -10;

end;

procedure MovePlayer(key: integer; screen: TImage);

function CheckBorder(direction: integer): boolean;

var

res: boolean;

begin

res := false;

case direction of

0:

res := ((PlayerTank.DP[0].Y > 0) and (PlayerTank.DP[1].Y > 0));

1:

res := ((PlayerTank.DP[1].X < MapSize - 1) and (PlayerTank.DP[3].X < MapSize - 1));

2:

res := ((PlayerTank.DP[2].Y < MapSize - 1) and (PlayerTank.DP[3].Y < MapSize - 1));

3:

res := ((PlayerTank.DP[0].X > 0) and (PlayerTank.DP[2].X > 0));

end;

result := res;

end;

function CheckObjId(Id: integer): boolean;

var

res: boolean;

begin

case Id of

0:

res := true;

4 .. 5:

res := true;

else

res := false;

end;

result := res;

end;

function CheckEnemyTank(Id: integer): boolean;

var

res: boolean;

begin

res := false;

case Id of

- 1, -2, -3, -4:

res := false;

0:

res := true;

end;

result := res;

end;

var

isAbleToMove1, isAbleToMove2, isAbleToMove3, isKeyPressed: boolean;

i, k, l: integer;

PntToMov1, PntToMov2, PntToMov3: TCoords;

begin

isAbleToMove3 := false;

screen.Canvas.Rectangle(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, PlayerTank.DP[3].X + 1,

PlayerTank.DP[3].Y + 1);

for k := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

for l := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

TankPxMap[k][l] := 0;

isKeyPressed := true;

case key of

38, 87:

PlayerTank.direction := 0;

40, 83:

PlayerTank.direction := 2;

37, 65:

PlayerTank.direction := 3;

39, 68:

PlayerTank.direction := 1;

-100, 13, 32:

isKeyPressed := false;

end;

if isKeyPressed then

begin

isAbleToMove1 := CheckBorder(PlayerTank.direction);

isAbleToMove2 := CheckBorder(PlayerTank.direction);

if isAbleToMove1 and isAbleToMove2 then

begin

case PlayerTank.direction of

0:

begin

PntToMov1 := PlayerTank.DP[0];

PntToMov2 := PlayerTank.DP[1];

dec(PntToMov1.Y);

dec(PntToMov2.Y);

PntToMov3.X := PntToMov1.X + TankSize div 2 - 1;

PntToMov3.Y := PntToMov1.Y;

end;

1:

begin

PntToMov1 := PlayerTank.DP[1];

PntToMov2 := PlayerTank.DP[3];

inc(PntToMov1.X);

inc(PntToMov2.X);

PntToMov3.X := PntToMov1.X;

PntToMov3.Y := PntToMov1.Y + TankSize div 2 - 1;

end;

2:

begin

PntToMov1 := PlayerTank.DP[2];

PntToMov2 := PlayerTank.DP[3];

inc(PntToMov1.Y);

inc(PntToMov2.Y);

PntToMov3.X := PntToMov1.X + TankSize div 2 - 1;

PntToMov3.Y := PntToMov1.Y;

end;

3:

begin

PntToMov1 := PlayerTank.DP[0];

PntToMov2 := PlayerTank.DP[2];

dec(PntToMov1.X);

dec(PntToMov2.X);

PntToMov3.X := PntToMov1.X;

PntToMov3.Y := PntToMov1.Y + TankSize div 2 - 1;

end;

end;

isAbleToMove1 := CheckObjId(PxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckObjId(PxMap[PntToMov2.Y][PntToMov2.X]);

isAbleToMove3 := CheckObjId(PxMap[PntToMov3.Y][PntToMov3.X]);

end;

if isAbleToMove1 and isAbleToMove2 and isAbletoMove3 then

begin

isAbleToMove1 := CheckEnemyTank(TankPxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckEnemyTank(TankPxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

for i := 0 to DimPntCnt - 1 do

case PlayerTank.direction of

0:

dec(PlayerTank.DP[i].Y, TankSpeed);

1:

inc(PlayerTank.DP[i].X, TankSpeed);

2:

inc(PlayerTank.DP[i].Y, TankSpeed);

3:

dec(PlayerTank.DP[i].X, TankSpeed);

end;

end;

end;

for k := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

for l := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

TankPxMap[k][l] := -10;

if not PlayerTank.isShotMade then

begin

for i := 0 to length(WaterObj) - 1 do

screen.Canvas.Draw(WaterObj[i].X, WaterObj[i].Y, StaticObjImg[7]);

end;

screen.Canvas.Draw(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, PlayerTankImg[PlayerTank.direction]);

for i := 0 to length(ForestObj) - 1 do

screen.Canvas.Draw(ForestObj[i].X, ForestObj[i].Y, StaticObjImg[4]);

for i := 0 to length(SteelObj) - 1 do

screen.Canvas.Draw(SteelObj[i].X, SteelObj[i].Y, StaticObjImg[6]);

end;

procedure ShootPlayer(key: integer);

begin

if not playerTank.isShotMade then

begin

case key of

32:

playerTank.isShotMade := true;

else

playerTank.isShotMade := false;

end;

if playerTank.isShotMade then

begin

CreatePlayerShell(playerTank.direction, playerTank.DP[0].X, playerTank.DP[0].Y);

GameInterface.PlayerShellMovement.Enabled := true;

end;

end

end;

procedure PlayerTankImgArrInit();

var

i: integer;

begin

for i := 0 to DirImgCnt - 1 do

PlayerTankImg[i] := TBitMap.Create;

PlayerTankImg[0].LoadFromFile('..\icons\PlayerTank\tUp.bmp');

PlayerTankImg[1].LoadFromFile('..\icons\PlayerTank\tRight.bmp');

PlayerTankImg[2].LoadFromFile('..\icons\PlayerTank\tDown.bmp');

PlayerTankImg[3].LoadFromFile('..\icons\PlayerTank\tLeft.bmp');

end;

end.

unit UShellType;

interface

uses

Vcl.Graphics, Vcl.ExtCtrls, System.Classes, System.SysUtils;

const

DimPntCnt = 4;

ShellSize = 10;

ShellSpeed = 10;

type

TCoords = record

X, Y: integer;

end;

TDimPntArr = array [0 .. DimPntCnt - 1] of TCoords;

TShellr = record

direction: integer;

DP: TDimPntArr;

img: TBitMap;

end;

procedure ExpImgInit();

procedure CreatePlayerShell(direction, x, y: integer);

procedure MovePlayerShell(screen: TImage);

var

PlayerShell: TShellr;

ExpBig, ExpSmall: TBitMap;

implementation

uses UGameInterface, UGameMap, UTTankType, UEnemyTanks, UEnemyShells;

procedure CreatePlayerShell(direction, x, y: integer);

begin

PlayerShell.direction := direction;

PlayerShell.img := TBitMap.Create;

PlayerShell.img.LoadFromFile('..\icons\Shell\s.bmp');

case direction of

0:

begin

PlayerShell.DP[0].X := X + TankSize div 2 - ShellSize div 2;

PlayerShell.DP[0].Y := Y - ShellSize;

end;

1:

begin

PlayerShell.DP[0].X := X + TankSize;

PlayerShell.DP[0].Y := Y + TankSize div 2 - ShellSize div 2;

end;

2:

begin

PlayerShell.DP[0].X := X + TankSize div 2 - ShellSize div 2;

PlayerShell.DP[0].Y := Y + TankSize;

end;

3:

begin

PlayerShell.DP[0].X := X - ShellSize;

PlayerShell.DP[0].Y := Y - ShellSize div 2 + TankSize div 2;

end;

end;

PlayerShell.DP[1].X := PlayerShell.DP[0].X + ShellSize - 1;

PlayerShell.DP[1].Y := PlayerShell.DP[0].Y;

PlayerShell.DP[2].X := PlayerShell.DP[0].X;

PlayerShell.DP[2].Y := PlayerShell.DP[0].Y + ShellSize - 1;

PlayerShell.DP[3].X := PlayerShell.DP[0].X + ShellSize - 1;

PlayerShell.DP[3].Y := PlayerShell.DP[0].Y + ShellSize - 1;

end;

procedure MovePlayerShell(screen: TImage);

function CheckBorder(direction: integer): boolean;

var

res: boolean;

begin

res := false;

case direction of

0:

res := ((PlayerShell.DP[0].Y > 0) and (PlayerShell.DP[1].Y > 0));

1:

res := ((PlayerShell.DP[1].X < MapSize - 1) and (PlayerShell.DP[3].X < MapSize - 1));

2:

res := ((PlayerShell.DP[2].Y < MapSize - 1) and (PlayerShell.DP[3].Y < MapSize - 1));

3:

res := ((PlayerShell.DP[0].X > 0) and (PlayerShell.DP[2].X > 0));

end;

result := res;

end;

function CheckObj(id: integer): boolean;

var

res: boolean;

begin

res := false;

case id of

0, 4, 5, 7:

res := true;

1, 2, 3, 6:

res := false;

end;

result := res;

end;

function CheckEnemyTank(id: integer): boolean;

var

res: boolean;

begin

res := false;

case id of

- 1, -2, -3, -4:

res := false;

0, -10:

res := true;

end;

result := res;

end;

var

i, line, column, k, l: integer;

isAbleToMove1, isAbleToMove2: boolean;

PntToMov1, PntToMov2: TCoords;

begin

screen.Canvas.Rectangle(PlayerShell.DP[0].X, PlayerShell.DP[0].Y, PlayerShell.DP[3].X + 1,

PlayerShell.DP[3].Y + 1);

isAbleToMove1 := CheckBorder(PlayerShell.direction);

isAbleToMove2 := CheckBorder(PlayerShell.direction);

case PlayerShell.direction of

2:

begin

PntToMov1 := PlayerShell.DP[0];

PntToMov2 := PlayerShell.DP[1];

dec(PntToMov1.Y);

dec(PntToMov2.Y);

end;

3:

begin

PntToMov1 := PlayerShell.DP[1];

PntToMov2 := PlayerShell.DP[3];

inc(PntToMov1.X);

inc(PntToMov2.X);

end;

0:

begin

PntToMov1 := PlayerShell.DP[2];

PntToMov2 := PlayerShell.DP[3];

inc(PntToMov1.Y);

inc(PntToMov2.Y);

end;

1:

begin

PntToMov1 := PlayerShell.DP[0];

PntToMov2 := PlayerShell.DP[2];

dec(PntToMov1.X);

dec(PntToMov2.X);

end;

end;

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckObj(PxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckObj(PxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckEnemyTank(TankPxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckEnemyTank(TankPxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

for i := 0 to DimPntCnt - 1 do

case PlayerShell.direction of

0:

dec(PlayerShell.DP[i].Y, ShellSpeed);

1:

inc(PlayerShell.DP[i].X, ShellSpeed);

2:

inc(PlayerShell.DP[i].Y, ShellSpeed);

3:

dec(PlayerShell.DP[i].X, ShellSpeed);

end;

for i := 0 to length(waterObj) - 1 do

screen.Canvas.Draw(waterObj[i].X, waterObj[i].Y, StaticObjImg[7]);

screen.Canvas.Draw(PlayerShell.DP[0].X, PlayerShell.DP[0].Y, PlayerShell.img);

for i := 0 to length(ForestObj) - 1 do

screen.Canvas.Draw(ForestObj[i].X, ForestObj[i].Y, StaticObjImg[4]);

end

else

begin

if not isAbleToMove1 then

begin

case TankPxMap[PntToMov1.Y][PntToMov1.X] of

- 1:

begin

gameInterface.EnemyTank1Movement.Enabled := false;

gameInterface.Enemy1Shoot.Enabled := false;

gameInterface.EnemyTank1SetDirection.Enabled := false;

Enemytanks[1].isDestroyed := true;

gameInterface.Enemy1Respawn.Enabled := true;

screen.Canvas.Draw(Enemytanks[1].DP[0].X,

Enemytanks[1].DP[0].Y, ExpBig);

gameInterface.DeleteExpBigEnemy1.Enabled := true;

for k := Enemytanks[1].DP[0].X to Enemytanks[1].DP[3].X do

for l := Enemytanks[1].DP[0].Y to Enemytanks[1].DP[3].Y do

TankPxMap[l][k] := 0;

end;

-2:

begin

gameInterface.EnemyTank2Movement.Enabled := false;

gameInterface.Enemy2Shoot.Enabled := false;

gameInterface.EnemyTank2SetDirection.Enabled := false;

Enemytanks[2].isDestroyed := true;

gameInterface.Enemy2Respawn.Enabled := true;

screen.Canvas.Draw(Enemytanks[2].DP[0].X,

Enemytanks[2].DP[0].Y, ExpBig);

gameInterface.DeleteExpBigEnemy2.Enabled := true;

for k := Enemytanks[2].DP[0].X to Enemytanks[2].DP[3].X do

for l := Enemytanks[2].DP[0].Y to Enemytanks[2].DP[3].Y do

TankPxMap[l][k] := 0;

end;

-3:

begin

gameInterface.EnemyTank3Movement.Enabled := false;

gameInterface.Enemy3Shoot.Enabled := false;

gameInterface.EnemyTank3SetDirection.Enabled := false;

Enemytanks[3].isDestroyed := true;

gameInterface.Enemy3Respawn.Enabled := true;

screen.Canvas.Draw(Enemytanks[3].DP[0].X,

Enemytanks[3].DP[0].Y, ExpBig);

gameInterface.DeleteExpBigEnemy3.Enabled := true;

for k := Enemytanks[3].DP[0].X to Enemytanks[2].DP[3].X do

for l := Enemytanks[3].DP[0].Y to Enemytanks[2].DP[3].Y do

TankPxMap[l][k] := 0;

end;

-4:

begin

gameInterface.EnemyTank4Movement.Enabled := false;

gameInterface.Enemy4Shoot.Enabled := false;

gameInterface.EnemyTank4SetDirection.Enabled := false;

gameInterface.Enemy4Respawn.Enabled := true;

Enemytanks[4].isDestroyed := true;

screen.Canvas.Draw(Enemytanks[4].DP[0].X,

Enemytanks[4].DP[0].Y, ExpBig);

gameInterface.DeleteExpBigEnemy4.Enabled := true;

for k := Enemytanks[4].DP[0].X to Enemytanks[4].DP[3].X do

for l := Enemytanks[4].DP[0].Y to Enemytanks[4].DP[3].Y do

TankPxMap[l][k] := 0;

end;

end;

end;

if not isAbleToMove2 then

begin

case TankPxMap[PntToMov2.Y][PntToMov2.X] of

- 1:

begin

gameInterface.EnemyTank1Movement.Enabled := false;

gameInterface.Enemy1Shoot.Enabled := false;

gameInterface.EnemyTank1SetDirection.Enabled := false;

Enemytanks[1].isDestroyed := true;

gameInterface.Enemy1Respawn.Enabled := true;

screen.Canvas.Draw(Enemytanks[1].DP[0].X,

Enemytanks[1].DP[0].Y, ExpBig);

gameInterface.DeleteExpBigEnemy1.Enabled := true;

for k := Enemytanks[1].DP[0].X to Enemytanks[1].DP[3].X do

for l := Enemytanks[1].DP[0].Y to Enemytanks[1].DP[3].Y do

TankPxMap[l][k] := 0;

end;

-2:

begin

gameInterface.EnemyTank2Movement.Enabled := false;

gameInterface.Enemy2Shoot.Enabled := false;

gameInterface.EnemyTank2SetDirection.Enabled := false;

Enemytanks[2].isDestroyed := true;

gameInterface.Enemy2Respawn.Enabled := true;

screen.Canvas.Draw(Enemytanks[2].DP[0].X,

Enemytanks[2].DP[0].Y, ExpBig);

gameInterface.DeleteExpBigEnemy2.Enabled := true;

for k := Enemytanks[2].DP[0].X to Enemytanks[2].DP[3].X do

for l := Enemytanks[2].DP[0].Y to Enemytanks[2].DP[3].Y do

TankPxMap[l][k] := 0;

end;

-3:

begin

gameInterface.EnemyTank3Movement.Enabled := false;

gameInterface.Enemy3Shoot.Enabled := false;

gameInterface.EnemyTank3SetDirection.Enabled := false;

Enemytanks[3].isDestroyed := true;

gameInterface.Enemy3Respawn.Enabled := true;

screen.Canvas.Draw(Enemytanks[3].DP[0].X,

Enemytanks[3].DP[0].Y, ExpBig);

gameInterface.DeleteExpBigEnemy3.Enabled := true;

for k := Enemytanks[3].DP[0].X to Enemytanks[3].DP[3].X do

for l := Enemytanks[3].DP[0].Y to Enemytanks[3].DP[3].Y do

TankPxMap[l][k] := 0;

end;

-4:

begin

gameInterface.EnemyTank4Movement.Enabled := false;

gameInterface.Enemy4Shoot.Enabled := false;

gameInterface.EnemyTank4SetDirection.Enabled := false;

Enemytanks[4].isDestroyed := true;

gameInterface.Enemy4Respawn.Enabled := true;

screen.Canvas.Draw(Enemytanks[4].DP[0].X,

Enemytanks[4].DP[0].Y, ExpBig);

gameInterface.DeleteExpBigEnemy4.Enabled := true;

for k := Enemytanks[4].DP[0].X to Enemytanks[4].DP[3].X do

for l := Enemytanks[4].DP[0].Y to Enemytanks[4].DP[3].Y do

TankPxMap[l][k] := 0;

end;

end;

end;

gameInterface.PlayerShellMovement.Enabled := false;

PlayerTank.isShotMade := false;

end;

end

else

begin

if not isAbleToMove1 then

begin

line := PntToMov1.Y div subobjlen;

column := PntToMov1.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deletePlayer[1].line := line;

deletePlayer[1].column := column;

gameInterface.DeleteExpSmallPlayer1.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

gameInterface.DeleteExpBigBase.Enabled := true;

end;

gameInterface.PlayerShellMovement.Enabled := false;

PlayerTank.isShotMade := false;

end;

if not isAbleToMove2 then

begin

line := PntToMov2.Y div subobjlen;

column := PntToMov2.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deletePlayer[2].line := line;

deletePlayer[2].column := column;

gameInterface.DeleteExpSmallPlayer2.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

gameInterface.DeleteExpBigBase.Enabled := true;

end;

gameInterface.PlayerShellMovement.Enabled := false;

PlayerTank.isShotMade := false;

end;

end;

end

else

begin

gameInterface.PlayerShellMovement.Enabled := false;

PlayerTank.isShotMade := false;

end;

end;

procedure ExpImgInit();

begin

ExpBig := TBitMap.Create;

ExpBig.LoadFromFile('..\icons\Shell\e80.bmp');

ExpSmall := TBitMap.Create;

ExpSmall.LoadFromFile('..\icons\Shell\e40.bmp');

end;

end.

unit UEnemyShells;

interface

uses

Vcl.Graphics, Vcl.ExtCtrls, System.Classes, System.SysUtils;

const

DimPntCnt = 4;

ShellSize = 10;

ShellSpeed = 10;

type

TCoords = record

X, Y: integer;

end;

TDimPntArr = array [0 .. DimPntCnt - 1] of TCoords;

TEnemyShellr = record

direction: integer;

DP: TDimPntArr;

img: TBitMap;

end;

TEnemyShells = array [1 .. 4] of TEnemyShellr;

procedure CreateEnemyShell(direction, X, Y, num: integer);

procedure MoveEnemyShell1(screen: TImage; num: integer);

procedure MoveEnemyShell2(screen: TImage; num: integer);

procedure MoveEnemyShell3(screen: TImage; num: integer);

procedure MoveEnemyShell4(screen: TImage; num: integer);

var

EnemyShells: TEnemyShells;

Sender: Tobject;

implementation

uses UGameInterface, UGameMap, UTTankType, UShellTYpe, UEnemyTanks;

procedure CreateEnemyShell(direction, X, Y, num: integer);

begin

EnemyShells[num].direction := direction;

EnemyShells[num].img := TBitMap.Create;

EnemyShells[num].img.LoadFromFile('..\icons\Shell\s.bmp');

case direction of

0:

begin

EnemyShells[num].DP[0].X := X + TankSize div 2 - ShellSize div 2;

EnemyShells[num].DP[0].Y := Y - ShellSize;

end;

1:

begin

EnemyShells[num].DP[0].X := X + TankSize;

EnemyShells[num].DP[0].Y := Y + TankSize div 2 - ShellSize div 2;

end;

2:

begin

EnemyShells[num].DP[0].X := X + TankSize div 2 - ShellSize div 2;

EnemyShells[num].DP[0].Y := Y + TankSize;

end;

3:

begin

EnemyShells[num].DP[0].X := X - ShellSize;

EnemyShells[num].DP[0].Y := Y - ShellSize div 2 + TankSize div 2;

end;

end;

EnemyShells[num].DP[1].X := EnemyShells[num].DP[0].X + ShellSize - 1;

EnemyShells[num].DP[1].Y := EnemyShells[num].DP[0].Y;

EnemyShells[num].DP[2].X := EnemyShells[num].DP[0].X;

EnemyShells[num].DP[2].Y := EnemyShells[num].DP[0].Y + ShellSize - 1;

EnemyShells[num].DP[3].X := EnemyShells[num].DP[0].X + ShellSize - 1;

EnemyShells[num].DP[3].Y := EnemyShells[num].DP[0].Y + ShellSize - 1;

end;

function CheckBorder(direction, num: integer): boolean;

var

res: boolean;

begin

res := false;

case direction of

0:

res := ((EnemyShells[num].DP[0].Y > 0) and

(EnemyShells[num].DP[1].Y > 0));

1:

res := ((EnemyShells[num].DP[1].X < MapSize - 1) and

(EnemyShells[num].DP[3].X < MapSize - 1));

2:

res := ((EnemyShells[num].DP[2].Y < MapSize - 1) and

(EnemyShells[num].DP[3].Y < MapSize - 1));

3:

res := ((EnemyShells[num].DP[0].X > 0) and

(EnemyShells[num].DP[2].X > 0));

end;

result := res;

end;

function CheckObj(id: integer): boolean;

var

res: boolean;

begin

res := false;

case id of

0, 4, 5, 7:

res := true;

1, 2, 3, 6:

res := false;

end;

result := res;

end;

function CheckEnemyTank(id: integer): boolean;

var

res: boolean;

begin

res := false;

case id of

- 10:

res := false;

0, -1, -2, -3, -4:

res := true;

end;

result := res;

end;

procedure SetPntToMov(var PntToMov1, PntToMov2: TCoords; dir, num: integer);

begin

case dir of

0:

begin

PntToMov1 := EnemyShells[num].DP[0];

PntToMov2 := EnemyShells[num].DP[1];

dec(PntToMov1.Y);

dec(PntToMov2.Y);

end;

1:

begin

PntToMov1 := EnemyShells[num].DP[1];

PntToMov2 := EnemyShells[num].DP[3];

inc(PntToMov1.X);

inc(PntToMov2.X);

end;

2:

begin

PntToMov1 := EnemyShells[num].DP[2];

PntToMov2 := EnemyShells[num].DP[3];

inc(PntToMov1.Y);

inc(PntToMov2.Y);

end;

3:

begin

PntToMov1 := EnemyShells[num].DP[0];

PntToMov2 := EnemyShells[num].DP[2];

dec(PntToMov1.X);

dec(PntToMov2.X);

end;

end;

end;

procedure MoveEnemyShell1(screen: TImage; num: integer);

var

i, line, column, k, l: integer;

isAbleToMove1, isAbleToMove2: boolean;

PntToMov1, PntToMov2: TCoords;

begin

screen.Canvas.Rectangle(EnemyShells[num].DP[0].X, EnemyShells[num].DP[0].Y,

EnemyShells[num].DP[3].X + 1, EnemyShells[num].DP[3].Y + 1);

isAbleToMove1 := CheckBorder(EnemyShells[num].direction, num);

isAbleToMove2 := CheckBorder(EnemyShells[num].direction, num);

SetPntToMov(PntToMov1, PntToMov2, EnemyShells[num].direction, num);

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckObj(PxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckObj(PxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckEnemyTank(TankPxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckEnemyTank(TankPxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

for i := 0 to DimPntCnt - 1 do

case EnemyShells[num].direction of

0:

dec(EnemyShells[num].DP[i].Y, ShellSpeed);

1:

inc(EnemyShells[num].DP[i].X, ShellSpeed);

2:

inc(EnemyShells[num].DP[i].Y, ShellSpeed);

3:

dec(EnemyShells[num].DP[i].X, ShellSpeed);

end;

for i := 0 to length(waterObj) - 1 do

screen.Canvas.Draw(waterObj[i].X, waterObj[i].Y, StaticObjImg[7]);

screen.Canvas.Draw(EnemyShells[num].DP[0].X, EnemyShells[num].DP[0].Y,

EnemyShells[num].img);

for i := 0 to length(ForestObj) - 1 do

screen.Canvas.Draw(ForestObj[i].X, ForestObj[i].Y, StaticObjImg[4]);

end

else

begin

if not isAbleToMove1 then

begin

if TankPxMap[PntToMov1.Y][PntToMov1.X] = -10 then

begin

GameInterface.Enemy1ShellMovement.Enabled := false;

GameInterface.PlayerTankMovement.Enabled := false;

PlayerTank.isDestroyed := true;

GameInterface.PlayerRespawn.Enabled := true;

screen.Canvas.Draw(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, ExpBig);

GameInterface.DeleteExpBigPlayer.Enabled := true;

for k := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

for l := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

TankPxMap[l][k] := 0;

EnemyTanks[num].isShotMade := false;

end;

end;

if not isAbleToMove2 then

begin

if TankPxMap[PntToMov2.Y][PntToMov2.X] = -10 then

begin

GameInterface.Enemy1ShellMovement.Enabled := false;

GameInterface.PlayerTankMovement.Enabled := false;

PlayerTank.isDestroyed := true;

GameInterface.PlayerRespawn.Enabled := true;

screen.Canvas.Draw(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, ExpBig);

GameInterface.DeleteExpBigPlayer.Enabled := true;

for k := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

for l := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

TankPxMap[l][k] := 0;

EnemyTanks[num].isShotMade := false;

end;

end;

end;

end

else

begin

if not isAbleToMove1 then

begin

line := PntToMov1.Y div subobjlen;

column := PntToMov1.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deleteEnemy[num][1].line := line;

deleteEnemy[num][1].column := column;

GameInterface.DeleteExpSmallEnemy11.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

GameInterface.DeleteExpBigBase.Enabled := true;

end;

GameInterface.Enemy1ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

if not isAbleToMove2 then

begin

line := PntToMov2.Y div subobjlen;

column := PntToMov2.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deleteEnemy[num][2].line := line;

deleteEnemy[num][2].column := column;

GameInterface.DeleteExpSmallEnemy12.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

GameInterface.DeleteExpBigBase.Enabled := true;

end;

GameInterface.Enemy1ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

end;

end

else

begin

GameInterface.Enemy1ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

end;

procedure MoveEnemyShell2(screen: TImage; num: integer);

var

i, line, column, k, l: integer;

isAbleToMove1, isAbleToMove2: boolean;

PntToMov1, PntToMov2: TCoords;

begin

screen.Canvas.Rectangle(EnemyShells[num].DP[0].X, EnemyShells[num].DP[0].Y,

EnemyShells[num].DP[3].X + 1, EnemyShells[num].DP[3].Y + 1);

isAbleToMove1 := CheckBorder(EnemyShells[num].direction, num);

isAbleToMove2 := CheckBorder(EnemyShells[num].direction, num);

SetPntToMov(PntToMov1, PntToMov2, EnemyShells[num].direction, num);

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckObj(PxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckObj(PxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckEnemyTank(TankPxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckEnemyTank(TankPxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

for i := 0 to DimPntCnt - 1 do

case EnemyShells[num].direction of

0:

dec(EnemyShells[num].DP[i].Y, ShellSpeed);

1:

inc(EnemyShells[num].DP[i].X, ShellSpeed);

2:

inc(EnemyShells[num].DP[i].Y, ShellSpeed);

3:

dec(EnemyShells[num].DP[i].X, ShellSpeed);

end;

for i := 0 to length(waterObj) - 1 do

screen.Canvas.Draw(waterObj[i].X, waterObj[i].Y, StaticObjImg[7]);

screen.Canvas.Draw(EnemyShells[num].DP[0].X, EnemyShells[num].DP[0].Y,

EnemyShells[num].img);

for i := 0 to length(ForestObj) - 1 do

screen.Canvas.Draw(ForestObj[i].X, ForestObj[i].Y, StaticObjImg[4]);

end

else

begin

if not isAbleToMove1 then

begin

if TankPxMap[PntToMov1.Y][PntToMov1.X] = -10 then

begin

GameInterface.Enemy2ShellMovement.Enabled := false;

GameInterface.PlayerTankMovement.Enabled := false;

PlayerTank.isDestroyed := true;

GameInterface.PlayerRespawn.Enabled := true;

screen.Canvas.Draw(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, ExpBig);

GameInterface.DeleteExpBigPlayer.Enabled := true;

for k := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

for l := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

TankPxMap[l][k] := 0;

EnemyTanks[num].isShotMade := false;

end;

end;

if not isAbleToMove2 then

begin

if TankPxMap[PntToMov2.Y][PntToMov2.X] = -10 then

begin

GameInterface.Enemy2ShellMovement.Enabled := false;

GameInterface.PlayerTankMovement.Enabled := false;

PlayerTank.isDestroyed := true;

GameInterface.PlayerRespawn.Enabled := true;

screen.Canvas.Draw(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, ExpBig);

GameInterface.DeleteExpBigPlayer.Enabled := true;

for k := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

for l := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

TankPxMap[l][k] := 0;

EnemyTanks[num].isShotMade := false;

end;

end;

end;

end

else

begin

if not isAbleToMove1 then

begin

line := PntToMov1.Y div subobjlen;

column := PntToMov1.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deleteEnemy[num][1].line := line;

deleteEnemy[num][1].column := column;

GameInterface.DeleteExpSmallEnemy21.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

GameInterface.DeleteExpBigBase.Enabled := true;

end;

GameInterface.Enemy2ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

if not isAbleToMove2 then

begin

line := PntToMov2.Y div subobjlen;

column := PntToMov2.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deleteEnemy[num][2].line := line;

deleteEnemy[num][2].column := column;

GameInterface.DeleteExpSmallEnemy22.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

GameInterface.DeleteExpBigBase.Enabled := true;

end;

GameInterface.Enemy2ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

end;

end

else

begin

GameInterface.Enemy2ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

end;

procedure MoveEnemyShell3(screen: TImage; num: integer);

var

i, line, column, k, l: integer;

isAbleToMove1, isAbleToMove2: boolean;

PntToMov1, PntToMov2: TCoords;

begin

screen.Canvas.Rectangle(EnemyShells[num].DP[0].X, EnemyShells[num].DP[0].Y,

EnemyShells[num].DP[3].X + 1, EnemyShells[num].DP[3].Y + 1);

isAbleToMove1 := CheckBorder(EnemyShells[num].direction, num);

isAbleToMove2 := CheckBorder(EnemyShells[num].direction, num);

SetPntToMov(PntToMov1, PntToMov2, EnemyShells[num].direction, num);

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckObj(PxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckObj(PxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckEnemyTank(TankPxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckEnemyTank(TankPxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

for i := 0 to DimPntCnt - 1 do

case EnemyShells[num].direction of

0:

dec(EnemyShells[num].DP[i].Y, ShellSpeed);

1:

inc(EnemyShells[num].DP[i].X, ShellSpeed);

2:

inc(EnemyShells[num].DP[i].Y, ShellSpeed);

3:

dec(EnemyShells[num].DP[i].X, ShellSpeed);

end;

for i := 0 to length(waterObj) - 1 do

screen.Canvas.Draw(waterObj[i].X, waterObj[i].Y, StaticObjImg[7]);

screen.Canvas.Draw(EnemyShells[num].DP[0].X, EnemyShells[num].DP[0].Y,

EnemyShells[num].img);

for i := 0 to length(ForestObj) - 1 do

screen.Canvas.Draw(ForestObj[i].X, ForestObj[i].Y, StaticObjImg[4]);

end

else

begin

if not isAbleToMove1 then

begin

if TankPxMap[PntToMov1.Y][PntToMov1.X] = -10 then

begin

GameInterface.Enemy3ShellMovement.Enabled := false;

GameInterface.PlayerTankMovement.Enabled := false;

PlayerTank.isDestroyed := true;

GameInterface.PlayerRespawn.Enabled := true;

screen.Canvas.Draw(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, ExpBig);

GameInterface.DeleteExpBigPlayer.Enabled := true;

for k := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

for l := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

TankPxMap[l][k] := 0;

EnemyTanks[num].isShotMade := false;

end;

end;

if not isAbleToMove2 then

begin

if TankPxMap[PntToMov2.Y][PntToMov2.X] = -10 then

begin

GameInterface.Enemy3ShellMovement.Enabled := false;

GameInterface.PlayerTankMovement.Enabled := false;

PlayerTank.isDestroyed := true;

GameInterface.PlayerRespawn.Enabled := true;

screen.Canvas.Draw(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, ExpBig);

GameInterface.DeleteExpBigPlayer.Enabled := true;

for k := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

for l := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

TankPxMap[l][k] := 0;

EnemyTanks[num].isShotMade := false;

end;

end;

end;

end

else

begin

if not isAbleToMove1 then

begin

line := PntToMov1.Y div subobjlen;

column := PntToMov1.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deleteEnemy[num][1].line := line;

deleteEnemy[num][1].column := column;

GameInterface.DeleteExpSmallEnemy31.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

GameInterface.DeleteExpBigBase.Enabled := true;

end;

GameInterface.Enemy3ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

if not isAbleToMove2 then

begin

line := PntToMov2.Y div subobjlen;

column := PntToMov2.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deleteEnemy[num][2].line := line;

deleteEnemy[num][2].column := column;

GameInterface.DeleteExpSmallEnemy32.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

GameInterface.DeleteExpBigBase.Enabled := true;

end;

GameInterface.Enemy3ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

end;

end

else

begin

GameInterface.Enemy3ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

end;

procedure MoveEnemyShell4(screen: TImage; num: integer);

var

i, line, column, k, l: integer;

isAbleToMove1, isAbleToMove2: boolean;

PntToMov1, PntToMov2: TCoords;

begin

screen.Canvas.Rectangle(EnemyShells[num].DP[0].X, EnemyShells[num].DP[0].Y,

EnemyShells[num].DP[3].X + 1, EnemyShells[num].DP[3].Y + 1);

isAbleToMove1 := CheckBorder(EnemyShells[num].direction, num);

isAbleToMove2 := CheckBorder(EnemyShells[num].direction, num);

SetPntToMov(PntToMov1, PntToMov2, EnemyShells[num].direction, num);

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckObj(PxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckObj(PxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

isAbleToMove1 := CheckEnemyTank(TankPxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckEnemyTank(TankPxMap[PntToMov2.Y][PntToMov2.X]);

if isAbleToMove1 and isAbleToMove2 then

begin

for i := 0 to DimPntCnt - 1 do

case EnemyShells[num].direction of

0:

dec(EnemyShells[num].DP[i].Y, ShellSpeed);

1:

inc(EnemyShells[num].DP[i].X, ShellSpeed);

2:

inc(EnemyShells[num].DP[i].Y, ShellSpeed);

3:

dec(EnemyShells[num].DP[i].X, ShellSpeed);

end;

for i := 0 to length(waterObj) - 1 do

screen.Canvas.Draw(waterObj[i].X, waterObj[i].Y, StaticObjImg[7]);

screen.Canvas.Draw(EnemyShells[num].DP[0].X, EnemyShells[num].DP[0].Y,

EnemyShells[num].img);

for i := 0 to length(ForestObj) - 1 do

screen.Canvas.Draw(ForestObj[i].X, ForestObj[i].Y, StaticObjImg[4]);

end

else

begin

if not isAbleToMove1 then

begin

if TankPxMap[PntToMov1.Y][PntToMov1.X] = -10 then

begin

GameInterface.Enemy4ShellMovement.Enabled := false;

GameInterface.PlayerTankMovement.Enabled := false;

PlayerTank.isDestroyed := true;

GameInterface.PlayerRespawn.Enabled := true;

screen.Canvas.Draw(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, ExpBig);

GameInterface.DeleteExpBigPlayer.Enabled := true;

for k := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

for l := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

TankPxMap[l][k] := 0;

EnemyTanks[num].isShotMade := false;

end;

end;

if not isAbleToMove2 then

begin

if TankPxMap[PntToMov2.Y][PntToMov2.X] = -10 then

begin

GameInterface.Enemy4ShellMovement.Enabled := false;

GameInterface.PlayerTankMovement.Enabled := false;

PlayerTank.isDestroyed := true;

GameInterface.PlayerRespawn.Enabled := true;

screen.Canvas.Draw(PlayerTank.DP[0].X, PlayerTank.DP[0].Y, ExpBig);

GameInterface.DeleteExpBigPlayer.Enabled := true;

for k := PlayerTank.DP[0].X to PlayerTank.DP[3].X do

for l := PlayerTank.DP[0].Y to PlayerTank.DP[3].Y do

TankPxMap[l][k] := 0;

EnemyTanks[num].isShotMade := false;

end;

end;

end;

end

else

begin

if not isAbleToMove1 then

begin

line := PntToMov1.Y div subobjlen;

column := PntToMov1.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deleteEnemy[num][1].line := line;

deleteEnemy[num][1].column := column;

GameInterface.DeleteExpSmallEnemy41.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

GameInterface.DeleteExpBigBase.Enabled := true;

end;

GameInterface.Enemy4ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

if not isAbleToMove2 then

begin

line := PntToMov2.Y div subobjlen;

column := PntToMov2.X div subobjlen;

if (PxMap[line \* subobjlen][column \* subobjlen] <> 6) and

(PxMap[line \* subobjlen][column \* subobjlen] <> 3) then

begin

for k := line \* subobjlen to line \* subobjlen + subobjlen - 1 do

for l := column \* subobjlen to column \* subobjlen + subobjlen - 1 do

PxMap[k][l] := 0;

screen.Canvas.Draw(column \* subobjlen, line \* subobjlen, ExpSmall);

deleteEnemy[num][2].line := line;

deleteEnemy[num][2].column := column;

GameInterface.DeleteExpSmallEnemy42.Enabled := true;

end;

if PxMap[line \* subobjlen][column \* subobjlen] = 3 then

begin

screen.Canvas.Draw(columnbase \* subobjlen,

lineBase \* subobjlen, ExpBig);

GameInterface.DeleteExpBigBase.Enabled := true;

end;

GameInterface.Enemy4ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

end;

end

else

begin

GameInterface.Enemy4ShellMovement.Enabled := false;

EnemyTanks[num].isShotMade := false;

end;

end;

end.

unit UEnemyTanks;

interface

uses

Vcl.Graphics, Vcl.ExtCtrls, System.Classes;

const

DirImgCnt = 4;

DimPntCnt = 4;

TankSize = 80;

EnemyCount = 4;

eTankSpeed = 5;

type

TCoords = record

X, Y: integer;

end;

TEnemyCoordSpawn = array [1 .. EnemyCount] of TCoords;

TDimPntArr = array [0 .. DirImgCnt - 1] of TCoords;

TDirImgArr = array [0 .. DirImgCnt - 1] of TBitMap;

TEnemyTankr = record

direction: integer;

DP: TDimPntArr;

isShotMade, isDestroyed: boolean;

end;

TEnemies = array [1 .. EnemyCount] of TEnemyTankr;

procedure EnemyTankImgArrInit();

procedure CreateEnemyTank(X: integer; Y: integer; num: integer);

procedure MoveEnemyTank(screen: TImage; num: integer);

procedure ShootEnemyTank1(num: integer);

procedure ShootEnemyTank2(num: integer);

procedure ShootEnemyTank3(num: integer);

procedure ShootEnemyTank4(num: integer);

var

EnemyTankImg: TDirImgArr;

EnemyTanks: TEnemies;

EnemyCoordsSpawn: TEnemyCoordSpawn;

implementation

uses

UGameInterface, UGameMap, UTTankType, UShellTYpe, UEnemyShells;

procedure CreateEnemyTank(X: integer; Y: integer; num: integer);

var

i, j: integer;

begin

EnemyTanks[num].direction := 2;

EnemyTanks[num].isDestroyed := false;

EnemyTanks[num].isShotMade := false;

EnemyTanks[num].DP[0].X := X;

EnemyTanks[num].DP[0].Y := Y;

EnemyTanks[num].DP[1].X := X + TankSize - 1;

EnemyTanks[num].DP[1].Y := Y;

EnemyTanks[num].DP[2].X := X;

EnemyTanks[num].DP[2].Y := Y + TankSize - 1;

EnemyTanks[num].DP[3].X := X + TankSize - 1;

EnemyTanks[num].DP[3].Y := Y + TankSize - 1;

for i := EnemyTanks[num].DP[0].Y to EnemyTanks[num].DP[3].Y do

for j := EnemyTanks[num].DP[0].X to EnemyTanks[num].DP[3].X do

TankPxMap[i][j] := -num;

end;

procedure MoveEnemyTank(screen: TImage; num: integer);

function CheckBorder(direction: integer): boolean;

var

res: boolean;

begin

res := false;

case direction of

0:

res := ((EnemyTanks[num].DP[0].Y > 0) and

(EnemyTanks[num].DP[1].Y > 0));

1:

res := ((EnemyTanks[num].DP[1].X < MapSize - 1) and

(EnemyTanks[num].DP[3].X < MapSize - 1));

2:

res := ((EnemyTanks[num].DP[2].Y < MapSize - 1) and

(EnemyTanks[num].DP[3].Y < MapSize - 1));

3:

res := ((EnemyTanks[num].DP[0].X > 0) and

(EnemyTanks[num].DP[2].X > 0));

end;

result := res;

end;

function CheckObjId(Id: integer): boolean;

var

res: boolean;

begin

case Id of

0:

res := true;

4 .. 5:

res := true;

else

res := false;

end;

result := res;

end;

function CheckTank(Id, num: integer): boolean;

var

res: boolean;

begin

res := false;

case num of

1:

case Id of

- 2, -3, -4, -10:

res := false;

0:

res := true;

end;

2:

case Id of

- 1, -3, -4, -10:

res := false;

0:

res := true;

end;

3:

case Id of

- 2, -1, -4, -10:

res := false;

0:

res := true;

end;

4:

case Id of

- 2, -3, -1, -10:

res := false;

0:

res := true;

end;

end;

result := res;

end;

var

isAbleToMove1, isAbleToMove2, isAbleToMove3: boolean;

i, k, l: integer;

PntToMov1, PntToMov2, PntToMov3: TCoords;

begin

isAbleToMove3 := false;

screen.Canvas.Rectangle(EnemyTanks[num].DP[0].X, EnemyTanks[num].DP[0].Y,

EnemyTanks[num].DP[3].X + 1, EnemyTanks[num].DP[3].Y + 1);

for k := EnemyTanks[num].DP[0].Y to EnemyTanks[num].DP[3].Y do

for l := EnemyTanks[num].DP[0].X to EnemyTanks[num].DP[3].X do

TankPxMap[k][l] := 0;

isAbleToMove1 := CheckBorder(EnemyTanks[num].direction);

isAbleToMove2 := CheckBorder(EnemyTanks[num].direction);

if isAbleToMove1 and isAbleToMove2 then

begin

case EnemyTanks[num].direction of

0:

begin

PntToMov1 := EnemyTanks[num].DP[0];

PntToMov2 := EnemyTanks[num].DP[1];

dec(PntToMov1.Y);

dec(PntToMov2.Y);

PntToMov3.X := PntToMov1.X + TankSize div 2 - 1;

PntToMov3.Y := PntToMov1.Y;

end;

1:

begin

PntToMov1 := EnemyTanks[num].DP[1];

PntToMov2 := EnemyTanks[num].DP[3];

inc(PntToMov1.X);

inc(PntToMov2.X);

PntToMov3.X := PntToMov1.X;

PntToMov3.Y := PntToMov1.Y + TankSize div 2 - 1;

end;

2:

begin

PntToMov1 := EnemyTanks[num].DP[2];

PntToMov2 := EnemyTanks[num].DP[3];

inc(PntToMov1.Y);

inc(PntToMov2.Y);

PntToMov3.X := PntToMov1.X + TankSize div 2 - 1;

PntToMov3.Y := PntToMov1.Y;

end;

3:

begin

PntToMov1 := EnemyTanks[num].DP[0];

PntToMov2 := EnemyTanks[num].DP[2];

dec(PntToMov1.X);

dec(PntToMov2.X);

PntToMov3.X := PntToMov1.X;

PntToMov3.Y := PntToMov1.Y + TankSize div 2 - 1;

end;

end;

isAbleToMove1 := CheckObjId(PxMap[PntToMov1.Y][PntToMov1.X]);

isAbleToMove2 := CheckObjId(PxMap[PntToMov2.Y][PntToMov2.X]);

isAbleToMove3 := CheckObjId(PxMap[PntToMov3.Y][PntToMov3.X]);

end;

if isAbleToMove1 and isAbleToMove2 and isAbleToMove3 then

begin

isAbleToMove1 := CheckTank(TankPxMap[PntToMov1.Y][PntToMov1.X], num);

isAbleToMove2 := CheckTank(TankPxMap[PntToMov2.Y][PntToMov2.X], num);

if isAbleToMove1 and isAbleToMove2 then

for i := 0 to DimPntCnt - 1 do

case EnemyTanks[num].direction of

0:

dec(EnemyTanks[num].DP[i].Y, eTankSpeed);

1:

inc(EnemyTanks[num].DP[i].X, eTankSpeed);

2:

inc(EnemyTanks[num].DP[i].Y, eTankSpeed);

3:

dec(EnemyTanks[num].DP[i].X, eTankSpeed);

end;

end;

screen.Canvas.Draw(EnemyTanks[num].DP[0].X, EnemyTanks[num].DP[0].Y,

EnemyTankImg[EnemyTanks[num].direction]);

for k := EnemyTanks[num].DP[0].Y to EnemyTanks[num].DP[3].Y do

for l := EnemyTanks[num].DP[0].X to EnemyTanks[num].DP[3].X do

TankPxMap[k][l] := -num;

for i := 0 to length(ForestObj) - 1 do

screen.Canvas.Draw(ForestObj[i].X, ForestObj[i].Y, StaticObjImg[4]);

for i := 0 to length(SteelObj) - 1 do

screen.Canvas.Draw(SteelObj[i].X, SteelObj[i].Y, StaticObjImg[6]);

end;

procedure ShootEnemyTank1(num: integer);

var

code: integer;

begin

if not EnemyTanks[num].isShotMade then

begin

code := random(2);

case code of

1:

EnemyTanks[num].isShotMade := true;

0:

EnemyTanks[num].isShotMade := false;

end;

if EnemyTanks[num].isShotMade then

begin

CreateEnemyShell(EnemyTanks[num].direction, EnemyTanks[num].DP[0].X,

EnemyTanks[num].DP[0].Y, num);

GameInterface.Enemy1ShellMovement.Enabled := true;

end;

end;

end;

procedure ShootEnemyTank2(num: integer);

var

code: integer;

begin

if not EnemyTanks[num].isShotMade then

begin

code := random(2);

case code of

1:

EnemyTanks[num].isShotMade := true;

0:

EnemyTanks[num].isShotMade := false;

end;

if EnemyTanks[num].isShotMade then

begin

CreateEnemyShell(EnemyTanks[num].direction, EnemyTanks[num].DP[0].X,

EnemyTanks[num].DP[0].Y, num);

GameInterface.Enemy2ShellMovement.Enabled := true;

end;

end;

end;

procedure ShootEnemyTank3(num: integer);

var

code: integer;

begin

if not EnemyTanks[num].isShotMade then

begin

code := random(2);

case code of

1:

EnemyTanks[num].isShotMade := true;

0:

EnemyTanks[num].isShotMade := false;

end;

if EnemyTanks[num].isShotMade then

begin

CreateEnemyShell(EnemyTanks[num].direction, EnemyTanks[num].DP[0].X,

EnemyTanks[num].DP[0].Y, num);

GameInterface.Enemy3ShellMovement.Enabled := true;

end;

end;

end;

procedure ShootEnemyTank4(num: integer);

var

code: integer;

begin

if not EnemyTanks[num].isShotMade then

begin

code := random(2);

case code of

1:

EnemyTanks[num].isShotMade := true;

0:

EnemyTanks[num].isShotMade := false;

end;

if EnemyTanks[num].isShotMade then

begin

CreateEnemyShell(EnemyTanks[num].direction, EnemyTanks[num].DP[0].X,

EnemyTanks[num].DP[0].Y, num);

GameInterface.Enemy4ShellMovement.Enabled := true;

end;

end;

end;

procedure EnemyTankImgArrInit();

var

i: integer;

begin

for i := 0 to DirImgCnt - 1 do

EnemyTankImg[i] := TBitMap.Create;

EnemyTankImg[0].LoadFromFile('..\icons\EnemyTank\etUp.bmp');

EnemyTankImg[1].LoadFromFile('..\icons\EnemyTank\etRight.bmp');

EnemyTankImg[2].LoadFromFile('..\icons\EnemyTank\etDown.bmp');

EnemyTankImg[3].LoadFromFile('..\icons\EnemyTank\etLeft.bmp');

EnemyCoordsSpawn[1].X := 0;

EnemyCoordsSpawn[1].Y := 0;

EnemyCoordsSpawn[2].X := 320;

EnemyCoordsSpawn[2].Y := 0;

EnemyCoordsSpawn[3].X := 640;

EnemyCoordsSpawn[3].Y := 0;

EnemyCoordsSpawn[4].X := 960;

EnemyCoordsSpawn[4].Y := 0;

end;

end.

unit UFileRouthine;

interface

uses

vcl.Dialogs, System.SysUtils;

type

TString = string[8];

TListData = record

Name: TString;

Password: TString;

Score: integer;

end;

Tlist = array of TListData;

procedure AddUser();

procedure CheckUser(str1, str2: string);

procedure ReadUsersData();

procedure SaveData();

var

header: Tlist;

current: integer;

implementation

uses

UAuthentication, UMainMenu, UGameInterface, ULeadersTable;

procedure ReadUsersData();

var

Name, pass, Score: TextFile;

begin

AssignFile(Name, '..\UsersList\UserName.txt');

AssignFile(pass, '..\UsersList\UserPass.txt');

AssignFile(Score, '..\UsersList\UserScore.txt');

reset(name);

reset(pass);

reset(Score);

while not EOF(name) do

begin

Setlength(header, length(header) + 1);

readln(name, header[length(header) - 1].Name);

readln(pass, header[length(header) - 1].Password);

readln(Score, header[length(header) - 1].Score);

end;

closeFile(Name);

closeFile(pass);

closeFile(Score);

end;

procedure AddUser();

var

Name, pass, Score: TextFile;

i: integer;

begin

AssignFile(Name, '..\UsersList\UserName.txt');

AssignFile(pass, '..\UsersList\UserPass.txt');

AssignFile(Score, '..\UsersList\UserScore.txt');

Rewrite(name);

Rewrite(pass);

Rewrite(Score);

for i := 0 to length(header) - 1 do

begin

writeln(name, header[i].Name);

writeln(pass, header[i].Password);

writeln(score, header[i].score);

end;

closeFile(Name);

closeFile(pass);

closeFile(Score);

end;

procedure CheckUser(str1, str2: string);

var

i: integer;

flag: boolean;

begin

i := 0;

flag := false;

while (i <= length(header) - 1) and not flag do

begin

if header[i].Name = str1 then

begin

current := i;

flag := true;

end;

inc(i);

end;

if not flag then

begin

ShowMessage('Новый пользователь под ником: ' + str1 + ' создан');

SetLength(header, length(header) + 1);

current := length(header) - 1;

header[current].Name := str1;

header[current].Password := str2;

header[current].Score := 0;

Adduser();

isAuthorised := true;

Auth.Destroy;

Mainmenu.Show;

end;

if flag and (header[current].Password = str2) then

begin

ShowMessage('Пользователь под ником ' + str1 + ' авторизован.');

isAuthorised := true;

Auth.Destroy;

Mainmenu.Show;

end;

if flag and (header[current].Password <> str2) then

ShowMessage('Пароль неверный.');

end;

procedure SaveData();

var

Name, pass, Score: TextFile;

i: integer;

begin

AssignFile(Name, '..\UsersList\UserName.txt');

AssignFile(pass, '..\UsersList\UserPass.txt');

AssignFile(Score, '..\UsersList\UserScore.txt');

rewrite(name);

rewrite(pass);

rewrite(Score);

for i := 0 to length(header) - 1 do

begin

writeln(name, header[i].Name);

writeln(pass, header[i].Password);

writeln(score, header[i].score);

end;

SetLength(header, 0);

closeFile(Name);

closeFile(pass);

closeFile(Score);

end;

end.