**Федеральное государственное образовательное бюджетное учреждение  
высшего образования**

**«Финансовый университет при Правительстве Российской Федерации»**

**КОЛЛЕДЖ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ**

**ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем**

**Группа: 3ПКС-115**

**УТВЕРЖДАЮ**

**Председатель цикловой комиссии**

**программирования и баз данных**

**( ) Пестов А. И.**

**\_\_\_\_.\_\_\_\_. 2018 г.**

**ПРОЕКТ КУРСОВОЙ**

**На тему: Реализация графического пользовательского интерфейса и системы диалогов в играх на языке C++**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**Руководитель курсового проекта**

**( ) E. Д. Перхункова**

**Исполнитель курсового проекта**

**( ) М. Д. Шамшурин**

**Оценка за проект:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_.\_\_\_\_.2018 г.**

**2017**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc502003388)

[1 ОБЩИЙ РАЗДЕЛ 4](#_Toc502003389)

[1. 1 Системные требования. 4](#_Toc502003390)

[1. 2 Характеристика системы программирования. 4](#_Toc502003391)

[2 ТЕХНИЧЕСКИЙ РАЗДЕЛ 6](#_Toc502003392)

[2. 1 Постановка задачи 6](#_Toc502003393)

[2. 2 Логическая модель 8](#_Toc502003394)

[2. 3 Тестирование программы 9](#_Toc502003395)

[2. 4 Анализ результатов тестирования 21](#_Toc502003396)

[2. 5 Надежность программного обеспеченья 21](#_Toc502003397)

[3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЮ ПРОГРАММЫ 23](#_Toc502003398)

[3. 1 Руководство программиста 23](#_Toc502003399)

[3.2 Руководство пользователя 24](#_Toc502003400)

[ЗАКЛЮЧЕНИЕ 26](#_Toc502003401)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc502003402)

[ПРИЛОЖЕНИЕ А 28](#_Toc502003403)

ВВЕДЕНИЕ

На сегодняшний день в нашей жизни имеют место компьютерные игры. Первые игры были более простые – аркадные игры, которые в свою очередь не имели такового сюжета. Они лишь ставили перед игроком простую цель. Например, сбить несколько самолётов, не задевая огнём другие. Но сегодня, как правило игры (если это большой проект), имеют сюжетную составляющую. Он нередко передаётся через диалог между двумя или несколькими игровыми персонажами. Так же обязательной частью игры является постоянная «связь» с пользователем – взаимодействие персонажей и игрока, которая отличает компьютерную игру от какого-либо фильма.

Основная проблема – отображение диалога в игре. Множество продуктов (игр) имеет диалоги, будь то 3D или 2D графика. Они позволяют передать игроку сеттинг игры, с помощью отрисовки персонажа, нарисованного в его стиле. Помогает понять игроку суть происходящего, не нарушая атмосферы игры. Так же диалог может связывать игру и игрока по средству выбора варианта ответа на поставленный вопрос, как в игре «Бесконечное лето», где геймплей представляет ответы на вопросы или выбор определенных действий.

Актуальность вопроса заключается в помощи начинающему разработчику корректно отобразить диалог персонажей и взаимодействие с игроком. Это может помочь в передаче сюжета путём разговора или его изменение, к примеру, заменой имени персонажа именем пользователя, после его введения.

1. ОБЩИЙ РАЗДЕЛ

1. 1. Системные требования.

Минимальные системные требования к электронной вычислительной машине (ЭВМ):

* Операционная система: Windows 7/8/10;
* Процессор: Intel Celeron или аналогичный AMD;
* Частота: 1500 МГЦ;
* Оперативная память (ОЗУ): 512МБ;
* Жесткий диск: 100 МБ;
* Видеокарта: видеокарта с поддержкой OpenGL 3.0;
* Память видеокарты: 256 МБ и выше;
* Клавиатура.

Рекомендуемые системные требования к электронной вычислительной машине (ЭВМ):

* Операционная система: Windows 7/8/10;
* Процессор: Intel CoreI3 или аналогичный AMD;
* Частота: 2400 МГЦ;
* Оперативная память (ОЗУ): 1ГБ;
* Жесткий диск: 100 МБ;
* Видеокарта: видеокарта с поддержкой OpenGL 3.0;
* Память видеокарты: 512 МБ и выше;
* Клавиатура и мышь.

1. 2. **Характеристика системы программирования.**

Для написание курсового проекта был выбран язык программирования С++, так как имеет ряд определенных качеств, присущих этому проекту. Как известно, что программа (среда разработки) Visual Studio является наиболее популярной для написания программ на языке C++.

В C++ для «рисования» графики используется графическая библиотека Open Graphics Library (OpenGL), которая является низкоуровневой. В своём курсовом проекте я использовал библиотеку, основанную на OpenGL – Simple and Fast Multimedia Library (SFML), которая является высокоуровневой и работа с картинками более удобная и простая, чем в OpenGL. Она позволяет работать не только с прорисовкой картинок или анимации, а также работой со звуком, текстом или интернетом (сетями). Это существенно облегчает процесс для создания игры. К примеру, можно использовать сочетание анимации и звукового сопровождения для имитации разговора персонажа. То есть, пока играет звуковая дорожка с озвученным текстом показывать анимацию говорящего персонажа. Как только звуковая дорожка заканчивается анимацию остановить и заменить анимацией молчащего персонажа.

2. ТЕХНИЧЕСКИЙ РАЗДЕЛ

2. 1. Постановка задачи

Перед программистом стоит задача – передать диалог между двумя или несколькими персонажами, или персонажа с пользователем (например, узнать его имя). Пользователь не сможет понять, что происходит в игре, если он ничего не увидит. Появляется еще проблема, которая заключается в отрисовке персонажа(ей) на экране монитора. Пользователь не сможет никак взаимодействовать с игрой, если у него не будет «пульта управления». но в нашем случае эта проблема решена, так как у пользователя есть устройства ввода, с помощью которых он может взаимодействовать с игрой. Также нам нужно иногда поддерживать диалог с пользователем, то есть спрашивать его о чем-либо или предоставлять ему какой-либо выбор, который сможет при необходимости поменять сюжетную часть. Например, если пользователю задают вопрос: «Хотите ли вы кушать?» Где под «вы» подразумевается ваш игровой персонаж, олицетворяющий вас. И если пользователь отвечает да, то его игровому персонажу даётся яблоко, с помощью которого он может утолить голод. Еще одной проблемой является запуск этой игры или же настройка разрешения экрана для более приятного отображения. Эта проблема называется формирование пользовательского интерфейса, а в нашем случае графического. Пользователь не сможет что-либо сделать, если он не будет выбирать соответствующие пункты или переключать выключатели.

Для решения первой проблемы, то есть отрисовки персонажей, используются графические библиотеки, способные прорисовывать картинку или анимацию (серию кадров с определенной частотой). Так же эта анимация должна быть динамической и могла реагировать на действия пользователя, чтобы он понимал, что игра продолжается. Для решения второй проблемы, которая заключается в поддержке диалога с пользователем, нам нужно связать нашу графику с событиями ввода. Например, при нажатии клавиши 1 или 0 мы могли выполнять булевы операции и отвечать «да» или «нет». Решение последней проблемы мы будем использовать текст, для прорисовки меню и использовать те же события ввода, для выбора или переключением между пунктами. Например, чтобы мы смогли переместить указатель на «Начать игру» и по нажатии клавиши «ENTER» выбрать его и начать.

Входной информацией в такую программу будут является нажатые клавиши, то есть их коды. После ввода нажатой клавиши будет происходить определенное действие. Например, будет выводиться какой-либо диалог и при нажатии клавиши «ENTER» будет пропущена часть, которая печатает текст и будет выводить его полностью. После завершения прочтения пользователем диалога при повторном нажатии на клавишу будет осуществлен переход к следующей части диалога.

Выходной информацией является 2D-графика, с помощью которой отображается говорящий персонаж, его слова в текстовой форме и различные графические украшения и оформления. Слова персонажа будут иметь какой-либо «вес» в сюжете. Поэтому, у игрока появляется желание просматривать сказанные персонажем слова для выбора последующих действий.

Например, у нас есть сюжет, где сеттингом является Россия конца 90-х годов. Сюжет таков, что главный герой, чьё имя Иван Гордов работает в центре по ремонту компьютеров. В один день к нему приходит странный мужчина лет 30-40 в темных непроглядных очках и просит починить Ивана его компьютер. На вопрос о том, что именно не работает компьютер этот мужчина отвечает, что он не знает и просит его починить в кратчайшие сроки. После осмотра Иван говорит, что работа не сложная и продлится она максимум 2 часа. После чего загадочный мужчина просит позвонить ему после завершения работы, на что Ваня соглашается. После просьбы мужчина удаляется не попрощавшись. Сюжет является очень простым, так как был написан только для тестирования программы. Итак, для подачи сеттинга нам понадобятся стилизованные рисунки главного персонажа – Ивана, и загадочного мужчины, так же нам понадобится рисунок телефона. После того, как рисунки будут нарисованы можно переходить к раскадровки, которая послужит анимацией.

2. 2. Логическая модель

Логическая схема не является особо сложной. Сначала мы попадаем главное меню откуда мы выбираем: запуск диалога, настройка, выход. При выборе диалога на экране рисуется анимация персонажа, который произносит текст, после, когда текст произнесен, рисуется новый персонаж «напротив него», который поддерживает разговор. Таким образом у нас получается симуляция диалога между двумя персонажами. После, в зависимости от диалога игроку предоставляется вариант ответа на поставленный вопрос и при определенном ответе будет происходить ответвление, описанное в программе, которое и будет влиять на выполнение определенного действия. Логическая модель проекта изображена на рисунке 1.



Рис. 1 - Логическая модель задачи

2. 3. Тестирование программы

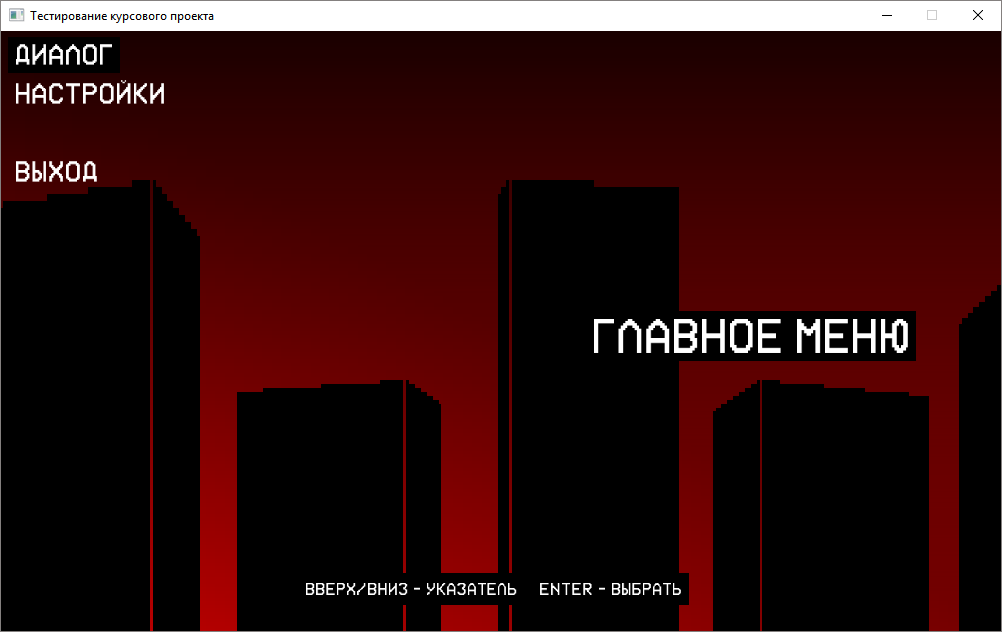


Рис. 2 – Главное меню программы

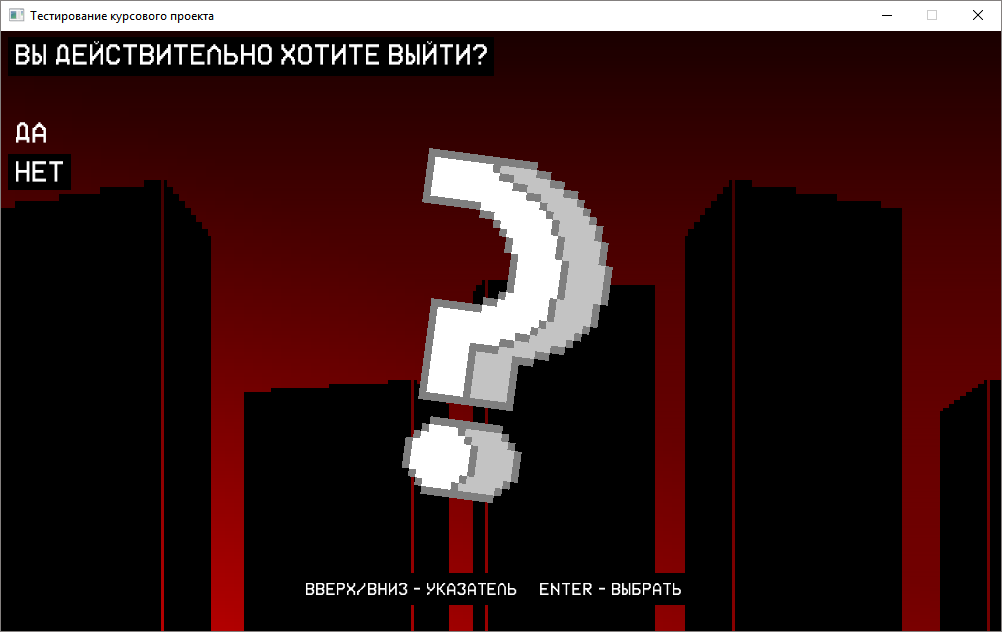


Рис. 3 – При нажатии на кнопку выход

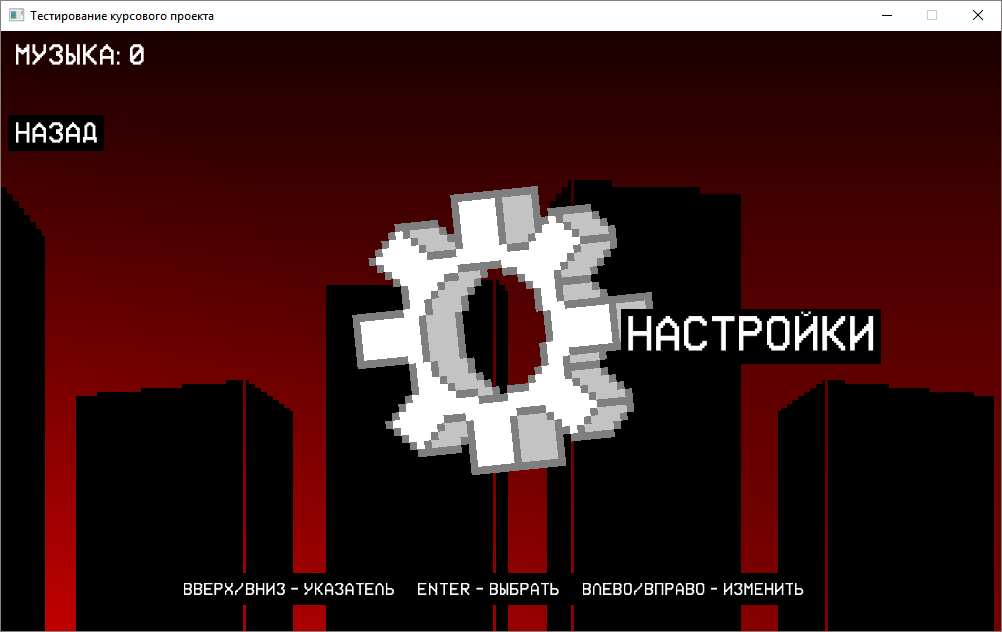


Рис. 4 – Настройки

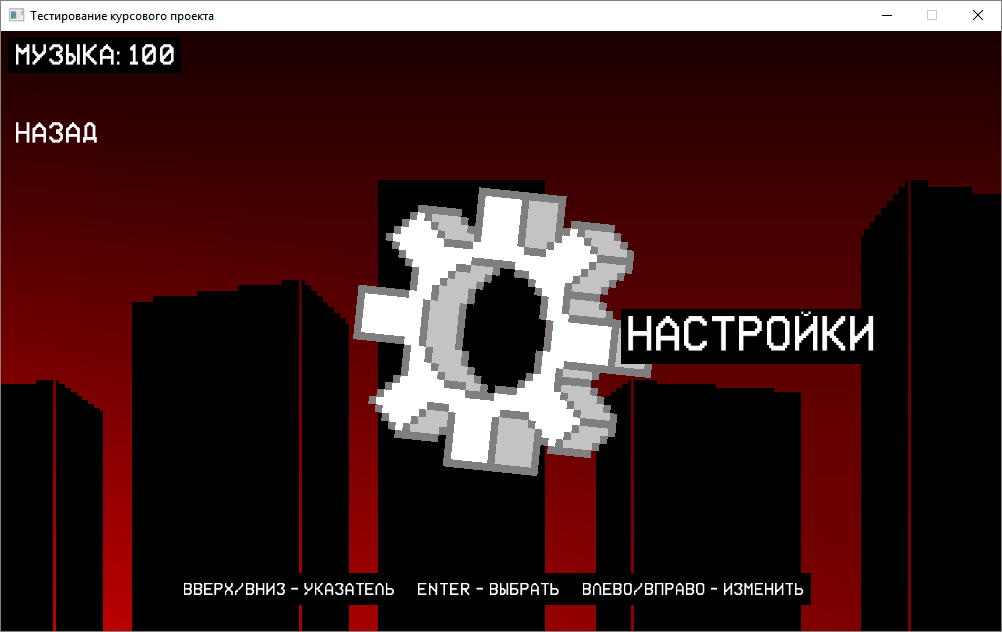


Рис. 5 – Настройки с измененным параметром уровня громкости музыки

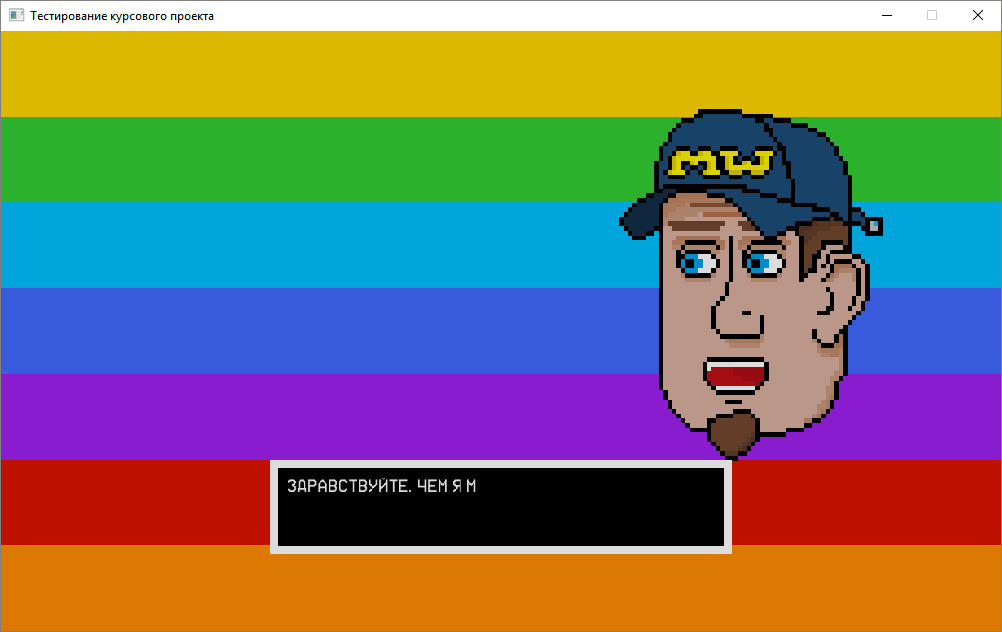


Рис. 6 – Отображение диалога (персонаж говорит и находится справа)

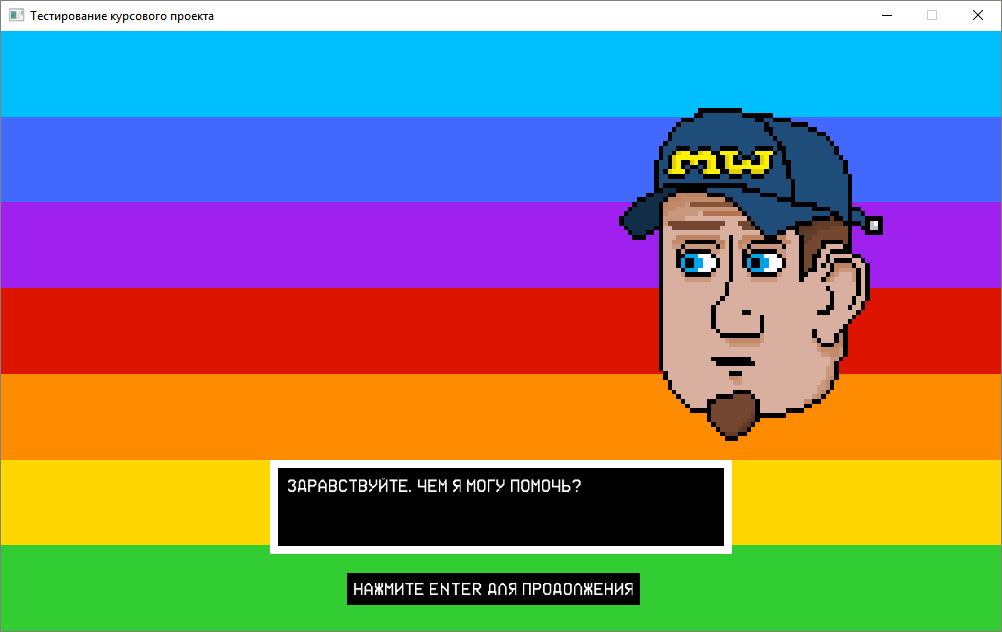


Рис. 7 – Отображение диалога (персонаж молчит и находится справа)

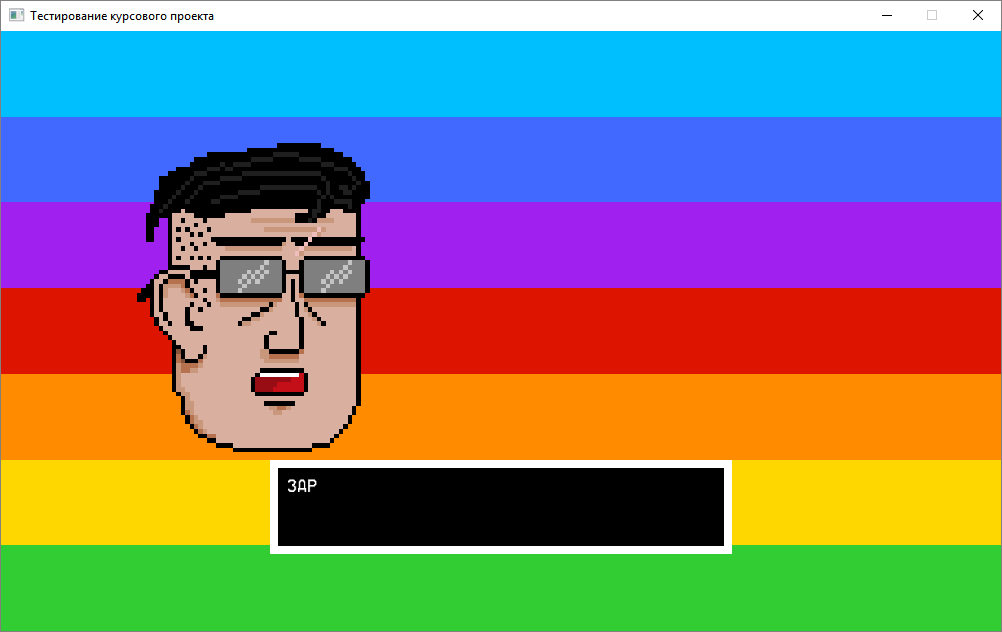


Рис. 8 – Отображение диалога (персонаж говорит и находится слева)

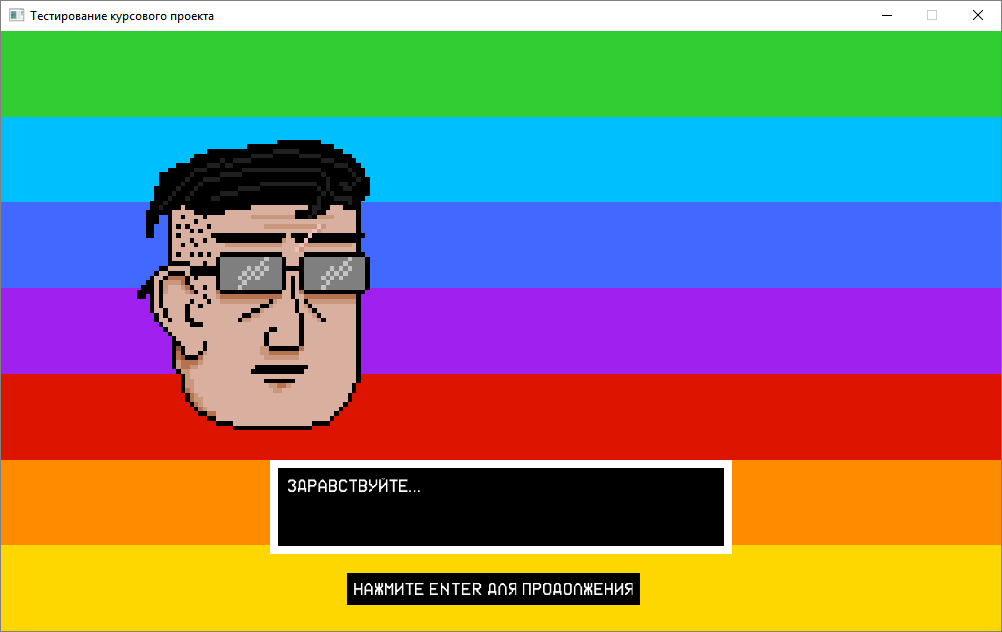


Рис. 9 – Отображение диалога (персонаж молчит и находится слева)

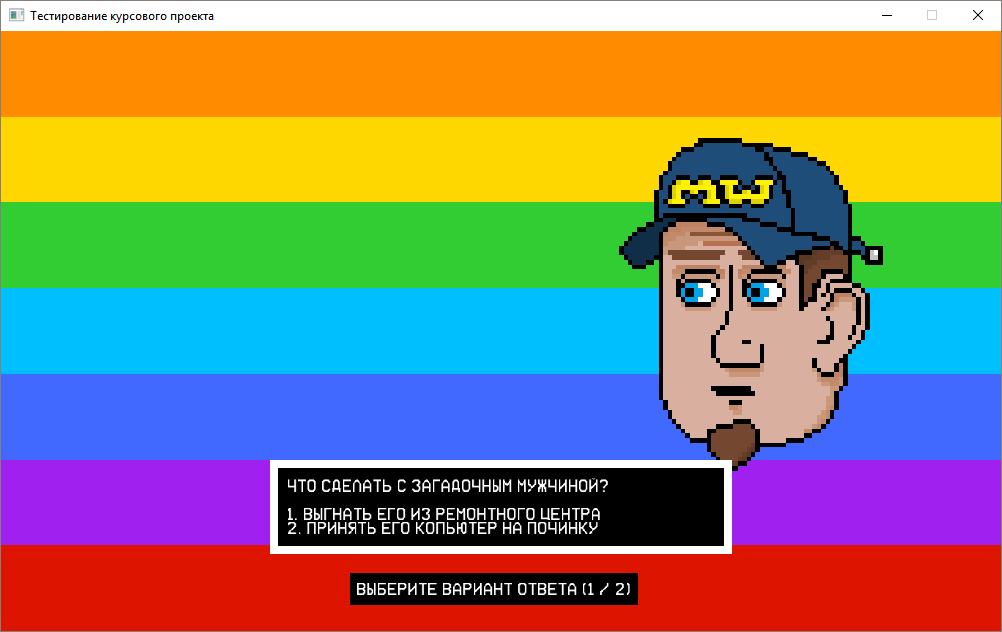


Рис. 10 – Отображение диалога (выбор действий)

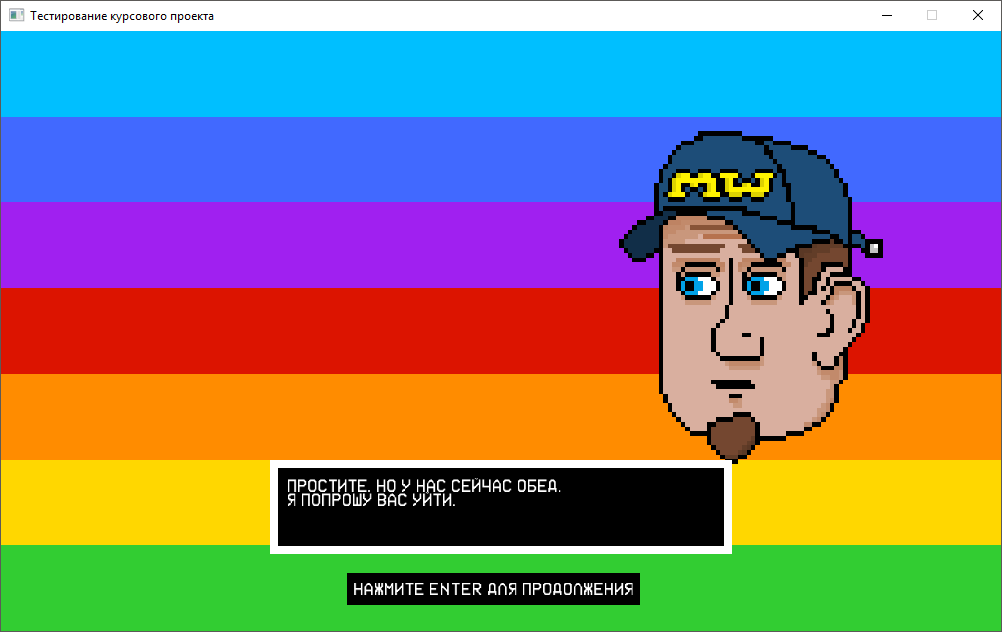


Рис. 11 – Ответ на действие игрока (1 нажатие на первый вариант)

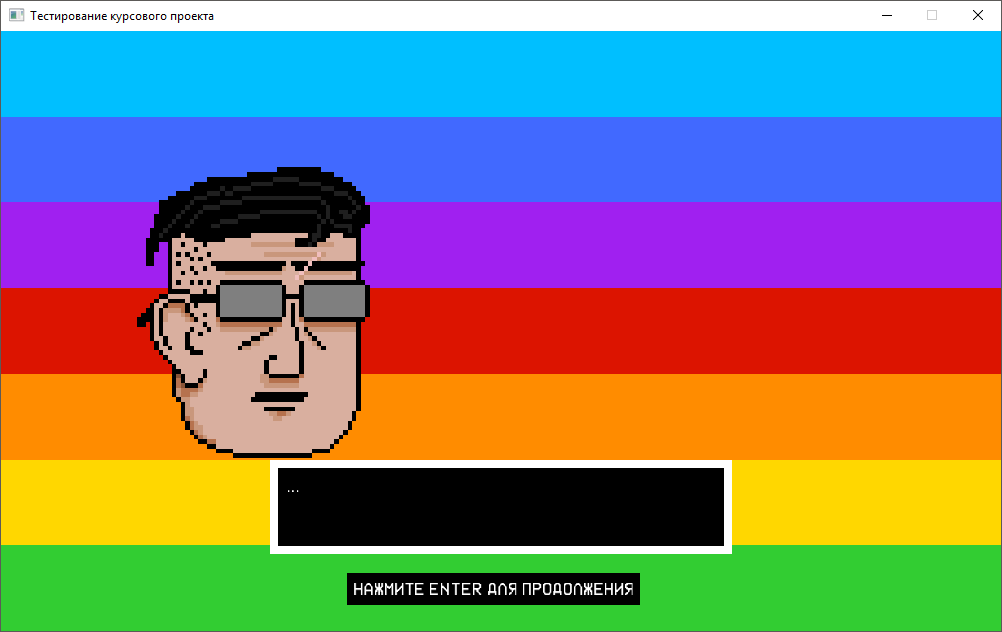


Рис. 12 – Ответ на действие игрока (1 нажатие на первый вариант)

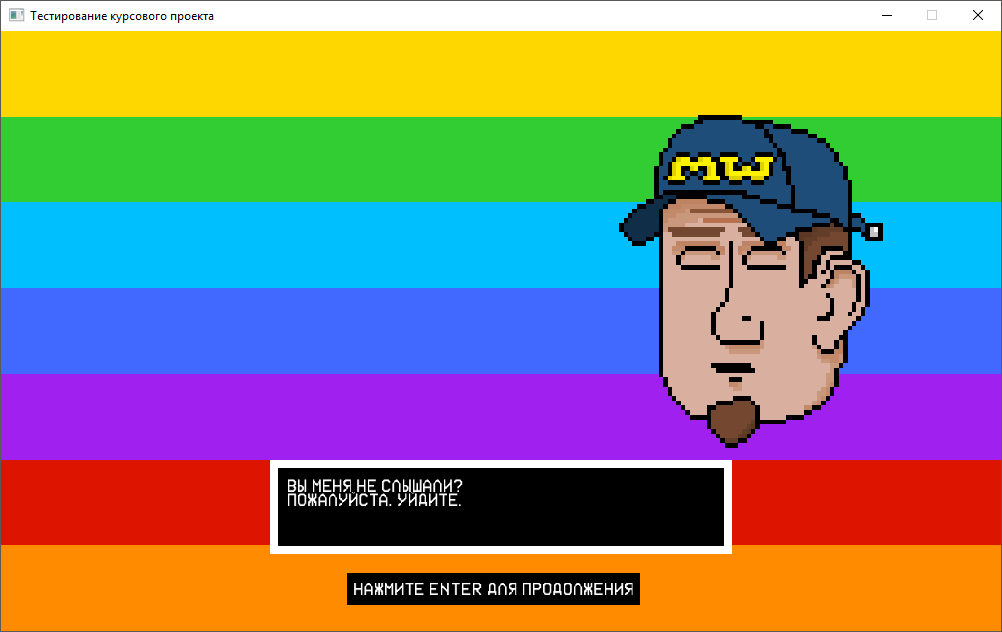


Рис. 13 – Ответ на действие игрока (2 нажатия на первый вариант)



Рис. 14 – Ответ на действие игрока (2 нажатия на первый вариант)



Рис. 15 – Ответ на действие игрока (2 нажатия на первый вариант)

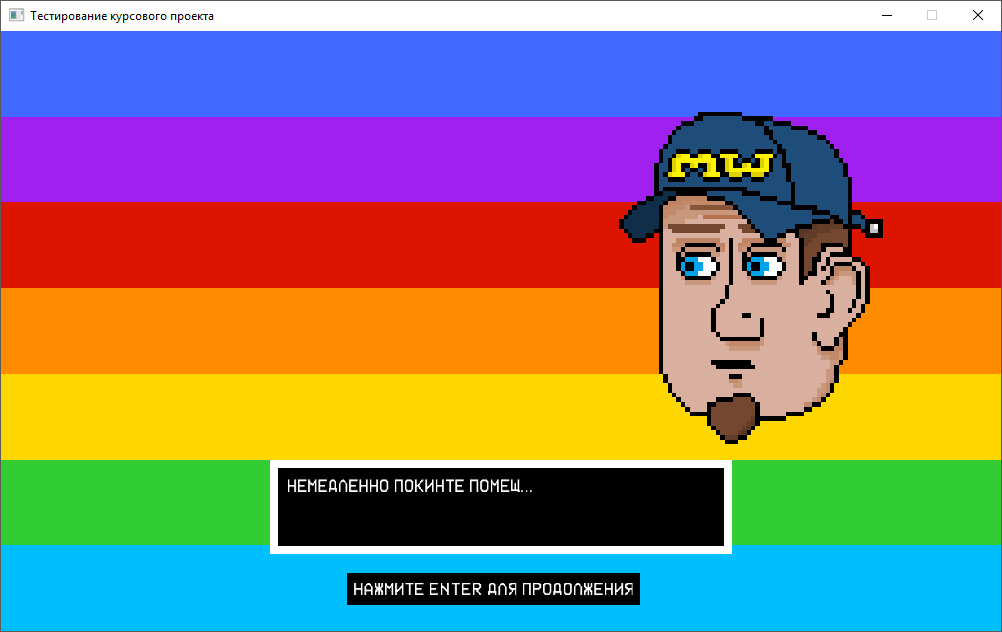


Рис. 16 – Ответ на действие игрока (3 нажатия на первый вариант)



Рис. 17 – Ответ на действие игрока (3 нажатия на первый вариант)

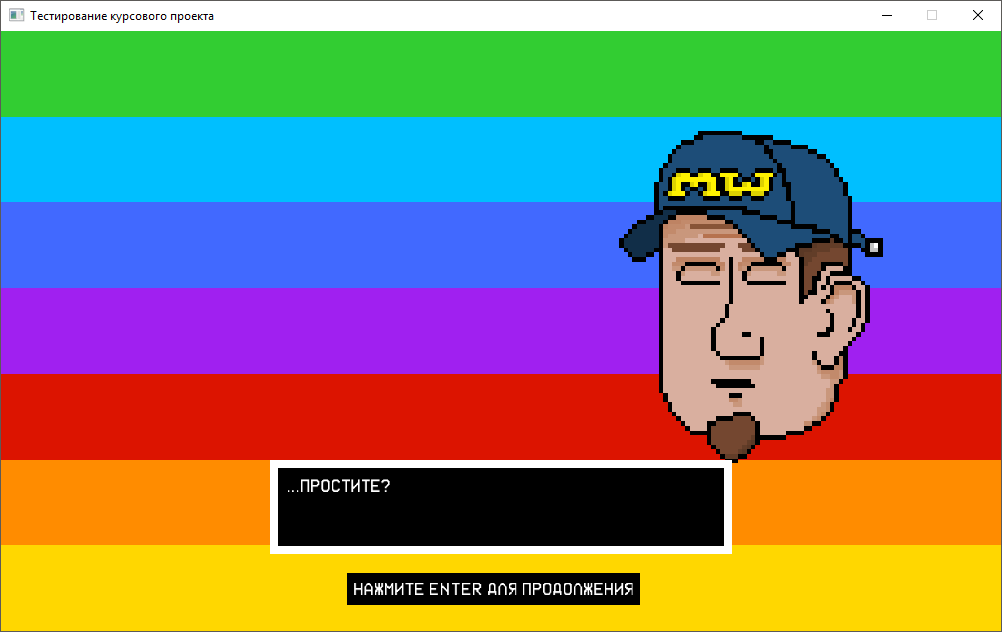


Рис. 18 – Ответ на действие игрока (3 нажатия на первый вариант)



Рис. 19 – Ответ на действие игрока (3 нажатия на первый вариант)

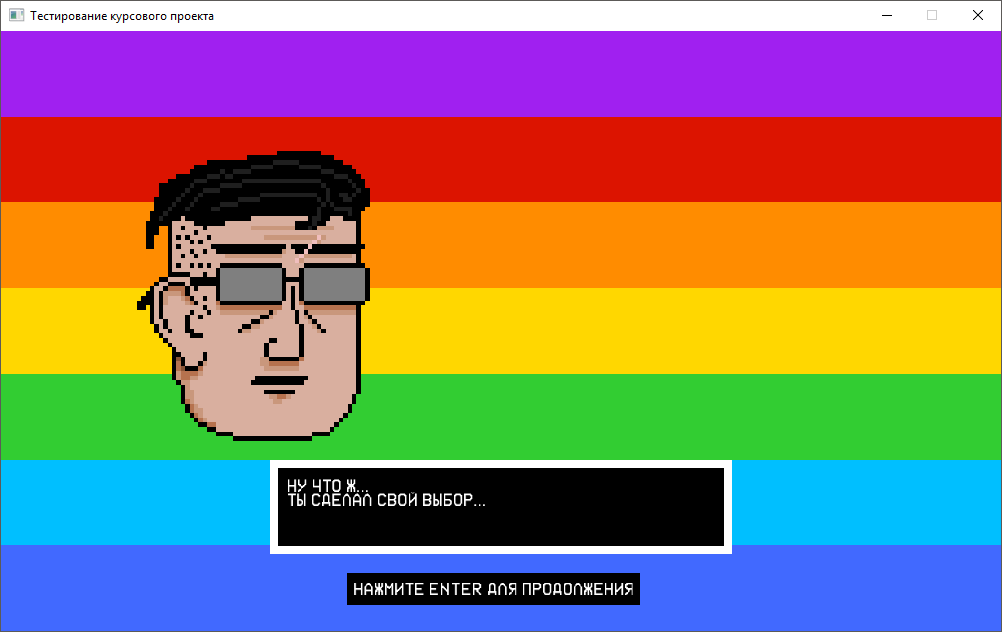


Рис. 20 – Ответ на действие игрока (4 нажатия на первый вариант)



Рис. 21 – Ответ на действие игрока (4 нажатия на первый вариант)

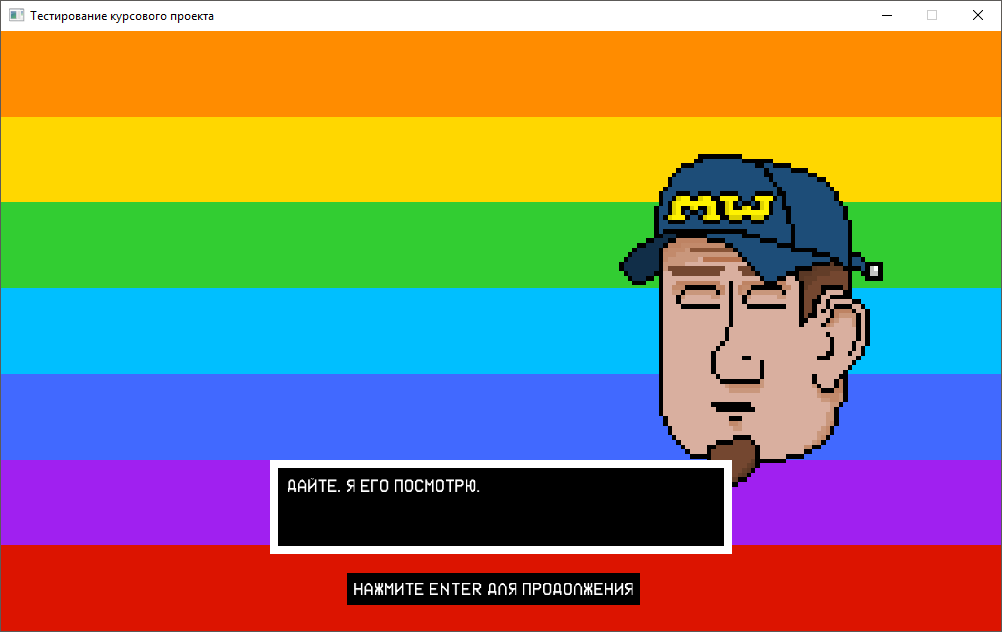


Рис. 22 – Ответ на действие игрока (1 нажатие на второй вариант)

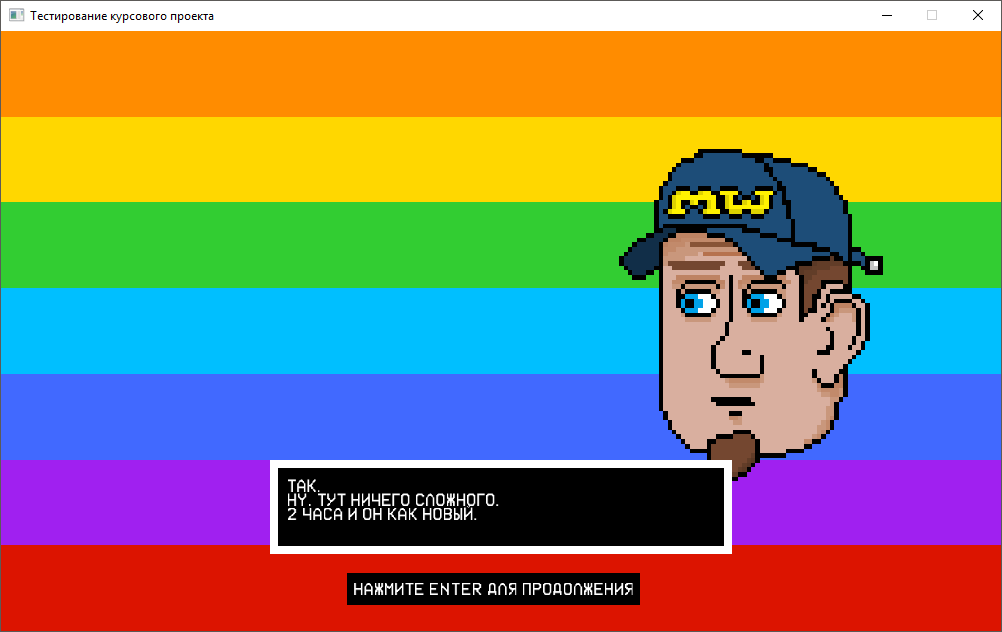


Рис. 23 – Ответ на действие игрока (1 нажатие на второй вариант)

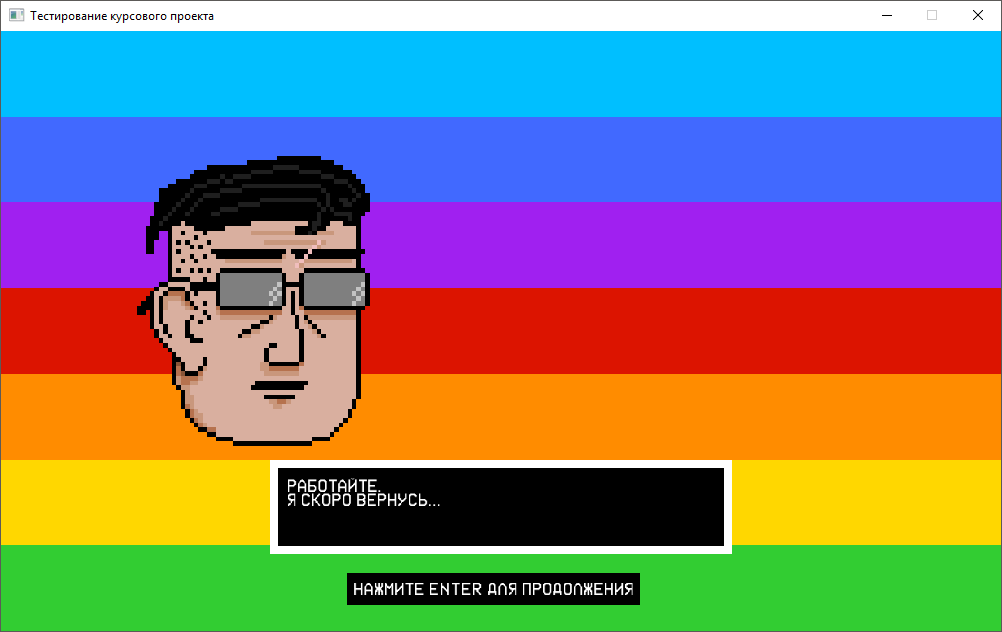


Рис. 24 – Ответ на действие игрока (1 нажатие на второй вариант)

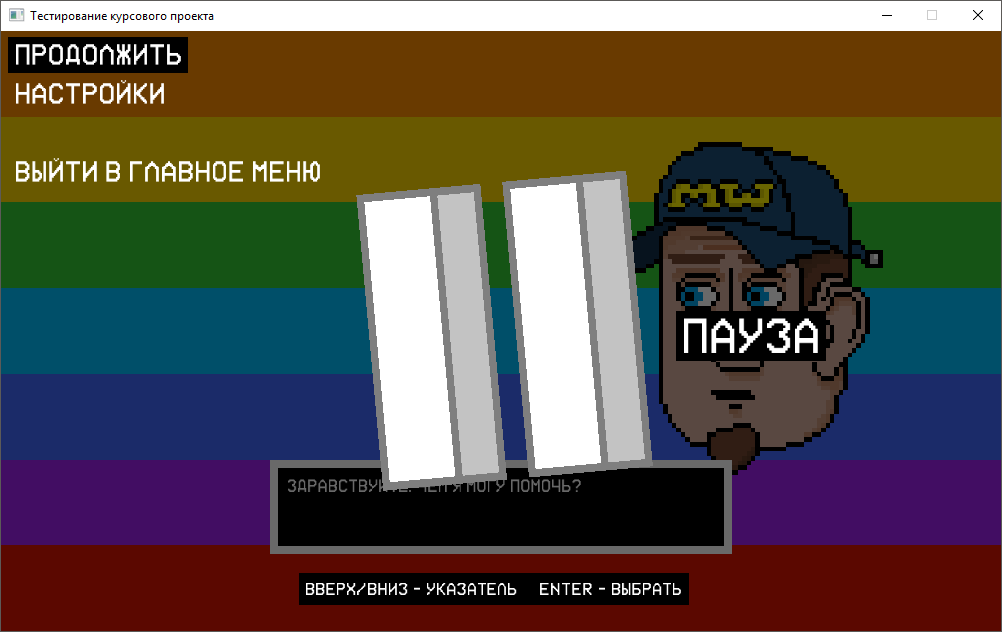


Рис. 25 – Меню паузы

2. 4. Анализ результатов тестирования

По ходу тестирования можно понять, что диалог не будет продолжаться до того момента пока пользователь не нажмет кнопку. Это же видно, когда перед игроком поставлен вопрос, требующий определенного ответа. Так же пользователь может поставить игру на паузу по нажатию клавиши ESCAPE, зайти в настройки или же вовсе прекратить игру и выйти в главное меню.

Когда пользователь нажимает клавишу продолжения диалога, которой является ENTER, происходит анимация диалога – персонажи начинают разговаривать, а текст, который они говорят, отображается постепенно. Когда же нас просят выбрать какое-либо действие, то от него будет зависеть последующее продолжение диалога: смена реплик и тому подобное. Можно заметить, что при нажатии клавиши продолжения диалога здесь не сработает, пользователю приходится делать выбор.

Интерфейс очень простой и понятен для пользователя. Внизу имеются подсказки, показывающие назначение клавиш для выполнения определенных действий. В главном меню и в последующих других курсором выступает черный квадрат, который перемещается в зависимости от выбранного пункта меню. Перемещение курсора осуществляется нажатиями кнопок ВНИЗ и ВВЕРХ, а выбор выделенного пункта осуществляется нажатием на клавишу ENTER. В меню «Настройки», не важно в меню паузы или в главном меню, изменение значения громкости осуществляется клавишами ВПРАВО – увеличить, и ВЛЕВО – уменьшить громкость музыки.

2. 5. Надежность программного обеспеченья

В ходе тестирования и отладки программы, выяснилось, что программа отвечает всем необходимым требованиям в плане надёжности:

* разработанная программа должна обладать средствами защиты от ошибочных действий пользователей;
* все ошибки должны отображаться с комментариями или подсказками по их устранению (где подсказкой является возвращение к вопросу);
* гарантировать сохранность данных при сбоях в работе внешних устройств (сохранение настроек происходит после их изменения и возврата в предыдущее меню, других данных не предусмотрено).

Также для повышения надёжности были приняты следующие меры:

* сконфигурировать аппаратные и программные средства в соответствии с техническими требованиями (по данным из программы «Windows Диспетчер задач» программа соответствует техническим средствам);

3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЮ ПРОГРАММЫ

3. 1. Руководство программиста

Программа предназначена для отображения примера создания пользовательского интерфейса и системы диалога в игре. Главная функция программы – показать пример разработчику, как можно отобразить диалог и наладить игровую связь с пользователем. Также показать вариант представления пользовательского интерфейса. Для ее использования необходимо иметь персональный компьютер, работающий на операционной системе Windows 7 и выше, клавиатуру, монитор.

Данная программа представляет собой графическое приложение на базе SFML. Состоит из исполнительного файла «Course work.exe», папок с необходимыми файлами (картинки и шрифты) и файла «Settings.dat», хранящего настройки.

Обращение к программе происходит после распаковки программы из архива «Course.zip». После распаковки, в папке, в которой распаковали запустить исполнительный файл «Course work.exe».

Входными данными являются нажатия на клавиши, которые в программе считываются как коды. Выходными же данными является графическое представление персонажей и текст их слов, а также элементы управления, разные графические украшательства, музыка и файл «Settings.dat», который хранит настройки программы.

Сообщения прямо говорят пользователю что ему нужно делать. Поэтому данный пункт не имеет описания.

3.2. Руководство пользователя

Эта часть программы создана для отображения диалога и передача информации, необходимой для игры, в которой присутствует диалог. Ею будут пользоваться игроки для запуска игры и получения необходимой части сюжета.

Условия использования программы включают в себя системные требования приведенные ниже.Минимальные системные требования к электронной вычислительной машине (ЭВМ):

* Операционная система: Windows 7/8/10;
* Процессор: Intel Celeron или аналогичный AMD;
* Частота: 1500 МГЦ;
* Оперативная память (ОЗУ): 512МБ;
* Жесткий диск: 100 МБ;
* Видеокарта: видеокарта с поддержкой OpenGL 3.0;
* Память видеокарты: 256 МБ и выше;
* Клавиатура.

Рекомендуемые системные требования к электронной вычислительной машине (ЭВМ):

* Операционная система: Windows 7/8/10;
* Процессор: Intel CoreI3 или аналогичный AMD;
* Частота: 2400 МГЦ;
* Оперативная память (ОЗУ): 1ГБ;
* Жесткий диск: 100 МБ;
* Видеокарта: видеокарта с поддержкой OpenGL 3.0;
* Память видеокарты: 512 МБ и выше;
* Клавиатура и мышь.

Пользователь запускает программу, где он уже сам выбирает действие. Я рекомендую при запуске программы выбирпть пункт меню «Дилог», чтобы просмотреть тестовую версию диалога.

ЗАКЛЮЧЕНИЕ

Во время написания курсовой работы были достигнуты ранее поставленные задачи и разработана программа на языке программирования C++. Были показаны примеры разработки программы с использованием графической библиотеки SFML. Также были получены навыки работы с графикой.

Была создана структура работы с файлами, в которых хранится данные, необходимые для работы программы – пользовательские настройки.

Как и задумывалось программа отображает пример для разработчика по созданию пользовательского графического интерфейса и системы диалогов и реализации связи «Игра – Пользователь». Программа отображает диалог приятно для глаз и проста в использовании, как рядовому пользователю, так и программисту.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Стандарты

1. ГОСТ 7.32. – 2001. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21с.
2. ГОСТ 7.82 -2001. Библиографическая запись. Библиографическое описание электронных ресурсов. Общее требования и правила составления – М.: ИПК Издательство стандартов, 2001. – 21 с.
3. Единая система программной документации. – М.: СТАНДАРТИНФОРМ, 2005.
4. Васильев А.Н. Самоучитель С++ с примерами и задачами. Книга + CD. – СПб.: Наука и техника, 2010. – 480 с.
5. Шлее Макс.Qt 5.3. Профессиональное программирование на C++. Книга +CD. - СПб.: БХВ-Петербург, 2015. — 928 с.

Интернет-ресурсы

1. SFML: Работа с библиотекой SFML – [Электронный ресурс] – Режим доступа: <https://www.sfml-dev.org/>
2. СppStudio: Работа с файлами: язык программирования С++ – [Электронный ресурс] – Режим доступа: http://cppstudio.com/post/446/
3. YouTube: Подключение SFML в Visual Studio 2010 – [Электронный ресурс] – Режим доступа: https://www.youtube.com/watch?v=CVYjVCuPuY4&t=81s&list=WL&index=2
4. YouTube: Создание игр на С++: Марио (платформер) – [Электронный ресурс] – Режим доступа: https://www.youtube.com/watch?v=6OHMFwQK44k&list=WL&index=3

ПРИЛОЖЕНИЕ А

Файл «Libraries.h»

#pragma once

#include <SFML\Graphics.hpp>

#include <SFML\Audio.hpp>

#include <Windows.h>

#include <iostream>

#include <fstream>

using namespace std;

using namespace sf;

double time\_second; //Переменная, которое хранит себе значение для одной секунды

//RenderWindow window(VideoMode(GetSystemMetrics(SM\_CXSCREEN), GetSystemMetrics(SM\_CYSCREEN)), L"Курсовой проект", Style::Fullscreen);

RenderWindow window(VideoMode(1000, 600), L"Тестирование курсового проекта", Style::Close);

double global\_time; //Переменная времени в микросекундах

Clock global\_clock; //Объект измеряющий время за один такт

Texture last\_draw; //Буфер окна

Sprite layer\_sp; //Спрйт для буффера окна

Event event; //Класс србытия

Font global\_font; //Класс, который хранит в себе универсальный шрифт

Music main\_theme; //Класс, отвечающий за вопроизведение музыки

/\*ФУНКЦИЯ ИЗМЕНЕНИЯ ГРОМКОСТИ ЗВУКА\*/

void MusicUpdateVolume(Music &music, int &volume)

{

if (volume > 0)

{

music.setVolume(volume); //Установка громкости звука

/\*Проверка на игру музыки. Если она не играет, то мы ее запускаем.

Это сделано для того, чтобы избежать воспроизведения сначала, если музыка играет\*/

if (!(music.getStatus() == Music::Playing))

music.play(); //Запуск музыки

}

else music.stop();//Остановка музыки (полная остановка)

}

/\*ФУНКЦИЯ ПРОВЕРКИ ФАЙЛА НА ПУСТОТУ\*/

bool IsEmpty(string dir\_file)

{

string tst = "";

ifstream file(dir\_file);

file >> tst;

file.close();

if (tst == "")

{

file.clear();

return true;

}

return false;

}

/\*КЛАСС ХРАНЕНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ДАННЫХ (НАСТРОЕК)\*/

class PlayerData

{

private:

/\*ФУНКЦИЯ КОДИРОВАНИЯ ПРОБЕЛА, ДЛЯ ЕГО СОХРАНЕНИЯ В ФАЙЛЕ\*/

string code(string line)

{

string buffer = line;

for (int i = 0; i < buffer.length(); i++)

if (buffer[i] == ' ') buffer[i] = '\_';

else if (buffer[i] == '\_') buffer[i] = '\*';

return buffer;

}

/\*ФУНКЦИЯ ДЕКОДИРОВАНИЯ ПРОБЕЛА, ДЛЯ ЕГО СОХРАНЕНИЯ В ФАЙЛЕ\*/

string decode(string line)

{

string buffer = line;

for (int i = 0; i < buffer.length(); i++)

if (buffer[i] == '\_') buffer[i] = ' ';

else if (buffer[i] == '\*') buffer[i] = '\_';

return buffer;

}

public:

int settings[3] = { 1, 100, 75 }; //Массив настроек

/\*ЗАПИСЬ ДАННЫХ В ФАЙЛ\*/

void writeFile()

{

ofstream file\_date("Settings.dat"); //Указываем имя файла для записи и открываем его для записи

file\_date.clear(); //Очистка перед записью

for (int i = 0; i < 3; i++) //Запись массива поочередно

file\_date << settings[i] << endl;

file\_date.close(); //Закрытие файла после записи

}

/\*ЧТЕНИЕ ДАННЫХ ИЗ ФАЛА\*/

int readFile()

{

ifstream file\_date("Settings.dat");//Устанавливаем имя файла и открываем его для чтения

if (!file\_date)//Проверяем его на пустоту

{

for (int i = 0; i < 3; i++) //Считываем поочередно

file\_date >> settings[i];

file\_date.close(); //Закрываем файл после чтения

}

else

{

file\_date.close(); //Закрываем файл

writeFile(); //Записываем данные, которые есть.

}

return 0;

}

}player; //Создаем глобальный объект "ИГРОК"

int PauseMenu(bool trans = true, bool up = true); ///Ссылка на функцию

/\*ФУНКЦИЯ РАСЧЁТА МАСШТАБА, ЗАВИСИЩЕГО ОТ РАЗМЕРА ЭКРАНА\*/

Vector2f ScaleWindow(RenderWindow &window, float scale\_x, float scale\_y)

{

Vector2f v = Vector2f(float(window.getSize().x) / 1366.0 \* scale\_x, float(window.getSize().y) / 768.0 \* scale\_y);

return v;

}

/\*ФУНКЦИЯ РАСЧЁТА МАСШТАБА ПО У, ЗАВИСИЩЕГО ОТ РАЗМЕРА ЭКРАНА\*/

double ScaleWindowY(RenderWindow &window, float scale\_Y)

{

return float(window.getSize().y) / 768.0 \* scale\_Y;

}

/\*ФУНКЦИЯ РАСЧЁТА МАСШТАБА ПО Х, ЗАВИСИЩЕГО ОТ РАЗМЕРА ЭКРАНА\*/

double ScaleWindowX(RenderWindow &window, float scale\_X)

{

return float(window.getSize().x) / 1366.0 \* scale\_X;

}

/\*ФУНКЦИЯ СОХРАНЕНИЯ В БУФЕР\*/

void SaveLast()

{

last\_draw.create(window.getSize().x, window.getSize().y); //Очистка буфера

last\_draw.update(window); //Запись содержимого экрана

layer\_sp.setTexture(last\_draw); //Установка текстуры буфера в спрайт

}

/\*ФУНКЦИЯ ПРОРИСКОВКИ БУФЕРА\*/

void DrawLast(RenderWindow &window, Sprite sp)

{

window.draw(sp); //Прорисовка буфера

}

/\*ФУНКЦИЯ РАСЧЕТА ВРЕМЕНИ (ЦИКЛИЧЕСКАЯ)\*/

void WindowAdds()

{

time\_second = global\_clock.getElapsedTime().asSeconds(); //Берем время в секундах

global\_time = global\_clock.getElapsedTime().asMicroseconds(); //Берем время в микросекундах

global\_time /= 800; //Ускоряем в 800 раз (т.к. слишком малое)

global\_clock.restart(); //Перезапускаем таймер

}

/\*ОБРАБОТЧИК СОБЫТИЯ "ЗАКРИТИЕ ОКНА"\*/

void EventWindowClose()

{

if (event.type == Event::Closed)

exit(0);

}

/\*ОБРАБОТЧИК НАЖАТИЯ КЛАВИШИ С ОПРЕДЕЛЕННЫМ КОДОМ\*/

bool OneButtonPressed(Keyboard::Key button)

{

if (event.type == Event::KeyPressed && event.key.code == button) return true;

return false;

}

/\*ПЕРЕГРУЗКА ДЛЯ ВВОДА КОДА В РУЧНУЮ\*/

bool OneButtonPressed(int button)

{

if (event.type == Event::KeyPressed && event.key.code == button) return true;

return false;

}

/\*ПСЕВДОПРОСТРАНСТВО ИМЕН ПОЗИЦИЙ\*/

enum Position

{

Left = 1,

hCenter = 2,

Right = 3,

Top = 10,

vCenter = 20,

Bottom = 30

};

Файл «Menu.h»

#pragma once

#include"Libraries.h"

/\*КЛАСС ПУНКТА МЕНЮ\*/

class Menu

{

private:

static int

character\_size, //Размер шрифта

side\_off; //Сдвиг между элементами

float

general\_position\_x, //Смещение относительно экрана по Х

general\_position\_y; //Смещение относительно экрана по У

Text caption\_text; //Надпсь строки управления (Информация)

Color color\_text; //Цвет шрифта

Texture tx\_icon; //Текстура иконки главного меню

Sprite sprite\_icon; //Спрайт иконки главного меню

RectangleShape menu\_name\_hud; //Подкладка под текст названия главного меню

Text menu\_name; //Текст названия главного меню

public:

Menu()

{

menu\_name\_hud.setFillColor(Color::Black);

menu\_name\_hud.setOrigin(10, 5);

menu\_name\_hud.setScale(ScaleWindow(window, 1, 1));

color\_text = Color::White;

caption\_text.setFont(global\_font);

caption\_text.setScale(ScaleWindow(window, 1, 1));

caption\_text.setFillColor(color\_text);

menu\_name = caption\_text;

menu\_name.setCharacterSize(50);

menu\_name.setScale(ScaleWindow(window, 1, 1));

character\_size = 18;

general\_position\_x = ScaleWindowX(window, 20);

general\_position\_y = ScaleWindowY(window, 13);

side\_off = ScaleWindowY(window, 50.0);

}

/\*ПРОСТАЯ ЧАСТЬ МЕНЮ\*/

Text partVertivcal(wstring caption, int position, int counter, int pos = NULL)

{

if (pos == NULL) pos = position;

caption\_text.setString(caption); //Установка андписи

caption\_text.setPosition(general\_position\_x, general\_position\_y + side\_off\*(pos)); //Установка позиции на экране, в зависимости от порядкового номера

return caption\_text;

}

/\*ЧАСТЬ МЕНЮ С ПЕРЕКЛЮЧЕНИЕМ НАДПИСИ\*/

Text partHorizontal(wstring captions\_horizontal[], wstring caption = L"NO\_CAPTION", int position = 0, int ver\_counter = -100, int hor\_counter = -100, bool first = false)

{

caption\_text.setString(caption + ": " + captions\_horizontal[hor\_counter]); //Установка надписи, в зависимости от ее индекса

caption\_text.setPosition(general\_position\_x, general\_position\_y + side\_off\*(position + first)); //Установка позиции на экране, в зависимости от порядкового номера

return caption\_text;

}

/\*ПЕРЕГРЗКА ДЛЯ ЧИСЛЕННОГО ЗНАЧЕНИЯ\*/

Text partHorizontal(wstring caption, int position = 0, int ver\_counter = -100, int hor\_counter = -100, bool first = false)

{

static char buffer[3] = "\0";

wsprintf(buffer, "%i", hor\_counter);

caption\_text.setString(caption + ": " + buffer);

caption\_text.setPosition(general\_position\_x, general\_position\_y + side\_off\*(position + first));

return caption\_text;

}

/\*НАЗВАНИЕ ГЛАВНОГО МЕНЮ\*/

void MenuName(wstring caption = L"NO\_CAPTION")

{

menu\_name.setString(caption);

menu\_name.setOrigin(menu\_name.getLocalBounds().width / 2, menu\_name.getLocalBounds().height / 2);

menu\_name.setPosition(window.getSize().x/4\*3, window.getSize().y/2);

menu\_name\_hud.setSize(Vector2f(menu\_name.getLocalBounds().width + 20, menu\_name.getLocalBounds().height + 20));

menu\_name\_hud.setOrigin(menu\_name\_hud.getSize().x / 2, menu\_name\_hud.getSize().y / 2 - 7);

menu\_name\_hud.setPosition(menu\_name.getPosition().x, menu\_name.getPosition().y);

window.draw(menu\_name\_hud);

window.draw(menu\_name);

}

/\*УКАЗАТЕЛЬ\*/

RectangleShape textHud(Color color = Color::Black)

{

RectangleShape gg;

gg.setFillColor(color);

gg.setSize(Vector2f(caption\_text.getLocalBounds().width + 20, caption\_text.getLocalBounds().height + 20));

gg.setPosition(caption\_text.getPosition().x, caption\_text.getPosition().y);

gg.setScale(ScaleWindow(window, 1, 1));

gg.setOrigin(10, 5);

return gg;

}

/\*ИКОНКА МЕНЮ\*/

Sprite iconMenu(double time, double minus = 5)

{

static double rotate = 0;

static bool back = true;

if (back && rotate > -minus / (minus \* 10)) rotate -= 0.0001 \* time;

else if (!back && rotate < minus / (minus \* 10)) rotate += 0.0001 \* time;

if (sprite\_icon.getRotation() < 360 - minus && sprite\_icon.getRotation() > 360 - minus \* 2) back = false;

else if (sprite\_icon.getRotation() < minus \* 2 && sprite\_icon.getRotation() > minus) back = true;

sprite\_icon.rotate(rotate);

return sprite\_icon;

}

/\*ЗАГРУЗКА ТЕКСТУРЫ ДЛЯ ИКОНКИ МЕНЮ\*/

void loadTexture(char \*dir\_of\_file)

{

tx\_icon.loadFromFile(dir\_of\_file); //Загружаем из файла

sprite\_icon.setTexture(tx\_icon); //Устанавливаем текстуру в спрайт

sprite\_icon.setPosition(window.getSize().x / 2, window.getSize().y / 2); //Ставим по середине по Х и У

sprite\_icon.setScale(ScaleWindow(window, 10, 10)); //Установка масштаба

sprite\_icon.setOrigin(tx\_icon.getSize().x/2, tx\_icon.getSize().y/2); //Точка ориентирования объекта (если 0, 0 - левый верхний угол)

}

/\*УСТАНОВКА ОРИЕНТАЦИИ ПУНКТА МЕНЮ\*/

void SetPosition(Position pos)

{

if (pos == Position::hCenter) //Установка в центре (по горизонтали)

{

general\_position\_x = window.getSize().x / 2 - caption\_text.getLocalBounds().width / 2;

caption\_text.setPosition(general\_position\_x, caption\_text.getPosition().y);

}

else if (pos == Position::Right) //Установка справа

{

general\_position\_x = window.getSize().x - general\_position\_x - caption\_text.getLocalBounds().width;

caption\_text.setPosition(general\_position\_x, caption\_text.getPosition().y);

}

else if (pos == Position::Left) //Слева

{

general\_position\_x = 20;

caption\_text.setPosition(general\_position\_x, caption\_text.getPosition().y);

}

}

};

int Menu::character\_size, Menu::side\_off; ///Инициализация статических переменных

/\*ИЗМЕНЕНИЕ ЗНАЧЕНИЯ УКАЗАТЕЛЯ ПО ВЕРТИКАЛИ\*/

bool VChangeCounter(int &counter, int min, int max)

{

if (OneButtonPressed(Keyboard::Up)) --counter; //По нажатию кнопки вверх

if (OneButtonPressed(Keyboard::Down)) ++counter; //По нажатию кнопки вниз

//Обработка по ограничению

if (counter > max) { counter = min; return false; }

else if (counter < min) { counter = max; return false; }

else return true;

}

/\*ИЗМЕНЕНИЕ ЗНАЧЕНИЯ УКАЗАТЕЛЯ ПО ГОРИЗОНТАЛИ\*/

bool HChangeCounter(bool to\_change, int &counter, int min, int max)

{

window.setKeyRepeatEnabled(to\_change); //Установка повторений по удержанию клавиш

if (to\_change) //Разрешение на выполнение

{

if (OneButtonPressed(Keyboard::Left)) --counter; //По нажатию клавиши влево

if (OneButtonPressed(Keyboard::Right)) ++counter; //По нажатию клавиши вправо

}

//Обработка по ограничению

if (counter > max) { counter = max; return false; }

else if (counter < min) { counter = min; return false; }

else return true;

}

Файл «Drawing.h»

#pragma once

#include "Libraries.h"

/\*КЛАСС РИСОВАНИЯ ОБЩЕГО НАЗНАЧЕНИЯ\*/

class Draw

{

private:

Text text; //Текст для подсказок

RectangleShape gg; //Подкладка под текст подсказки

public:

double intensivity; //Прозрачность для перехода

bool fade\_in; //переключатель между анимацией входа и выхода

Draw()

{

intensivity = 0;

fade\_in = true;

}

/\*ПОДСКАЗКА\*/

Text getDescription(wstring description)

{

text.setFont(global\_font);

text.setCharacterSize(18);

text.setString(description);

text.setPosition(window.getSize().x / 2, window.getSize().y - 15 - 35);

text.setOrigin(gg.getSize().x/2, 0);

text.setOutlineColor(Color::Black);

text.setScale(ScaleWindow(window, 1, 1));

return text;

}

/\*ПОДКЛАДКА ПОД ПОДСКАЗКУ\*/

RectangleShape textHud(Color color = Color::Black)

{

gg.setFillColor(color);

gg.setSize(Vector2f(text.getLocalBounds().width + 20, text.getLocalBounds().height + 25));

gg.setPosition(text.getPosition().x, text.getPosition().y);

gg.setOrigin(gg.getSize().x / 2 + 10, 10);

gg.setScale(ScaleWindow(window, 1, 1));

return gg;

}

/\*ПЕРЕХОД ИЗ\*/

RectangleShape fadeOut(double time, double speed = 0.5)

{

RectangleShape rect;

if (intensivity < 254)

{

intensivity += speed \* time;

if (intensivity > 255) intensivity = 255;

rect.setFillColor(Color(0, 0, 0, intensivity));

}

else rect.setFillColor(Color::Black);

rect.setSize(Vector2f(window.getSize().x, window.getSize().y));

rect.setPosition(0, 0);

return rect;

}

/\*ПЕРЕХОД В\*/

RectangleShape fadeIn(double time, double speed = 0.5)

{

RectangleShape rect;

if (intensivity < 254)

{

if (intensivity < 255) intensivity += speed \* time;

if (intensivity > 255) intensivity = 255;

rect.setFillColor(Color(0, 0, 0, 255 - intensivity));

}

else rect.setFillColor(Color(0, 0, 0, 0));

rect.setSize(Vector2f(window.getSize().x, window.getSize().y));

rect.setPosition(0, 0);

return rect;

}

/\*ОБЪЕДЕНЕННЫЕ ПЕРЕХОДЫ С ТРИГЕРОМ В ВИДЕ ЛОГИЧЕСКОЙ ПЕРЕМЕННОЙ\*/

bool fades(bool end, double speed = 0.5)

{

if (end)

{

window.draw(fadeOut(global\_time, speed));

if (intensivity > 254) return true;

}

else if (fade\_in)

{

window.draw(fadeIn(global\_time, speed));

if (intensivity > 254)

{

intensivity = 0;

fade\_in = false;

}

}

return false;

}

~Draw(){ }

};

/\*КЛАСС АНИМАЦИИ\*/

class Animation

{

private:

Texture txt; //Текстура для анимации

int x = 0, y = 0, width = 0, height = 0, frames = 0; //Размеры и положения квадрата, а так же кадры

bool mirror = false; //Отзеркаливание

public:

double current\_frame = 0; //Текущий кадр

Sprite sprite; //Спрайт для анимации

Animation(const char \*dir\_of\_file, int x, int y, int width, int height, double scale\_X = 1, double scale\_Y = 1, float posX = 0, float posY = 0)

{

txt.loadFromFile(dir\_of\_file);

sprite.setTexture(txt);

sprite.setScale(ScaleWindow(window, scale\_X, scale\_Y));

this->x = x;

this->y = y;

this->width = width;

this->height = height;

sprite.setPosition(posX, posY);

update();

}

/\*ОБНОЛВЛЕНИЕ АНИМАЦИИ (СМЕНА КАДРОВ)\*/

void update(double time = 0)

{

current\_frame += time;

if (mirror)

{

if (current\_frame > 2\*frames) current\_frame = 0;

if (current\_frame < frames) sprite.setTextureRect(IntRect(int(current\_frame)\*x, y, width, height));

else sprite.setTextureRect(IntRect(x\*int(2\*frames - current\_frame) + x, y, -width, height));

}

else

{

if (current\_frame > frames) current\_frame = 0;

sprite.setTextureRect(IntRect(int(current\_frame)\*x, y, width, height));

}

}

/\*УСТАНОВИТЬ ОПРЕДЕЛЕННЫЙ КАДР\*/

void setFrame(double frame)

{

current\_frame = frame;

if (mirror)

{

if (current\_frame < frames) sprite.setTextureRect(IntRect(int(current\_frame)\*x, y, width, height));

else sprite.setTextureRect(IntRect(x\*int(2 \* frames - current\_frame) + x, y, -width, height));

}

else sprite.setTextureRect(IntRect(int(current\_frame)\*x, y, width, height));

}

/\*УСТАНОВКА ОРИЕНТАЦИИ\*/

void setOrigin(Position pos = Position::Bottom)

{

if(pos == Position::Bottom) sprite.setOrigin(width/2,height);

}

int setMirror(bool mirror) { this->mirror = mirror; return 0; }

int setFrames(int frames) { this->frames = frames; return 0; }

int getWidth() { return width; }

int getHeight() { return height; }

float getScaleX() { return sprite.getScale().x; }

float getScaleY() { return sprite.getScale().y; }

};

/\*РИСОВАНИЕ КРАСНО-ЧЕРНОГО ГРАДИЕНТА\*/

void SGradientRB()

{

Vertex rect[4] =

{

Vertex(Vector2f(0, 0), Color(25, 0, 0)),

Vertex(Vector2f(window.getSize().x, 0), Color(25, 0, 0)),

Vertex(Vector2f(window.getSize().x, window.getSize().y), Color(196, 0, 0, 150)),

Vertex(Vector2f(0, window.getSize().y), Color(196, 0, 0))

};

window.draw(rect, 4, Quads);

}

/\*АНИМАЦИИ "ЛИНИИ РАДУГИ"\*/

void LinesRainbow(double speed = 10)

{

static RectangleShape Rainbow[7];

static int stage = 0;

static double current\_frame = 0;

static Color color[7] =

{

Color(220, 20, 0),

Color(255, 140, 0),

Color(255, 215, 0),

Color(50, 205, 50),

Color(0, 191, 255),

Color(65, 105, 255),

Color(160, 32, 240)

}, buffer;

for (int i = 0; i < 7; i++)

{

Rainbow[i].setSize(Vector2f(window.getSize().x, double(window.getSize().y) / 7));

Rainbow[i].setPosition(0, double(window.getSize().y)/ 7 \* i);

}

current\_frame += speed\*time\_second;

if ((int)current\_frame % 2 == 0)

{

current\_frame = 1;

for (int i = 0; i < 6; i++)

{

buffer = color[i];

color[i] = color[i + 1];

color[i + 1] = buffer;

}

}

Rainbow[0].setFillColor(color[0]);

Rainbow[1].setFillColor(color[1]);

Rainbow[2].setFillColor(color[2]);

Rainbow[3].setFillColor(color[3]);

Rainbow[4].setFillColor(color[4]);

Rainbow[5].setFillColor(color[5]);

Rainbow[6].setFillColor(color[6]);

window.draw(Rainbow[0]);

window.draw(Rainbow[1]);

window.draw(Rainbow[2]);

window.draw(Rainbow[3]);

window.draw(Rainbow[4]);

window.draw(Rainbow[5]);

window.draw(Rainbow[6]);

}

/\*ГЛАВНЙ ФОН (ПРОЕЗД МИМО БАШЕН)\*/

void MainBackground(double time, bool reset = false)

{

const int n\_of = 5;

static Animation Panel[n\_of] =

{

Animation("Images\\Backgrounds\\Tower.png", 94, 0, 94, 79, 4, 9, window.getSize().x / (n\_of - 1) \* 0, window.getSize().y + 10 \* 9),

Animation("Images\\Backgrounds\\Tower.png", 94, 0, 94, 79, 4, 5, window.getSize().x / (n\_of - 1), window.getSize().y + 10 \* 5),

Animation("Images\\Backgrounds\\Tower.png", 94, 0, 94, 79, 4, 7, window.getSize().x / (n\_of - 1) \* 2, window.getSize().y + 10 \* 7),

Animation("Images\\Backgrounds\\Tower.png", 94, 0, 94, 79, 4, 9, window.getSize().x / (n\_of - 1) \* 3, window.getSize().y + 10 \* 9),

Animation("Images\\Backgrounds\\Tower.png", 94, 0, 94, 79, 4, 5, window.getSize().x / (n\_of - 1) \* 4, window.getSize().y + 10 \* 5),

};

if (reset) for (int i = 0; i < n\_of; i++)

Panel[i].sprite.setPosition(window.getSize().x / (n\_of - 1) \* i, window.getSize().y + 10\*Panel[i].getScaleY());

for (int i = 0; i < n\_of; i++)

{

Panel[i].setOrigin(Position::Bottom);

Panel[i].setMirror(true);

Panel[i].setFrames(5);

if (Panel[i].sprite.getPosition().x - Panel[i].getWidth() / 2 \* Panel[i].getScaleX() > window.getSize().x)

Panel[i].sprite.setPosition(0 - Panel[i].getWidth() / 2 \* Panel[i].getScaleX(), Panel[i].sprite.getPosition().y);

Panel[i].sprite.move(ScaleWindowX(window, 300 \* time\_second), 0);

if (Panel[i].sprite.getPosition().x <= double(window.getSize().x / 2)) Panel[i].setFrame(0);

if (Panel[i].sprite.getPosition().x >= double(window.getSize().x / 2 - window.getSize().x / 4)) Panel[i].setFrame(1);

if (Panel[i].sprite.getPosition().x >= double(window.getSize().x / 2 - window.getSize().x / 6)) Panel[i].setFrame(2);

if (Panel[i].sprite.getPosition().x >= double(window.getSize().x / 2 - window.getSize().x / 8)) Panel[i].setFrame(3);

if (Panel[i].sprite.getPosition().x >= double(window.getSize().x / 2) - Panel[i].getWidth() / 2) Panel[i].setFrame(4);

if (Panel[i].sprite.getPosition().x >= double(window.getSize().x / 2) + Panel[i].getWidth() / 2) Panel[i].setFrame(7);

if (Panel[i].sprite.getPosition().x >= double(window.getSize().x / 2 + window.getSize().x / 8)) Panel[i].setFrame(8);

if (Panel[i].sprite.getPosition().x >= double(window.getSize().x / 2 + window.getSize().x / 6)) Panel[i].setFrame(9);

if (Panel[i].sprite.getPosition().x >= double(window.getSize().x / 2 + window.getSize().x / 4)) Panel[i].setFrame(10);

}

SGradientRB();

for (int i = 0; i < n\_of; i++)

window.draw(Panel[i].sprite);

}

Файл «Character.h»

#pragma once

#include "Libraries.h"

/\*КЛАСС ПЕРСОНАЖА\*/

class Character

{

private:

Text text\_to\_say; //Текст для диалога

RectangleShape hud; //Текстовое поле

wstring string\_to\_say; ///Строка для добавки пробелов

Texture texture; //Текстура персонажа

Sprite sprite; //Спрайт для персонажа

static double breathing, posy; //Переменные дыхания

static bool up\_point; //Триггер вниз/вверх

double char\_to\_say = 0, current\_frame = 0; //Кол-во символов и текущий кадр

bool mirror = false; //Отзеркаливание

public:

bool talking = false; //Триггер анимации разговора

int x = 0, y = 0, width = 0, height = 0; //Размеры и позиция

int step = 0; //Стадия разговора

Character()

{

text\_to\_say.setFont(global\_font);

text\_to\_say.setCharacterSize(18);

hud.setFillColor(Color::Black);

hud.setOutlineColor(Color::White);

hud.setOutlineThickness(10);

hud.setSize(Vector2f(610, 100));

hud.setOrigin(hud.getSize().x / 2, 0);

text\_to\_say.setScale(ScaleWindow(window, 1, 1));

hud.setScale(ScaleWindow(window, 1, 1));

sprite.setScale(ScaleWindow(window, 6, 6));

hud.setPosition(window.getSize().x / 2, window.getSize().y - hud.getSize().y\*hud.getScale().y - hud.getOutlineThickness() - 75);

}

/\*ТЕКСТ (УРЕЗАННЫЙ ИЛИ ПОЛНЫЙ, В ЗАВИСИМОСТИ ОТ СТАДИИ)\*/

Text getText(double time)

{

if (char\_to\_say < string\_to\_say.length()) char\_to\_say += 0.01\*time;

text\_to\_say.setString(string\_to\_say.substr(0, int(char\_to\_say)));

if (char\_to\_say > string\_to\_say.length()) step = 1;

return text\_to\_say;

}

/\*ПЕРСОНАЖ С ОБНОВЛЕНИЕМ\*/

Sprite getCharacter(double time, bool ready = false)

{

current\_frame += 0.005\*time;

if (current\_frame > 4) current\_frame = 0;

sprite.setTextureRect(IntRect(x\*int(current\_frame) + mirror\*x, y \* !talking, width - mirror\*2\*width, height));

if (up\_point) breathing -= 0.001 \* time;

else if (!up\_point) breathing += 0.001 \*time;

if (breathing > 1) up\_point = true;

else if (breathing < -1) up\_point = false;

sprite.setPosition(sprite.getPosition().x, posy);

sprite.move(0, breathing);

posy = sprite.getPosition().y;

if (step == 1)

{

char\_to\_say = string\_to\_say.length();

talking = false;

}

return sprite;

}

/\*ТЕКСТОВОЕ ПОЛЕ\*/

RectangleShape getHud()

{

text\_to\_say.setOrigin(hud.getSize().x / 2, 0);

text\_to\_say.setPosition(hud.getPosition().x + 10, hud.getPosition().y + 10);

return hud;

}

/\*ЗАГРУЗКА ТЕКСТУРЫ\*/

void LoadTexture(string texture\_)

{

texture.loadFromFile(texture\_);

sprite.setTexture(texture);

sprite.setOrigin(Vector2f(width / 2, height / 2));

}

/\*УСТАНОВКА ПОЗИЦИИ\*/

void setPosition(Position pos)

{

if (pos == Position::Right) mirror = true;

else mirror = false;

sprite.setPosition(window.getSize().x / 4 \* pos, window.getSize().y / 2);

posy = sprite.getPosition().y;

}

/\*УСТАНОВКА ТЕКСТА C Добавлением переноса\*/

void setText(wstring line\_to\_say)

{

int symbols = 0;

char\_to\_say = 0;

if (line\_to\_say == "...") talking = false;

else talking = true;

string\_to\_say = line\_to\_say;

for (int i = 0; i < string\_to\_say.length(); i++)

{

if (string\_to\_say[i] == '\t') symbols += 4;

else if (string\_to\_say[i] != '\n') symbols = 0;

else symbols = 0;

if (symbols >= 30)

{

string\_to\_say = string\_to\_say.substr(0, i) + '\n' + string\_to\_say.substr(i, string\_to\_say.length());

symbols = 0;

}

}

step = 0;

}

~Character() { }

};

double Character::breathing = 0,

Character::posy = 0;

bool Character::up\_point = true;

Файл «main.cpp»

#include "Character.h"

#include "Libraries.h"

#include "Drawing.h"

#include "Menu.h"

///\*ССЫЛКИ

void Settings(wstring caption\_cursor = L"> ");

void SettingsP(wstring caption\_cursor = L"> ");

void MainMenu();

/\*ДИАЛОГ\*/

void DialogSay(Character &per, wstring line = L"NOTHING\_TO\_SAY")

{

per.setText(line);

Draw drawing;

while (window.isOpen())

{

WindowAdds();

window.clear(Color::Black);

LinesRainbow();

window.draw(per.getCharacter(global\_time));

window.draw(per.getHud());

window.draw(per.getText(global\_time));

if (per.step == 1)

{

window.draw(drawing.textHud());

window.draw(drawing.getDescription(L"Нажмите ENTER для продолжения"));

}

window.display();

while (window.pollEvent(event))

{

EventWindowClose();

if (OneButtonPressed(Keyboard::Escape)) PauseMenu();

if (per.step < 3 && OneButtonPressed(Keyboard::Return)) ++per.step;

}

if (per.step == 2) break;

}

}

/\*ДИАЛОГ С АНИМАЦИЕЙ ВХОДА\*/

void DialogSayIn(Character &per, wstring line = L"NOTHING\_TO\_SAY")

{

Draw draw;

per.setText(line);

WindowAdds();

while (window.isOpen())

{

WindowAdds();

while (window.pollEvent(event))

{

EventWindowClose();

if (OneButtonPressed(Keyboard::Escape)) PauseMenu();

if (per.step < 3 && OneButtonPressed(Keyboard::Return)) ++per.step;

}

window.clear(Color::Black);

LinesRainbow();

window.draw(per.getCharacter(global\_time));

window.draw(per.getHud());

window.draw(per.getText(global\_time));

if (per.step == 1)

{

window.draw(draw.textHud());

window.draw(draw.getDescription(L"Нажмите ENTER для продолжения"));

}

draw.fades(false, 0.1);

window.display();

if (per.step == 2) break;

}

draw.~Draw();

}

/\*ДИАЛОГ С АНИМАЦИЕЙ ВЫХОДА\*/

void DialogSayOut(Character &per, wstring line = L"NOTHING\_TO\_SAY")

{

bool end = false;

Draw draw;

per.setText(line);

WindowAdds();

while (window.isOpen())

{

WindowAdds();

while (window.pollEvent(event))

{

EventWindowClose();

if (OneButtonPressed(Keyboard::Escape)) PauseMenu();

if (per.step < 3 && OneButtonPressed(Keyboard::Return)) ++per.step;

}

window.clear(Color::Black);

//MainBackground(global\_time);

LinesRainbow();

window.draw(per.getCharacter(global\_time));

window.draw(per.getHud());

window.draw(per.getText(global\_time));

if (per.step == 1)

{

window.draw(draw.textHud());

window.draw(draw.getDescription(L"Нажмите ENTER для продолжения"));

}

if (end)

{

window.draw(draw.fadeOut(global\_time, 0.1));

if (draw.intensivity > 254) break;

}

window.display();

if (per.step == 2) end = true;

}

}

/\*ДИАЛОГ С ВОПРОСОМ ДЛЯ ПОЛЬЗОВАТЕЛЯ\*/

bool DialogQuestion(Character &per, wstring question, wstring answer1, wstring answer2)

{

bool end = false;

Draw draw;

per.setText(question + "\n\n1. " + answer1 + "\n2. " + answer2);

per.talking = false;

per.step = 1;

WindowAdds();

while (window.isOpen())

{

WindowAdds();

while (window.pollEvent(event))

{

EventWindowClose();

if (OneButtonPressed(Keyboard::Escape)) PauseMenu();

if (OneButtonPressed(Keyboard::Return)) { per.step = 1; }

if (OneButtonPressed(Keyboard::Num1))

{

++per.step;

if (per.step == 2) return true;

}

if (OneButtonPressed(Keyboard::Num2))

{

++per.step;

if (per.step == 2) return false;

}

}

window.clear(Color::Black);

//MainBackground(global\_time);

LinesRainbow();

window.draw(per.getCharacter(global\_time));

window.draw(per.getHud());

window.draw(per.getText(global\_time));

if (per.step == 1)

{

window.draw(draw.textHud());

window.draw(draw.getDescription(L"Выберите вариант ответа (1 / 2)"));

}

window.display();

if (per.step == 2) break;

}

}

/\*ТЕСТ ДИАЛОГА\*/

int T()

{

int pp = 0;

Character Sokol34, Ivan, Police, Phone;

Phone.width = Phone.x = 35;

Phone.height = Phone.y = 66;

Phone.LoadTexture("Images\\Characters\\Phone.png");

Phone.setPosition(Left);

Sokol34.width = Sokol34.x = 54;

Sokol34.height = Sokol34.y = 67;

Sokol34.LoadTexture("Images\\Characters\\Sokol34.png");

Sokol34.setPosition(Left);

Ivan.width = Ivan.x = 61;

Ivan.height = Ivan.y = 76;

Ivan.LoadTexture("Images\\Characters\\IvanGordov.png");

Ivan.setPosition(Right);

Police.width = Police.x = 57;

Police.height = Police.y = 79;

Police.LoadTexture("Images\\Characters\\Police.png");

Police.setPosition(Left);

DialogSayIn(Ivan, L"Здравствуйте, чем я могу помочь?");

DialogSay(Sokol34, L"Здравствуйте...");

DialogSay(Ivan, L"Что с вашим компьютером?");

DialogSay(Sokol34, L"Не работает...");

while (DialogQuestion(Ivan, L"Что сделать с загадочным мужчиной?", L"Выгнать его из ремонтного центра", L"Принять его копьютер на починку"))

{

pp++;

switch (pp)

{

case 2:

DialogSay(Ivan, L"Вы меня не слышали? \nПожалуйста, уйдите.");

DialogSay(Sokol34, L"...");

DialogSay(Ivan, L"...");

break;

case 3:

DialogSay(Ivan, L"НЕМЕДЛЕННО ПОКИНТЕ ПОМЕЩ...");

DialogSay(Sokol34, L"Ты знаешь зачем я пришел.");

DialogSay(Ivan, L"...Простите?");

DialogSay(Sokol34, L"Если ты хочешь, то я уйду. \nНо ты еще пожалеешь об этом, Ваня...");

break;

case 4:

DialogSay(Sokol34, L"Ну что ж... \nТы сделал свой выбор...");

DialogSayOut(Ivan, L"...");

return -1;

default:

DialogSay(Ivan, L"Простите, но у нас сейчас обед. \nЯ попрошу вас уйти.");

DialogSay(Sokol34, L"...");

break;

}

}

DialogSay(Ivan, L"Дайте, я его посмотрю.");

DialogSay(Ivan, L"Так. \nНy, тут ничего сложного. \n2 часа и он как новый.");

DialogSayOut(Sokol34, L"Работайте. \nЯ скоро вернусь...");

return 0;

}

/\*МЕНЮ С ВОПРОСОМ\*/

bool Question(wstring question\_ = L"NO\_QUESTION", bool trans = true)

{

int counter = 2;

bool end = false;

RectangleShape rr;

rr.setSize(Vector2f(window.getSize().x, window.getSize().y));

rr.setFillColor(Color(0, 0, 0, 150));

Menu menu[3];

Draw draw;

menu[0].loadTexture("Images\\Menu icons\\Question.png");

WindowAdds();

while (window.isOpen())

{

WindowAdds();

window.clear(Color::Black);

if (!trans) MainBackground(global\_time);

else DrawLast(window, layer\_sp);

if (trans) window.draw(rr);

window.draw(menu[0].iconMenu(global\_time));

window.draw(menu[0].textHud());

window.draw(menu[0].partVertivcal(question\_, 0, -1));

window.draw(menu[counter].textHud());

window.draw(menu[1].partVertivcal(L"Да", 1, counter, 2));

window.draw(menu[2].partVertivcal(L"Нет", 2, counter, 3));

window.draw(draw.textHud());

window.draw(draw.getDescription(L"ВВЕРХ/ВНИЗ - указатель \tENTER - выбрать"));

if (draw.fades(end, 0.5)) break;

window.display();

while (window.pollEvent(event))

{

EventWindowClose();

if (!end)

{

if (OneButtonPressed(Keyboard::Return)) end = true;

VChangeCounter(counter, 1, 2);

}

}

}

switch (counter)

{

case 1:

return true;

break;

case 2:

return false;

break;

default:

return false;

break;

}

}

/\*ГЛАВНОЕ МЕНЮ\*/

void MainMenu()

{

int counter = 0;

bool end = false;

Menu menu[3];

Draw draw;

WindowAdds();

window.draw(menu[0].partVertivcal(L"Диалог", 0, counter));

window.draw(menu[1].partVertivcal(L"Настройки", 1, counter));

window.draw(menu[2].partVertivcal(L"Выход", 2, counter));

counter = 0;

while (window.isOpen())

{

WindowAdds();

window.clear(Color::Black);

MainBackground(global\_time);

menu[0].MenuName(L"Главное меню");

window.draw(menu[counter].textHud());

window.draw(menu[0].partVertivcal(L"Диалог", 0, counter));

window.draw(menu[1].partVertivcal(L"Настройки", 1, counter));

window.draw(menu[2].partVertivcal(L"Выход", 2, counter, 3));

window.draw(draw.textHud());

window.draw(draw.getDescription(L"ВВЕРХ/ВНИЗ - указатель \tENTER - выбрать"));

if (draw.fades(end, 0.5)) break;

window.display();

while (window.pollEvent(event))

{

EventWindowClose();

if (!end)

{

if (OneButtonPressed(Keyboard::Return)) end = true;

VChangeCounter(counter, 0, 2);

}

}

}

switch (counter)

{

case 0:

T();

MainMenu();

break;

case 1:

Settings();

MainMenu();

break;

case 2:

if (Question(L"Вы действительно хотите выйти?", false))

{

window.close();

main\_theme.stop();

exit(0);

}

MainMenu();

break;

default:

MainMenu();

break;

}

}

/\*МЕНЮ ПАУЗЫ\*/

int PauseMenu(bool trans, bool up)

{

if (up) SaveLast();

int counter = 0;

bool end = false;

RectangleShape rr;

rr.setSize(Vector2f(window.getSize().x, window.getSize().y));

rr.setFillColor(Color(0, 0, 0, 150));

Menu menu[3];

Draw draw;

menu[0].loadTexture("Images\\Menu icons\\Pause.png");

WindowAdds();

while (window.isOpen())

{

WindowAdds();

window.clear(Color::Black);

DrawLast(window, layer\_sp);

window.draw(rr);

window.draw(menu[0].iconMenu(global\_time));

window.draw(menu[counter].textHud());

menu[0].MenuName(L"Пауза");

window.draw(menu[0].partVertivcal(L"Продолжить", 0, counter));

window.draw(menu[1].partVertivcal(L"Настройки", 1, counter));

window.draw(menu[2].partVertivcal(L"Выйти в главное меню", 2, counter, 3));

window.draw(draw.textHud());

window.draw(draw.getDescription(L"ВВЕРХ/ВНИЗ - указатель \tENTER - выбрать"));

if (end) break;

window.display();

while (window.pollEvent(event))

{

EventWindowClose();

if (!end)

{

VChangeCounter(counter, 0, 2);

if (OneButtonPressed(Keyboard::Return)) end = true;

if (OneButtonPressed(Keyboard::Escape)) end = true;

}

}

}

switch (counter)

{

case 0:

break;

case 1:

SettingsP();

PauseMenu(true, false);

break;

case 2:

if (Question(L"Вы действительно хотите выйти в главное меню?", trans)) MainMenu();

else PauseMenu(true, false);

break;

default:

break;

}

return 0;

}

/\*МЕНЮ НАСТРОЕК\*/

void Settings(wstring caption\_cursor)

{

int counter = 1;

bool end = false;

Menu menu[2];

menu[0].loadTexture("Images\\Menu icons\\Settings.png");

Draw draw;

WindowAdds();

player.readFile();

while (window.isOpen())

{

WindowAdds();

window.clear();

MainBackground(global\_time);

window.draw(menu[0].iconMenu(global\_time));

window.draw(menu[counter].textHud());

menu[0].MenuName(L"Настройки");

window.draw(menu[0].partHorizontal(L"Музыка", 0, counter, player.settings[2]));

window.draw(menu[1].partVertivcal(L"Назад", 1, counter, 2));

window.draw(draw.textHud());

window.draw(draw.getDescription(L"ВВЕРХ/ВНИЗ - указатель \tENTER - выбрать \tВЛЕВО/ВПРАВО - изменить"));

if (draw.fades(end, 0.5)) break;

window.display();

while (window.pollEvent(event))

{

EventWindowClose();

if (HChangeCounter(counter == 0, player.settings[2], 0, 100))

MusicUpdateVolume(main\_theme, player.settings[2]);

VChangeCounter(counter, 0, 1);

if (OneButtonPressed(Keyboard::Return) && counter == 1) end = true;

}

}

player.writeFile();

}

/\*МЕНЮ НАСТРОЕК ДЛЯ ПАУЗЫ\*/

void SettingsP(wstring caption\_cursor)

{

int counter = 1;

bool end = false;

Menu menu[2];

menu[0].loadTexture("Images\\Menu icons\\Settings.png");

RectangleShape rr;

rr.setSize(Vector2f(window.getSize().x, window.getSize().y));

rr.setFillColor(Color(0, 0, 0, 150));

Draw draw;

WindowAdds();

player.readFile();

while (window.isOpen())

{

WindowAdds();

window.clear();

DrawLast(window, layer\_sp);

window.draw(rr);

window.draw(menu[0].iconMenu(global\_time));

window.draw(menu[counter].textHud());

menu[0].MenuName(L"Настройки");

window.draw(menu[0].partHorizontal(L"Музыка", 0, counter, player.settings[2]));

window.draw(menu[1].partVertivcal(L"Назад", 1, counter, 2));

window.draw(draw.textHud());

window.draw(draw.getDescription(L"ВВЕРХ/ВНИЗ - указатель \tENTER - выбрать \tВЛЕВО/ВПРАВО - изменить"));

if (draw.fades(end, 0.5)) break;

window.display();

while (window.pollEvent(event))

{

EventWindowClose();

if (HChangeCounter(counter == 1, player.settings[2], 0, 100))

MusicUpdateVolume(main\_theme, player.settings[2]);

VChangeCounter(counter, 0, 1);

if (OneButtonPressed(Keyboard::Return) && counter == 1) end = true;

}

}

player.writeFile();

}

/\*ВХОД ПРОГРАММЫ\*/

int main()

{

main\_theme.openFromFile("Music\\Main\_theme.wav");

main\_theme.setLoop(true);

if (!IsEmpty("PlayerData.dat")) player.readFile();

MusicUpdateVolume(main\_theme, player.settings[2]);

Font console\_font;

console\_font.loadFromFile("Images\\Fonts\\Pixel.ttf");

global\_font.loadFromFile("Images\\Fonts\\VHS.ttf");

window.clear(Color::Black);

window.display();

window.setKeyRepeatEnabled(false);

window.setFramerateLimit(60);

MainMenu();

}