**Содержание**

Введение5

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству6

2. Анализ требований к программному средству и разработка функциональных требований14

3. Проектирование программного средства 21

3.1 Общая схема программного средства21

3.2 Проектирование алгоритмов подачи сюжета22

3.3 Проектирование алгоритмов работы с графикой и звуком27

3.4 Проектирование алгоритмов, реализующих игровую механику 31

3.5 Проектирование алгоритмов работы с настройками35

4. Создание (конструирование) программного средства 38

4.1 Конструирование алгоритма подачи сюжета(загадки) 38

4.2 Работа с графикой 39

**4.2.1 Переходы на другие подуровни39**

**4.2.2 Механизм обзора персонажа40**

4.3 Влияние уровня сложности на урон ловушек, эффекты от применения зелий и магии 40

4.4 Описание классов 41

5. Тестирование и проверка работоспособности программного средства 44

5.1 Таблица тестов44

5.2 Внесённые изменения57

6. Руководство по установке и использованию59

Заключение4

Список использованных источников4

Приложение А4

Приложение Б4

**Введение:**

Широкое развитие информационных технологий и информационных коммуникаций открыло возможность для передачи через сети не только текстовых сообщений, но также файлов различных типов. С развитием локальных сетей появилась потребность в наличии обособленного хранилища данных, которое хранит всю необходимую информацию компании или просто группы лиц.

Однако, развитие технологий хранения и распределения информации неизбежно привело и к развитию методов нелегального её получения и вопрос безопасности файловых серверов встал довольно остро. Организации, чьи локальные сети имеют доступ и в глобальную сеть серьёзно занимаются вопросами защиты своей информации, но владельцы небольших изолированных локальных сетей зачастую уверены, что они в безопасности и банально игнорируют вопрос защиты своих данных, что позволяет злоумышленникам проникнуть в их сеть и без особых проблем получить данных.

Изучив данный вопрос, я поставил себе цель – создать клиент-серверное приложение для малых организаций и локальных сетей, которое бы обеспечивало приемлемый уровень безопасности и при этом было бы удобным в использовании.

**Глава 1.**

На данный момент существует достаточно много решений, дающих пользователям возможность хранить свои данных удалённых серверах.

Основными примерами таковых являются:

1. Google Диск

Google Диск — это файловый хостинг, созданный и поддерживаемый компанией Google. Его функции включают хранение файлов в Интернете, общий доступ к ним и совместное редактирование. В состав Google Диска входят Google Документы, Таблицы и Презентации — набор офисных приложений для совместной работы над текстовыми документами, электронными таблицами, презентациями, чертежами, веб-формами и другими файлами. Общедоступные документы на Диске индексируются поисковыми системами.

Google Диск был представлен 24 апреля 2012 года, а к октябрю 2014 года насчитывал 240 миллионов ежемесячно активных пользователей.

15 мая 2018 года компания Google объявила о переименовании Google Drive (Google Диск) в Google One.

(Google Drive [Электронный ресурс]. – Электронные данные. – Режим доступа: http://wikipedia.org/.)

Как утверждает компания Google: Файлы, загруженные на Google Диск, хранятся в защищенных центрах обработки данных. Больше ничего о защите данных в данном сервисе неизвестно. Впрочем, обещания компании Google по поводу абсолютной безопасности оказались ложными и в 2015 Google Диск был взломан.

Исследователи выяснили, что все из указанных сервисов предоставляют непрерывный доступ приложений-клиентов к серверам с помощью токенов, которые генерируются при первой аутентификации пользователя. Эти токены хранятся на ПК в специальных файлах, в реестре Windows или в Windows Credential Manager (зависит от приложения).

Аналитики написали приложение под названием Switcher, которое способно проникать на компьютер жертвы в виде вложения в почтовом сообщении или через уязвимость в браузере. Попав на машину, Switcher заменяет токен для доступа к аккаунту в определенном облачном сервисе на свой собственный, связанный с учетной записью злоумышленника.

Затем Switcher перезагружает приложение-клиент облачного сервиса, после чего оно уже использует подменный токен. В результате синхронизация папки на ПК пользователя происходит с облачным аккаунтом злоумышленника. Switcher помещает в эту папку копию подлинного токена, которую он создал заранее. Таким образом, хакер получает в свое распоряжение не только файлы жертвы, но и токен для последующего доступа к аккаунту жертвы уже со своего компьютера.

(Google Drive – безопасность [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://support.google.com/drive/answer/141702?hl=ru&co=GENIE.Platform%3DDesktop>.)

(Google Drive – взлом [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://safe.cnews.ru/news/top/najden_legkij_sposob_vzloma_akkuantov>.)

Исходя из этого, к плюсам данной системы можно отнести:

+ широкие возможности по редактированию информации

+ синхронизация файлов

+ привязка к аккаунту

Основные минусы:

- сомнительная безопасность системы

- наличие только 15 гб бесплатного пространства

- индексация файлов поисковой системой, что может приводит к утечке вашей информации в общий доступ, если вы не успели настроить приватность

1. FTP-сервер

FTP (англ. *File Transfer Protocol* — протокол передачи файлов) — стандартный протокол, предназначенный для передачи файлов по TCP-сетям (например, Интернет). Использует 21-й порт. FTP часто используется для загрузки сетевых страниц и других документов с частного устройства разработки на открытые сервера хостинга.

Протокол построен на архитектуре «клиент-сервер» и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль открытым текстом, или же, если это разрешено на сервере, они могут подключиться анонимно. Можно использовать протокол SSH для безопасной передачи, скрывающей (шифрующей) логин и пароль, а также шифрующей содержимое.

FTP не разрабатывался как защищённый (особенно по нынешним меркам) протокол и имеет многочисленные уязвимости в защите. В мае 1999 авторы RFC 2577 свели уязвимости в следующий список проблем:

* Скрытые атаки (bounce attacks)
* Спуф-атаки (spoof attacks)
* Атаки методом грубой силы (brute force attacks)
* Перехват пакетов, сниффинг (packet capture, sniffing)
* Защита имени пользователя
* Захват портов (port stealing)

FTP не может зашифровать свой трафик, все передачи — открытый текст, поэтому имена пользователей, пароли, команды и данные могут быть прочитаны кем угодно, способным перехватить пакет по сети. Эта проблема характерна для многих спецификаций Интернет-протокола (в их числе SMTP, Telnet, POP, IMAP), разработанных до создания таких механизмов шифрования, как TLS и SSL. Обычное решение этой проблемы — использовать «безопасные», TLS-защищённые версии уязвимых протоколов (FTPS для FTP, TelnetS для Telnet и т. д.) или же другой, более защищённый протокол, вроде SFTP/SCP, предоставляемого с большинством реализаций протокола Secure Shell.

(FTP/FTPS [Электронный ресурс]. – Электронные данные. – Режим доступа: http://wikipedia.org/.)

(SSH [Электронный ресурс]. – Электронные данные. – Режим доступа: http://wikipedia.org/.)

Основные плюсы:  
+ бесплатность  
+ возможность настройки доступа и аутентификации  
+ расширение безопасности за счёт использования SSL, SSH, FTPS

Минусы:  
- необходимость ручной настройки

Таким образом, первичными требованиями к программному средству являются

1. Обеспечение шифрования данных в процессе передачи и на самом сервере

2. Реализация механизмов регистрации и аутентификации на сервере

3. Разделение рабочих пространств пользователей

4. Обеспечение получения файлов с сервера

5. Обеспечение загрузки файлов на сервер

«Список литературы»

[1] Google Drive [Электронный ресурс]. – Электронные данные. – Режим доступа: http://wikipedia.org/.;

[2] Google Drive – безопасность [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://support.google.com/drive/answer/141702?hl=ru&co=GENIE.Platform%3DDesktop>.;

[3] Google Drive – взлом [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://safe.cnews.ru/news/top/najden_legkij_sposob_vzloma_akkuantov>.;

[4] FTP/FTPS [Электронный ресурс]. – Электронные данные. – Режим доступа: http://wikipedia.org/.;

[5] SSH [Электронный ресурс]. – Электронные данные. – Режим доступа: http://wikipedia.org/.;

**2.Анализ требований к программному средству и разработка функциональных требований**

В данной главе будет приведено текстовое описание работы программного средства, на основании которого будут расширены и уточнены первоначальные функциональные требования.

Клиент:

При запуске приложения открывается окно в котором присутствуют поля для ввода логина и пароля, кнопки «Регистрация», «Авторизация» и «Выход». Также должны присутствовать кнопки «Загрузить на сервер» и «Загрузить с сервера». В середине окна располагается поле, в котором пользователь видит список загруженных на сервер файлов.

При нажатии на кнопку регистрация при заполненных полях ввода пароля и логина регистрируется новый пользователь. Серверу отправляется запрос на регистрацию, после чего приходит ответ, который выводится во всплывающем окне.

При нажатии на кнопку авторизация и заполненных полях ввода серверу отправляется запрос на авторизацию вместе с данными для авторизации. После чего на клиент приходит ответ с информации от сервера, а именно статусе авторизации.

При нажатии на кнопку выход серверу отправляется запрос на выход клиента, после чего сервер сохраняет последние изменения и разрывает соединение с клиентом.

При нажатии на кнопку «Загрузить на сервер», в случае, если клиент успешно авторизован происходит открытие диалогового окна, в котором пользователю будет предложено выбрать файл для загрузки, после того, как пользователь выбрал файл и подтвердил выбор, начинается передача данных серверу в зашифрованном виде. В случае, если пользователь не авторизован, то будет выведено соответствующее предупреждение и отправка файла не будет осуществлена.

При нажатии на кнопку «Загрузить с сервера» происходит отправка запроса на сервер, в случае успеха клиенту будет представлено диалоговое окно, в котором можно будет указать место для сохранения файла и имя сохраняемого файла. Если пользователь не авторизован, то будет выведено соответствующее предупреждение.

Список функциональных требований

1. Открытие окна программы при её запуске
2. Получение данных о нахождении сервера при запуске программы
3. Установления соединения с сервером при запуске программы
4. Получения ключа для шифрования от сервера
5. Отправка зашифрованного запроса на регистрацию при нажатии на кнопку регистрация и заполненных полях логина и пароля
6. Отправка шифрованного запроса на авторизацию при нажатии на кнопку авторизации и заполненных полях логина и пароля
7. Отправка шифрованного запроса на получение файла
8. Отправка шифрованного запроса на передачу файла
9. Отображение списка текущих файлов пользователя на сервере
10. Шифрование данных и их расшифровка
11. Закрытие соединения при нажатии на кнопку «Выход»
12. Корректная обработка ошибок с выводом пользователю информации о них
13. Возможность попытаться восстановить потерянное соединение

**3. Проектирование программного средства**

**3.1 Общая схема программного средства**

Программное средство Praetorian будет представлять собой оконное приложение, которое будет открывать диалоговые окна для выполнения действий по сохранению или отправке файлов, а также всплывающие окна для уведомления пользователя об ошибках или необходимости совершения предварительных действий. Помимо этого, в состав приложения входит сервер, с которым будут взаимодействовать клиентские приложения.



Рисунок 3.1 – Общая схема программы

Главное окно – стартовое окно программы, в котором пользователю будут доступны возможности по регистрации и авторизации на сервере, возможность загрузки своих файлов на сервер, а также по загрузке своих файлов с сервера. Все данные будут передаваться между клиентом и сервером в зашифрованном виде, а также храниться в зашифрованном виде на самом сервере.

**3.2 Проектирование алгоритмов определения адреса сервера в сети LAN**

Одним из функциональных требований к проекту было возможность динамического определения IP-адреса сервера в локальной сети, для того, чтобы пользователь не был обязан задумываться об этом вопросе, однако возможность ручной настройки IP-адреса сервера будет оставлена и реализовываться через изменение config файла клиента. Для динамического определения IP-адреса можно использовать широковещательный UDP-запрос на локальную сеть, после получением сервером данного запроса в ответ будет выслан IP-адрес сервера. В случае, если ответ от сервера не был получен с первой попытки, то происходит повторная отправка UDP-запроса. Это будет происходить до тех пор, пока не будет получен ответ от сервера, либо не будет превышено число попыток.



Рисунок 3.2. – алгоритм получения адреса сервера в сети

**3.3 Проектирование алгоритмов передачи файлов по сети**

Основной возможностью данного приложения является возможность передавать файлы в шифрованном виде по локальной сети. Для её реализации я буду использовать протокол TCP, поскольку он обеспечивает надёжную передачу пакетов по сети с контролем за потерями, что критически важно при передаче зашифрованных файлов, так как в зависимости от алгоритма шифрования это может привести либо к частичному, либо к полному искажению передаваемой информации. Однако, для осуществления передачи файлов различных размеров необходимо задать определённую структуру передаваемых данных, так как TCP не осуществляет контроль за полным размером передаваемых данных. Для этого, мною была предложена следующая структура для передачи данных серверу и с сервера.

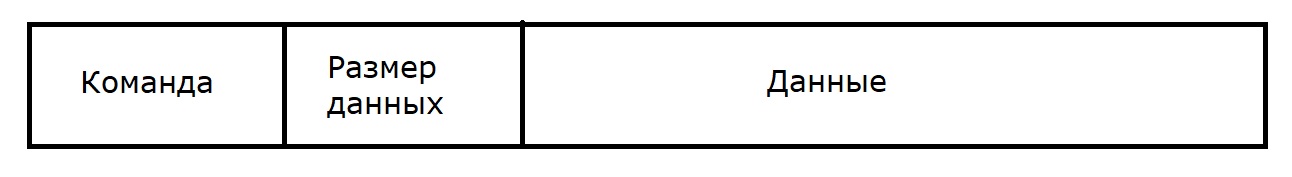


Рисунок 3.3 – Формат запроса

1. Команда – поле команды будет иметь фиксированный размер в 1 байт и содержать в себе численное представление команды, которую будет присылать клиент. Для размера поля команды в 1 байт можно реализовать 255 команд, что, по-моему мнению, на данный момент избыточно, однако позволяет продолжить поддержку проекта и добавление новых команд, без изменения в структуре запроса.
2. Размер данных – поле размер данных будет иметь фиксированный размер в 8 байт и содержать в себе длинное целое число, в котором будет храниться размер поля данных в байтах. Поле размера данных позволит серверу корректно определить размер получаемых данных и выделить на их обработку ровно столько памяти, сколько для этого нужно и позволит снять ограничение на размер передаваемого файла, так как длинное целое число может вместить в себя значение максимальное значение которого равно 18 446 744 073 709 551 615.
3. Данные – поле в котором будут храниться данные, передаваемые серверу, такие как логин/пароль или непосредственно содержимое файла.

Структура ответа от сервера отличается лишь тем, что сервер вместо команды будет присылать свой ответ на команду и в зависимости от его успешности, либо запрошенное содержимое, либо описание ошибки.



Рисунок 3.4. – Алгоритм формирования запроса/ответа

* 1. **Проектирование алгоритмов шифрования**

На данный момент существует большое количество типов и стандартов шифрования. В данной работе я буду использовать готовые библиотеки для реализации шифрования, так как они были написаны специалистами, что минимизирует риск допустить ошибку при реализации или написании собственного алгоритма шифрования, либо самостоятельной реализации уже готового алгоритма.

Шифрованием называется обратимый процесс преобразования информации с целью их сокрытия от непривилегированных лиц. Важной особенностью любого метода шифрования считается использование ключа, утверждающего выбор определённого преобразования из всех возможных для данного метода.

Для того, чтобы информация, прошедшая шифрование, превратилась в бессмысленный для постороннего пользователя набор символов, были разработаны специальные алгоритмы шифрования. В целом, все алгоритмы можно разбить на 2 группы: симметричные и асимметричные.

Симметричные алгоритмы (*AES, CAST, ГОСТ, Blowfish, DES*) используют для шифровки и дешифровки один ключ данных. Главный недостаток этих алгоритмов – в случае кражи ключа шифрования, злоумышленник может похитить и расшифрованные данные. Кроме того, существуют технологии криптоатак, позволяющие дешифровать данные без использования ключа шифрования.

Асимметричные алгоритмы (El-Gamal, RSA) используют для шифрования разные ключи – открытый и закрытый. Открытый ключ передаётся по незащищённому каналу и предназначен для проверки ЭЦП и шифровки сообщения. Для генерации ЭЦП и дешифровки применяется секретный ключ. Асимметричные алгоритмы шифрования частично решают проблему перехвата ключей, так как для успешного расшифрования текста ему придётся решить проблему дискретного логарифма.

Так как шифрование данных будет происходить при передаче с клиента на сервер и на самом сервере, то необходимо выбрать алгоритмы для обоих случаев.

1. Передача между клиентом и сервером

А) Симметричные алгоритмы  
Возможность применения в данном случае симметричных алгоритмов представляется неудобным и ненадёжным вариантом, так как у клиента и сервера должен быть единый и одинаковый ключ, который необходимо передавать по сети в открытом виде, либо настраивать его на каждом клиенте. Поэтому использование симметричных алгоритмов шифрования в данном случае отвергается.

Б) Ассиметричные алгоритмы шифрования

Основным плюсом данных алгоритмов является наличие двух ключей – открытого и закрытого, при этом сообщение зашифрованное открытым/закрытым ключом можно расшифровать только имея закрытый/открытый ключ шифрования. Для обмена ключами шифрования между клиентом и сервером можно использовать протокол Диффи-Хелмана, суть которого представлена на рисунке ниже

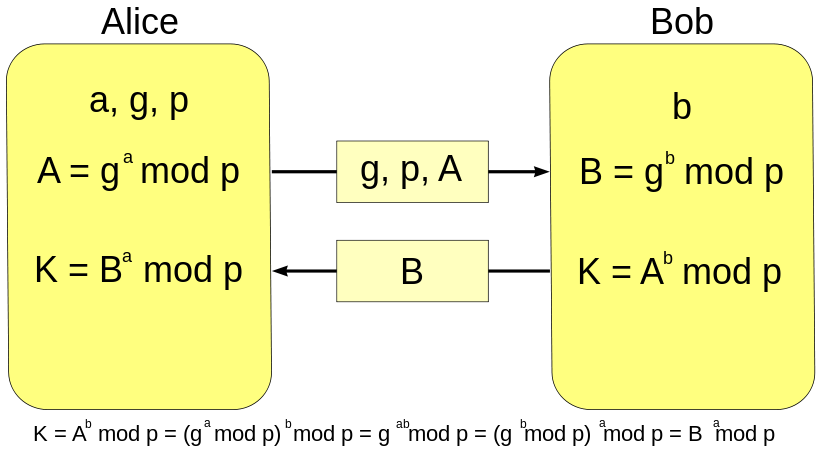
****

Рисунок 3.5. – протокол Диффи-Хелмана

При работе алгоритма каждая сторона:

1. генерирует случайное натуральное число *a* — закрытый ключ
2. совместно с удалённой стороной устанавливает открытые параметры *p* и *g* (обычно значения *p* и *g* генерируются на одной стороне и передаются другой), где

*p* является случайным простым числом

*(p-1)/2* также должно быть случайным простым числом (для повышения безопасности)

*g* является первообразным корнем по модулю *p (*также является простым числом*)*

1. вычисляет открытый ключ *A*, используя преобразование над закрытым ключом

*A = ga* mod *p*

1. обменивается открытыми ключами с удалённой стороной
2. вычисляет общий секретный ключ *K*, используя открытый ключ удаленной стороны *B* и свой закрытый ключ *a*

*K = Ba* mod *p*

*К* получается равным с обеих сторон, потому что:

*Ba* mod *p = (gb* mod *p)a* mod *p = gab* mod *p = (ga* mod *p)b* mod *p = Ab* mod *p*

В практических реализациях для *a* и *b* используются числа порядка 10100 и *p* порядка 10300. Число *g* не обязано быть большим и обычно имеет значение в пределах первого десятка.

2) Шифрование на сервере

Для шифрования на сервере может использоваться любой тип шифрования так как передачи ключа не происходит. Поэтому данный анализ будет производиться в контексте сложности операции шифрования/дешифрирования, поскольку серверу необходимо будет обрабатывать большие объёмы данных и работать с большим числом пользователей.

А) Симметричные алгоритмы

DES — алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый правительством США в 1977 году как официальный стандарт (FIPS 46-3). Размер блока для DES равен 64 бита. В основе алгоритма лежит сеть Фейстеля с 16 циклами и ключом, имеющим длину 56 бит.

Входными данными для DES служат:

* блок размером n бит;
* ключ размером k бит.

На выходе (после применения шифрующих преобразований) получается зашифрованный блок размером n бит, причём незначительные различия входных данных, как правило, приводят к существенному изменению результата.

Блочные шифры реализуются путём многократного применения к блокам исходного текста некоторых базовых преобразований.

Базовые преобразования:

* сложное преобразование на одной локальной части блока;
* простое преобразование между частями блока.

Так как преобразования производятся поблочно, требуется разделение исходных данных на блоки необходимого размера. При этом формат исходных данных не имеет значения (будь то текстовые документы, изображения или другие файлы). Данные должны интерпретироваться в двоичном виде (как последовательность нулей и единиц) и только после этого должны разбиваться на блоки. Все вышеперечисленное может осуществляться как программными, так и аппаратными средствами.

Таким образом основная сложность алгоритма DES заключается в выполнениях операции перестановки и хранению таблиц шифрования для DES (так называемых S-box), а также в необходимости написания алгоритма генерации ключа, так как на основании алгоритма DES выделяются множества слабых и частично слабых ключей.

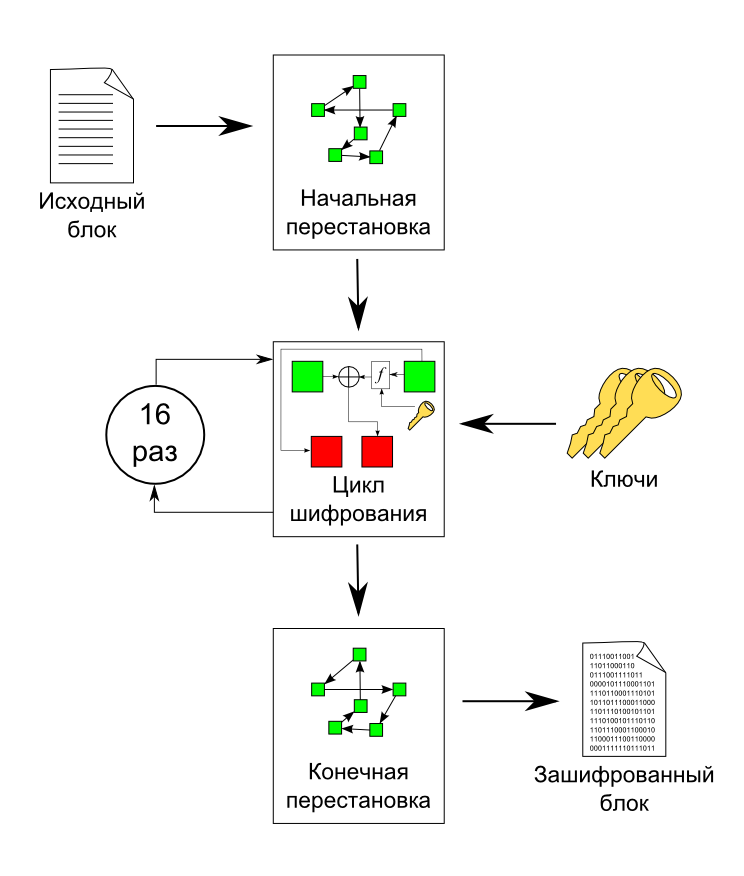


Рисунок 3.6 – Схема шифрования в алгоритме DES

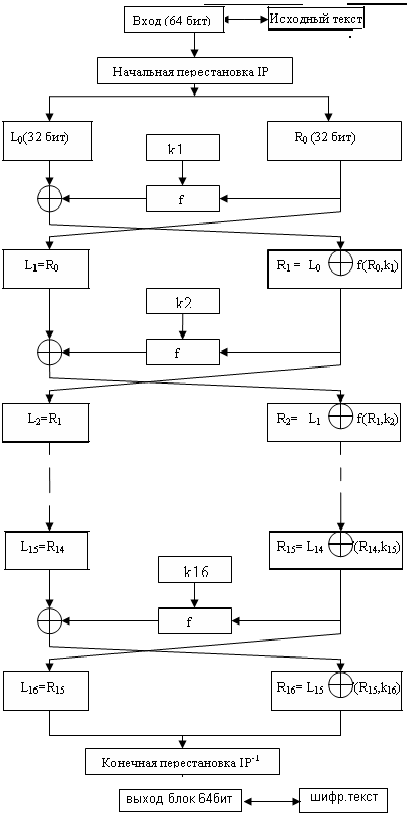


Рисунок 3.7 – подробная схема шифрования алгоритма DES

Blowfish — криптографический алгоритм, реализующий блочное симметричное шифрование с переменной длиной ключа. Разработан Брюсом Шнайером в 1993 году. Представляет собой сеть Фейстеля. Выполнен на простых и быстрых операциях: XOR, подстановка, сложение. Является незапатентованным и свободно распространяемым.

Алгоритм состоит из двух частей: расширение ключа и шифрование данных. На этапе расширения ключа исходный ключ (длиной до 448 бит) преобразуется в 18 32-битовых подключей и в 4 32-битных S-блока, содержащих 256 элементов.

Blowfish зарекомендовал себя как надёжный алгоритм, поэтому реализован во многих программах, где не требуется частая смена ключа и необходима высокая скорость шифрования/расшифровывания.

* хэширование паролей
* защита электронной почты и файлов
  + GnuPG (безопасное хранение и передача)
* в линиях связи: связка ElGamal (не запатентован) или RSA (действие патента закончилось в 2000 году) и Blowfish вместо IDEA
  + в маршрутизаторе Intel Express 8100 с ключом длиной 144 бита
* обеспечение безопасности в протоколах сетевого и транспортного уровня
  + SSH (транспортный уровень)
  + OpenVPN (создание зашифрованных каналов)

В целом алгоритм Blowfish подходит для реализации шифрования данных на сервере, так как базируется на простых операциях, что положительно сказывается на быстродействии сервера. Однако, в рамках разработки планируется ввести систему, которая осуществляет перешифрование данных через определённый промежуток времени, что может привести к появлению слабых S-блоков в шифраторе.

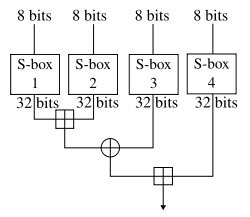


Рисунок 3.8 – функция преобразования Blowfish

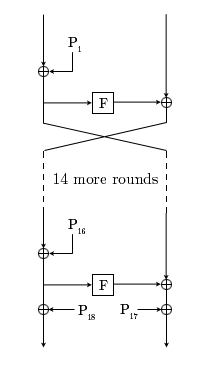


Рисунок 3.9 – алгоритм шифрования Blowfish

Потоковое шифрование – основная идея потоковых криптосистем заключается в шифровании исходного текста М с помощью криптографического ключа К, длина которого равна длине текста. Каждый бит шифротекста С является исключающим или соответствующих битов исходного текста и ключевого потока. Для генерации ключа используют последовательные сдвиговые регистры с обратной связью, криптостойкость алгоритма зависит от генерирующего многочлена.

Основным плюсом данного метода является его простота реализации и высокая производительность, так как операция xor является одной из самой малозатратных, при этом можно обеспечить высокую степень криптоскойкости за счёт изменения генерирующего многочлена.

Б) Ассиметричные алгоритмы

Несмотря на то, что на данный момент существует большое количество криптосистем с открытым и закрытым ключом, их всех объединяет один недостаток – это достаточно сложные вычислительные операции, которые нужно проводить для шифрования, дешифрования данных, такие как нахождение большого целого простого числа и возведение в степень по модулю. Поэтому, я считаю, что применение ассиметричных алгоритмов для шифрования данных на сервере не рациональным, поскольку представленные выше алгоритмы шифрования в данном случае будут иметь почти аналогичную криптостойкость, при этом экономя ресурсы сервера.

Таким образом, на основании анализа, мною был предпочтён алгоритм потокового шифрования, так как он наилучше сочетает в себе качества, соответствующие требованиям проекта.

**4. Создание (конструирование) программного средства**

На этапе конструирования основной моей задачей стала реализация функциональных требований на основе разработанных алгоритмов. В данной главе я планирую рассмотреть наиболее интересные решения, внесённые мной в программное средство.

* 1. **Конструирование алгоритма подачи сюжета(загадки)**

Игровое событие «загадка» имеет место при переходе персонажа в особую область игровой локации. На маске локации данная область обозначается символом «h». Происходит открытие нового окна, в котором описан текст загадки, присутствует поле для дачи ответа и кнопка, нажатие которой приводит к подтверждению данного пользователем ответа. Для того, чтобы при первом открытии окна загадок выводилось приветственное сообщение в класс TPlayer (описание ниже) было введено поле IsFirstMeetWithHerma. При значении поля равным true открывается вводное сообщение, после чего данному полю присваивается значение False. И при последующих открытиях будут выводиться уже загадки.

Загрузка загадок в динамическую структуру происходит при создании окна.

При каждом новом вызове после приветственного слова выбирается и выводится на экран загадка. Для того, чтобы загадки не повторялись, выбранная загадка удаляется из списка.

Переда началом процедуры выборки производится проверка, что EnigmaHead.Next<>nil, в противном случае пользователю выводится сообщении о благосклонности Хермеуса Моры к нему.

Для дополнительного усложнения процесса отгадывания было введено временное ограничение в 60 секунд. Время отсчитывается при помощи компонента TTimer. Остаток времени выводится на экран.

Для дачи ответа необходимо ввести его в поле ввода и нажать кнопку «Дать ответ». Правильный ответ будет принят независимо от регистра. Всего возможны три окончания мини игры «загадка»:  
а) Дан правильный ответ

1.Вывод сообщения об успехе

2.Закрытие окна загадок

3.Возврат к игре

б) Дан неверный ответ

1.Вывод сообщения о провале

2.Закрытие окна загадок

3.Возврат к игре

4.Окончание игры

в) Время вышло

1.Вывод сообщения об истечении времени

2.Закрытие окна загадок

3.Возврат к игре

4.Окончание игры

* 1. **Работа с графикой**

**4.2.1 Переходы на другие подуровни**

Одним из основных элементов, обеспечивающих данную возможность является способ представления игрового уровня в программе. Реализация алгоритма загрузки и разбиения проводилась в соответствии с алгоритмом, рассмотренным в ходе фазы проектирования. Хранение локации осуществляется в текстовом файле. Локация загружается в динамическую структуру двусвязный список при создании окна игры. Выбор локации осуществляется в зависимости от уровня сложности.

Так как в конце каждой строки текстового файла помещаются управляющие символы #13#10, то для их обработки длина динамического массива задаётся больше, чем размер подуровня.

Обработка перемещений персонажа происходит в событии формы OnKeyDown. Поскольку переход между подуровнями осуществляется только в двух направления, то соответствующие проверки были помещены в блоки обработки нажатий клавиш, осуществляющих действия перехода вперёд и назад.

По первоначальной задумке, проверка на равенство указателя на следующий/предыдущий подуровень не предусматривалась, так как локации со всех сторон окружены стенами, однако для обеспечения гибкости в создании новых карт было решено добавить данную проверку. Координата персонажа меняется при перемещении с подуровня на подуровень в зависимости от вида перехода: следующий-предыдущий, предыдущий-следующий. При переходе вперёд загружается и отрисовывается следующий подуровень. После чего персонаж отрисовывается в начала подуровня. При переходе назад загружается и отрисовывается предыдущий подуровень. После чего персонаж отрисовывается в конце подуровня.

**4.2.2 Механизм обзора персонажа**

Темнота реализуется при помощи спрайтов темных квадратов и затемнённых версий исходных игровых текстур. Игровой персонаж обладает областью видимости следующего вида:

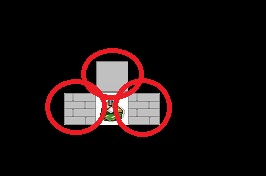


Рис 4.1 – Красным выделена область видимости игрового персонажа

В зоны, помеченные на рисунке красным цветом происходит отрисовка затемнённых версий исходных текстур, в зависимости от значения элемент маски данной области.

В качестве параметров в данную процедуру передаются координаты области, которую необходимо отобразить и канву, на которой необходимо это отобразить.

В процессе передвижения по подуровню, увиденные области не затемняются, но при переходе на другой подуровень и возвращении назад, подуровень вновь будет полностью тёмным.

**4.3 Влияние уровня сложности на урон ловушек, эффекты от применения зелий и магии**

Уровень сложности представляет собой игровую характеристику, которая отвечает за уровень враждебности игрового окружения, что отражается в количестве наносимого игроку урона, стоимости и эффектов заклинаний, эффективности зелий.

Вариация уровней сложности в игре была реализована при помощи использования типа перечисление и переменных данного типа.

Type

TDifficult = (Easy,Medium,Hard);

Влияние на ловушки:

Суммарный негативный эффект от ловушки складывается из наносимого урона здоровью и увеличения уровня стресса персонажа при попадании в данный игровой объект.

Вводится понятие базового урона ловушки – это средний урон, ловушкой при попадании в неё. Урон здоровью есть случайное значение, взятое из промежутка

, т.е. отклонение урона от базового составляет . Из полученного значения вычитается величина равная , где Luck – это величина параметра удачи игрового персонажа.

Листинг процедуры расчёта урона представлен в описании класса TTrap.

Увеличение стресса рассчитывается исходя из уровня сложности и полученного значения урона ловушки следующим образом:

Коэффициентом является поле класса TTrap factor, которая зависит от уровня сложности следующим образом:

На лёгком уровне сложности значение Factor = 0, то есть при попадании в ловушку повышение уровня стресса не происходит. На среднем уровне сложности значение Factor = 0.5, то есть при попадании ловушку стресс увеличивается на половину от значения полученного урона. На тяжёлом уровне сложности значение Factor = 0.8, то есть при попадании в ловушку увеличение стресса составляет 80% от полученного персонажем урона.

Влияние уровня сложности на зелья и магию реализовано на основе похожих принципов.

**4.4 Описание классов**

**Класс TPlayer**

Класс TPlayer является наследником класса TImage. Предназначен для хранения параметров игрового персонажа.

Описание класса:

* Поля
* Health – поле, хранящее текущее значение очков здоровья у персонажа
* Mana – поле, хранящее текущее значение очков магии у персонажа
* Stress – поле, хранящее количество очков стресса у персонажа
* Luck – поле, хранящее количество очков удачи у персонажа
* isRithual – поле, хранящее признак принадлежности персонажа к знаку Ритуал
* Orientation – поле, хранящее текущее направление персонажа
* isFirstMeetWithHerma – поле, хранящее признак первой встречи с игровым персонажем Хермеус Мора
* Методы
* Create – конструктор класса

**Класс TBottle**

Класс TBottle является наследником класса TImage. Предназначен для отображения иконок зелий и хранения их эффективности.

Описание класса:

* + Поля
* Number – поле, хранящее количество зелий данного вида у персонажа
  + Методы
* UpdateInf – данный метод используется для вывода на экран текущего количества зелий в инвентаре
* HealAbility – возвращает эффективность восстанавливающей способности зелья, в зависимости от уровня сложности
* Create – конструктор класса

**Класс TMagic**

Класс TMagic является наследником класса TImage. Предназначен для обеспечения магии в игре, хранения эффективности и стоимости магии, а также расчёта данных параметров в зависимости от уровня сложности.

Описание класса:

* + Поля
* Number – поле, хранящее количество зелий данного вида у персонажа
  + Методы
* Create – конструктор класса

**Класс TTrap**

Предназначен для реализации объекта «ловушка» в игре, обеспечивает хранение и расчёт таких игровых параметров, как урон ловушки.

Описание класса:

* + Поля
* BaseDamage – поле, хранящее урон, наносимый ловушкой
* Factor – поле, предназначенное для реализации одного из аспектов игровой механики, а именно увеличения стресса при попадании персонажа в ловушку.
* StressAdditing – поле, хранящее значение, которое добавляется к значению стресса персонажа при попадании его в ловушку
  + Методы
* Create – конструктор класса
* RetCurrentDamage – метод, предназначенный для определения урона, который нанесёт ловушка персонажу, попавшему в неё.

Примечание:Все пользовательские типы данных и использованные файлы описаны в Приложении Б.

**5. Тестирование и проверка работоспособности программного средства**

Процессу тестирования подвергались версии проекта различной степени завершённости. Если какая-то из проблем, решённая в предыдущих версиях, не описана в следующих, то она была решена и больше не возникала. В случае, если программное средство не прошло какой-либо из тестов, то описание проблемы и её решения можно найти ниже таблицы тестов.

**5.1 Таблица тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер тестируемого функционального требования | Описание цели тестирования | Ожидаемый результат | Полученный результат |
| 1 | Проверка отображения на экране окна приложения | Окно приложения меню отобразилось после запуска программы | Соответствует ожидаемому |
| 2 | Проверка получения данных о нахождении сервера при запуске программы | При открытии приложения сервер присылает свой IP адрес в локальной сети | Соответствует ожидаемому |
| 3 | Проверка установления соединения с сервером при запуске программы | При запуске программы устанавливается TCP-соединение с сервером | Соответствует ожидаемому |
| 4 | Проверка получения ключа для шифрования от сервера | После установления соединения с сервером клиенту присылается ключ для шифрования данных | Соответствует ожидаемому |
| 5 | Проверка отправки зашифрованного запроса на регистрацию при нажатии на кнопку регистрация и заполненных полях логина и пароля | Клиент отправляет серверу зашифрованный запрос на регистрацию | Соответствует ожидаемому |
| 6 | Проверка отправки шифрованного запроса на авторизацию при нажатии на кнопку авторизации и заполненных полях логина и пароля | Клиент отправляет серверу зашифрованный запрос на авторизацию | Соответствует ожидаемому |
| 7 | Проверка отправки шифрованного запроса на получение файла | Клиент отправляет серверу зашифрованный запрос на получение файла | Соответствует ожидаемому |
| 8 | Проверка отправки шифрованного запроса на передачу файла | Клиент отправляет серверу зашифрованный запрос на передачу файла и передаёт файл в зашифрованном виде | Соответствует ожидаемому |
| 9 | Проверка отображения списка текущих файлов пользователя на сервере | Пользователю виден список всех его файлов, которые загружены на сервер | Соответствует ожидаемому |
| 10 | Проверка шифрования данных и их расшифровки | Клиент корректно расшифровывает и зашифровывает передаваемые данных и запросы | Соответствует ожидаемому |
| 11 | Проверка закрытия соединения при нажатии на кнопку «Выход» | При нажатии кнопки «Выход» осуществляется разрыв соединения и деавторизация пользователя | Соответствует ожидаемому |
| 12 | Проверка корректной обработки ошибок с выводом пользователю информации о них | Программа ведёт себя корректно при возникновении ошибок и уведомляет пользователя текстовыми сообщениями с содержанием ошибки | Соответствует ожидаемому |
| 13 | Проврка возможности попытаться восстановить потерянное соединение | В случае потери соединения приложения производит попытки переподключения | Соответствует ожидаемому |

**6. Руководство по установке и использованию**

Игровой проект Escape From Cave будет распространяться в виде архива с расширением .rar или .zip, поэтому для установки его на ваш компьютер вам потребуется программа архиватор, способная распаковать файлы данных форматов. Примером такой программы является WinRar.

Игровой проект Escape From Cave не требователен к ресурсам системы, а потому может быть запущен и использован на компьютере следующей конфигурации:

ОС:Windows XP, Windows -7 32,64 bit  
Процессор и память: Процессор с тактовой частотой 2.24 ГГц, 2 Гб ОЗУ  
Графика: 32 Мб графической памяти

Для запуска игрового проекта, необходимо открыть файл Escapefromcave.exe, для быстрого доступа рекомендуется создать ярлык на рабочем столе. Вид окна может отличаться, в зависимости от используемой операционной системы и цветовой схемы.

При открытии приложения на экране должно появится окно следующего вида:

**Главное меню**

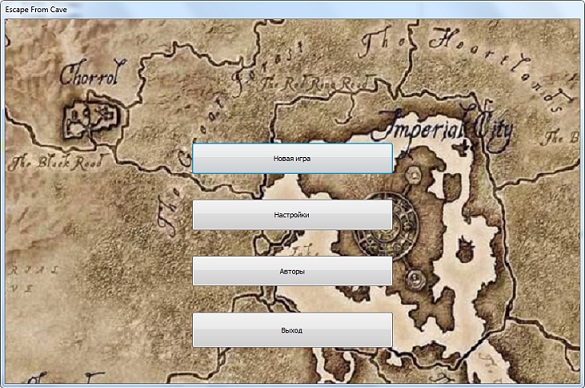


Рис 6.1 – Скриншот окна главного меню

1. Кнопка «Новая игра»  
 При нажатии на кнопку «Новая игра» произойдёт переход к окну настройки параметров персонажа для подготовки его к игре

2. Кнопка «Настройки»  
 При нажатии на кнопку настройки произойдёт открытие окна «Настройки», в котором при необходимости вы можете изменить текущую схему управления игровым персонажем.

3. Кнопка «Авторы»  
 При нажатии на кнопку «Авторы» произойдёт открытие диалогового окна, с информацией об игровом проекте.  
4. Кнопка «Выход»  
 При нажатии на кнопку выход произойдёт закрытие программного средства.

**Окно настроек**

Окно «Настройки» имеет следующий вид:

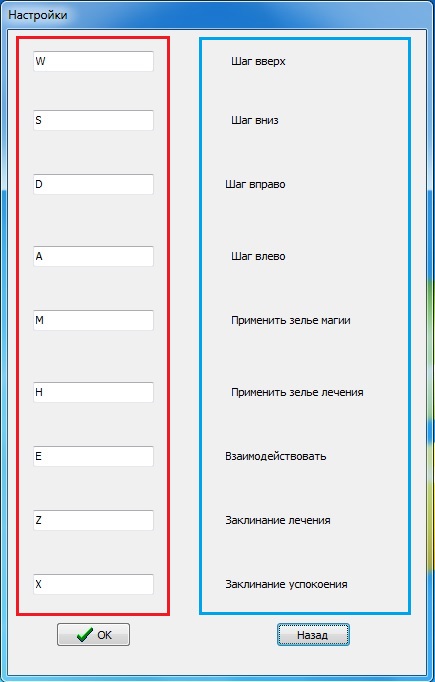


Рис. 6.2 – Скриншот окна настроек

Красным цветом выделены поля ввода. При открытии в них содержится текущее состояние функциональных клавиш. Синей областью выделены описания, соответствующие действию, которая каждая из назначенных клавиш выполняет.

При необходимостью изменения текущего состояния настроек необходимо ввести в поле ввода английский литерал(регистр не важен), для задания новой клавиши управления. Для подтверждения своего выбора нажмите кнопку «Ок». В случае успешного изменения настроек будет выведено сообщение следующего вида:

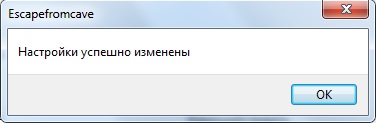


Рис 6.3 – Скриншот сообщения об успешном изменении настроек

Для выхода из меню изменения настроек нажмите кнопку «Назад». После чего откроется окно главного меню.

**Окно настроек персонажа**

Окно настроек персонажа имеет следующий вид:

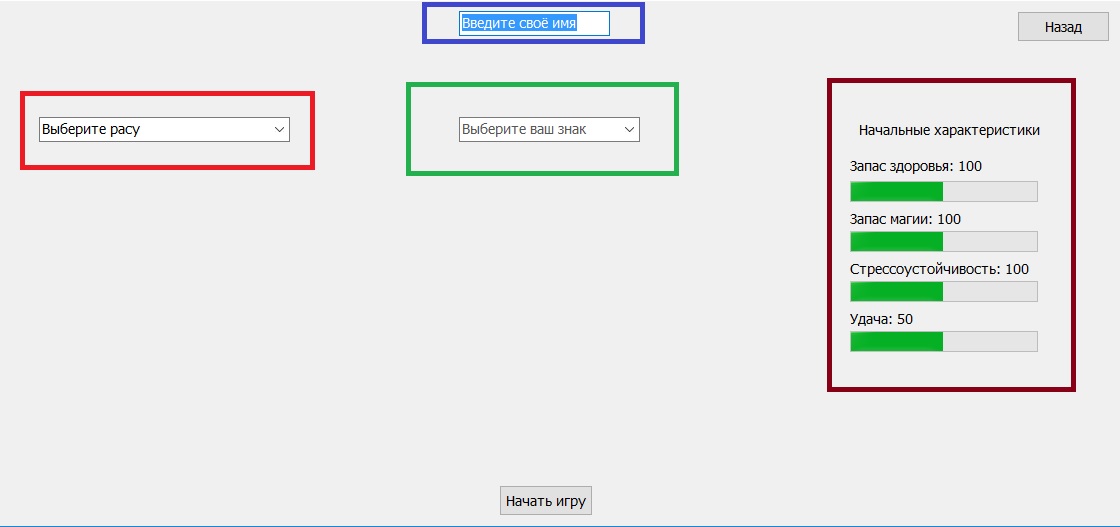


Рис 6.4 – Скриншот окна настроек персонажа

Красным цветом выделено меню выбора расы. Всего в игре представлено 7 игровых рас: Имперец, Бретонец, Орк, Каджит, Аргонианен, Эльф, Тёмный Эльф. Каждая из игровых рас имеет свои базовые параметры.

Зелёным цветом выделено меню выбора знака рождения. Всего в игре представлено 5 знаков: Воин, Маг, Вор, Риуал и Остап. Выбор знака может как усилить, так и ослабить некоторые стартовые параметры персонажа.

Коричневым цветом выделена область отображения игровых характеристик. Изменённые значения отображаются после выбора расы и знака рождения.

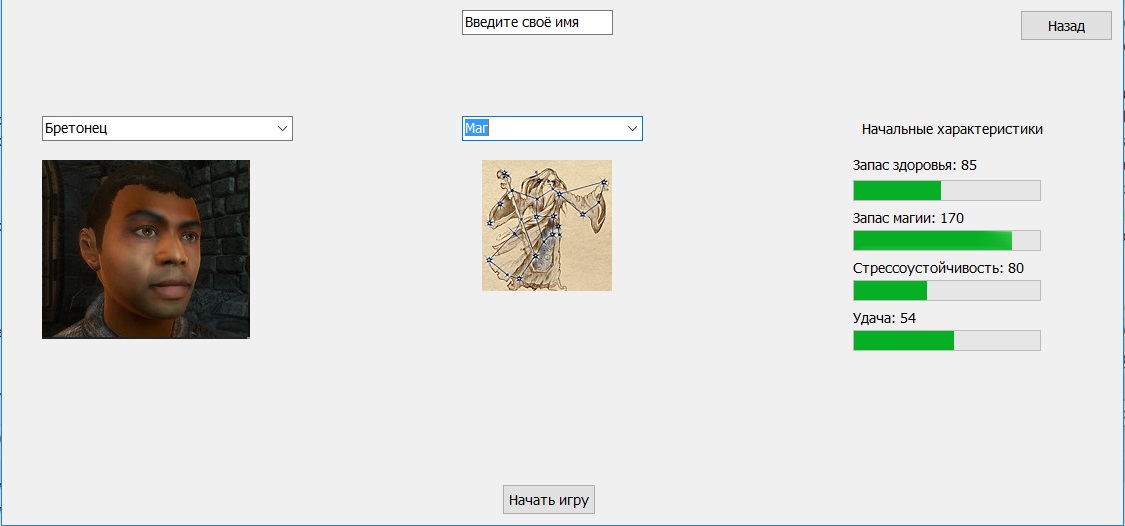


Рис. 6.5 – показ стартовых параметров персонажа

Синим цветом выделено поле для ввода имени.

В случае, необходимости вернуться в окно главного меню присутствует кнопка «Назад». При её нажатии произойдёт закрытие текущего окна и возвращение в окно главного меню.

Для перехода к игре необходимо нажать на кнопку «Начать игру». При этом, если настройка персонажа завершена не до конца, то будет выведено соответствующее сообщение.

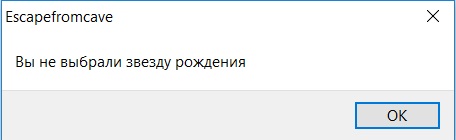


Рис. 6.6 – Скриншот сообщения, при невыбранной звезде рождения

В случае, если настройка персонажа завершена и нажата кнопка «Начать игру», то произойдёт открытие окна выбора уровня сложности.

**Окно выбора уровня сложности**

Окно выбора уровня сложности имеет следующий вид:

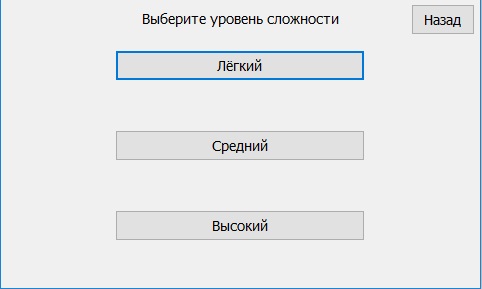


Рис. 6.7 – Скриншот окна выбора уровня сложности

При нажатии на одну из кнопок, происходит выбор соответствующего уровня сложности и начинается игра. При необходимости вернуться к окну настройки персонажа, нажмите кнопку «Назад».

Подробнее об уровнях сложности читайте в разделе «Основы игровой механики»

**Окно игры**

Окно игры состоит из следующих областей:

1. Область отображения лабиринта.  
 В данной области отображается текущее состояние лабиринта и положение персонажа в нём. Для перемещения персонажа в лабиринте используются функциональные клавиши, назначенные в меня настроек.

2. Область отображения текущего состояния персонажа

3. Область отображения количества зелий и иконок зелий, а также иконок магии

4. Область вывода сюжетной информации

В случае необходимости завершить прохождение игры присутствует кнопка «Выход в главное меню». В случае выхода, результаты текущей сессии не будут сохранены.

**Окно загадок**

При попадании игрового персонажа в особую область на игровой локации будет открыто окно загадок. При первом открытии окно будет иметь следующий вид:

Рекомендуется внимательно ознакомиться с отображённым текстом, так как он может дать полезную информацию, которая может пригодиться при решении загадок.

При следующих открытиях в окно будет иметь следующий вид:

Красным цветом выделена область, в которую будет выводиться текст загадки.

Зелёным цветом выделено поле для ввода ответа на загадку. Прописные и строчные буквы в ответе являются эквивалентными, то есть ответ можно дать как: «Снег», «СНЕг», «снег».

Синим цветом выделена кнопка «Подтвердить ответ», нажатие которой приведёт к обработке полученного ответа и выводу результата на экран. В случае дачи неправильного ответа игровой персонаж погибает.

Коричневым цветом выделена область, в которую выводится оставшееся время. Всего пользователю даётся 60 секунд на дачу ответа, если время истечёт, то игровой персонаж погибнет.

**Окно взлома**

При взаимодействии игрового персонажа с сундуком может активироваться игровое событие взлом, после чего откроется соответствующее окно.

Красным цветом выделено поле ввода числового пароля. Длина пароля – 6 цифр. Повторяющихся цифр нет.

Зелёным цветом выделена кнопка, нажатие которой приводит к вводу пароля и проверке его правильности.

Синим цветом выделены анимированные элементы сундука. В зависимости от правильности активируется или дезактивируется определённое количество элементов с той или иной стороны.

Количество активированных элементов с левой стороны равно количеству угаданных чисел, стоящих на своих местах.

Количество активированных элементов с правой стороны равно количеству угаданных цифр без совпадения с их позициями в пароле.

**Заключение**

В ходе выполнения курсовой работы были проанализированы и рассмотрены значительные объёмы текстовой, графической и видеоинформации по теме анализа, планирования и конструирования программных средств. Были изучены примеры коммерчески успешных игровых проектов, проведено ознакомление с дневниками разработчиков данных проектов. Изучены их достоинства и недостатки.

На основании анализа полученной информации был сформулирован ряд общих требований к программному средству, выполнение которых должно было обеспечить пользователю комфортное использование игрового проекта.

Сформулирован ряд функциональных требований к программному средству, выполнение которых обеспечило бы игровому проекту достаточно широкие возможности. Был проведён опрос потенциальных пользователей, по результатам которого список функциональных требований был расширен.

Выполнен анализ различных существующих вариантов решения, на основании которых выдвинуты собственные решения, использованные в данной курсовой работе. Следует отметить, что данные решения могут быть использованы в других игровых проектах подобного жанра.

Произведено конструирование программного средства и реализация спроектированных алгоритмов в программном коде. Некоторые алгоритмы были подвергнуты переработке и включены в программное средство в улучшенном варианте. Реализован графический пользовательский интерфейс и графически отображён игровой мир, добавлено музыкальное сопровождение. Создано множество интересных и разнообразных игровых локаций.

Проведено тестирование программного средства на предмет ошибок и недочётов, в ходе которого были выявлены и устранены существовавшие проблемы. Проведён анализ ошибок, с целью использования данного опыта в будущих проектах.

Создано руководство пользователя, для обеспечения его информацией, необходимой для удобного использования игрового проекта. Помимо описания функционала, рассмотрены некоторые аспекты игровой механики и описаны основные приёмы построения локаций, что позволит пользователям создавать свои собственные игровые локации, текстуры, загадки и истории и включать их в игру.

Данный проект может быть улучшен, путём добавления в него новых возможностей, таких как сохранение игры, новых элементов игровой механики и сюжетный линий.

**Список использованной литературы**

[1] Фаронов. В.В. Delphi Программирование на языке высокого уровня/ В.В.Фаронов – СПб. : Питер, 2004. – 639 с.

[2] Доусон. М. Изучаем С++ через программирование игр/ М.Доусон – СПб. : Питер, 2016. – 353 с.

[3] Серебряная Л.В. учебно-методическое пособие по курсу «Структуры и алгоритмы обработки данных» / Л. В. Серебряная, И.М. Марина. – Минск: БГУИР, 2013. – 50 с.

[4] Ахо А. Структуры данных и алгоритмы/ А. Ахо, Д. Хопкрофт, Д. Ульман. – Москва: Издательский дом «Вильямс», 2003. – 382 с.

[5] Старинные загадки [Электронный ресурс]. – Электронные данные. – Режим доступа: http://folklor.igraemsdetmy.ru/starinnye-zagadki/

[6] The Elder Scrolls IV – Oblivion [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://wikipedia.org/>.

[7] The Elder Scrolls V – Skyrim [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://wikipedia.org/>.

[8] Warcraft 3 Frozen Throne [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://wikipedia.org/>.

[9] Warcraft 3 Frozen Throne [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://wikipedia.org/>.

[10] Sherlock Holmes The Devil’s Daughter [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://wikipedia.org/>.

[11] The Elder Scrolls IV – Oblivion [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://steamcommunity.com/>.

[12] TES V Дневники разработчиков [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.youtube.com/>.

[13] Спрайтовая графика [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://wikipedia.org/>.;

**Приложение А**

**Описание файлов и типов, используемых в программном средстве**

**Типы данных**

|  |  |  |
| --- | --- | --- |
| Наименование типа | Описание типа | Назначение типа |
| TSettings | TSettings = record  UpKey, DownKey, RightKey, LeftKey, ManaKey, HealthKey, EnterractKey, hMagicKey,sMagicKey:integer;  end; | Предназначен для использования в переменных, осуществляющих работу с настройками |
| TFileOfSettings | TFileOfSettings = file of TSettings; | Предназначен для описания файлов, работающих с настройками |
| TDifficult | TDifficult = (Easy,Medium,Hard); | Предназначен для реализации уровней сложности |
| TEnigma | TEnigma = record  question:string[100];  Answer:string[15];  end; | Предназначен для использования в переменных, работающих с загадками |
| EnigmaPtr | EnigmaPtr = ^TEnigmaRing; | Указатель на элемент кольцевой структуры загадок |
| TEnigmaRing | TEnigmaRing = record  Next:EnigmaPtr;  Enigma:TEnigma;  Prev:EnigmaPtr;  end; | Предназначен для использования при создании кольцевой структуры загадок |
| FEnigma | FEnigma = file of TEnigma; | Предназначен для использования в файлах, хранящих загадки |
| TNode | TNode = record  Text1:string[255];  Text2:string[255];  Name:string[20];  end; | Предназначен для использования в переменных, работающих с историями |
| FNode | file of TNode | Предназначен для файлов, хранящих истории |
| TNodeElemPtr | TNodeElemPtr = ^TNodeElem; | Указатель на элемент кольцевой структуры историй |
| TNodeElem | TNodeElem = record  Next:TNodeElemPtr;  Story:TNode;  Prev:TNodeElemPtr;  end; | Предназначен для использования в кольцевой структуре историй |
| TOrientation | TOrientation = (Up,Down,Left,Right) | Предназначен для реализации механизма работы с направлениями движения |
| TEndOfGame | TEndOfGame = (Escape,Death,Madness) | Предназначен для определения окончания игры |
| TMap | TMap = array of array of Char; | Предназначен для использования в переменных, хранящих подуровни |
| TMapHolder | TMapHolder = record  NextMap:TMapPointer;  Map:TMap;  PrevMap:TMapPointer;  end; | Предназначен для использования в списке подуровней |
| TMapPointer | TMapPointer = ^TMapHolder | Указатель на элемент списка подуровней |

**Файлы**

|  |  |  |
| --- | --- | --- |
| Наименование файла | Тип файла | Назначение файла |
| Settings.dat | File of TSettings | Хранит коды клавиш, отвечающих за управление |
| StoryOfAsvalt.dat | File of TNode | Хранит сюжетную информацию, подаваемую при помощи записок |
| Enigma.dat | File of TEnigma | Хранит загадки и ответы на них |
| FirstWords.dat | File of TEnigma | Хранит приветственное слово |
| Карты | TextFile | Хранят игровые локации |

**Приложение Б**

**Исходный код программного средства**