|  |  |
| --- | --- |
| Частное учреждение образования  «Колледж бизнеса и права»  Реализация 2D платформера “Колобок” | |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА | |
| к курсовому проекту по дисциплине  «Основы алгоритмизации и программирование»  КП Т.591004.401 | |
| Руководитель | Белокопыцкая Ю.А. |
| Учащийся | Жебин Н.Г. |
| 2017 | |

Содержание

[Введение. 3](#_Toc481847661)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

КП Т.591004.401 ПЗ

Разраб.

Жебин Н.Г.

Провер.

Белокопыцкая Ю.А.

Реценз.

Н. Контр.

Утверд.

Программная реализация игры “Колобок”

Лит.

Листов

КБП

57

у

[1 Объектно-ориентированный анализ и проектирование системы. 5](#_Toc481847662)

[1.1 Сущность задачи. 5](#_Toc481847663)

[1.2 Проектирование модели. 5](#_Toc481847664)

[2 Вычислительная система. 7](#_Toc481847665)

[2.1 Требования к аппаратным и операционным ресурсам. 7](#_Toc481847666)

[2.2 Инструменты разработки. 7](#_Toc481847667)

[3 Проектирование задачи. 8](#_Toc481847668)

[3.1 Требования к приложению. 8](#_Toc481847669)

[3.2 Концептуальный прототип. 8](#_Toc481847670)

[3.3 Организация данных. 9](#_Toc481847671)

[3.4 Функции: логическая и физическая организация 9](#_Toc481847672)

[3.5 Проектирование справочной системы приложения 9](#_Toc481847673)

[4 Описание программного средства. 9](#_Toc481847674)

[4.1 Общие сведения. 9](#_Toc481847675)

[4.2 Функциональное назначение. 10](#_Toc481847676)

[4.3 Входные и выходные данные. 10](#_Toc481847677)

[5 Методика испытаний. 10](#_Toc481847678)

[5.1 Технические требования 10](#_Toc481847679)

[5.2 Порядок проведение испытаний. 11](#_Toc481847680)

[5.2.1 Функциональное тестирование. 11](#_Toc481847681)

[5.2.2 Полное тестирование. 13](#_Toc481847682)

[6 Применение. 16](#_Toc481847683)

[6.1 Назначение программы. 16](#_Toc481847684)

[6.2 Условия применения. 17](#_Toc481847685)

[6.3 Справочная система. 17](#_Toc481847686)

Введение.

Данная курсовая работа является примером реализации 2D-платформера «Колобок», написанной с применением объектно-ориентированного программирования на языке высокого уровня С++ в среде Visual Studio. C++ (Си++) — компилируемый статически типизированный язык программирования общего назначения. Поддерживает разные парадигмы программирования, но, в сравнении с его предшественником — языком Си, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования. Название «C++» происходит от Си (C), в котором унарный оператор ++ обозначает инкремент переменной. В 1990-х годах язык стал одним из наиболее широко применяемых языков программирования общего назначения.

Основной задачей разрабатываемого приложения будет совершенствование и расширение игровых умений.

На данный момент существуют аналоги данного программного приложения (Red Ball).

Целью курсового проекта является применение полученных знаний по дисциплине «Основы алгоритмизации и программирование».

Пояснительная записка состоит из шести разделов и содержит необходимую и достаточную информацию по использованию данного программного продукта.

В первом разделе «Объектно-ориентированный анализ и проектирование системы» описывается исследование предметной области, определение круга задач, которые должны быть автоматизированы или модифицированы.

Описывается задача, перечисляются основные функции программы (обоснование необходимости компьютерной обработки информации или процессов, как задача будет упрощать или усложнять решение поставленной задачи), приводится обзор существующих аналогов программы с указанием их достоинств и недостатков.

Во втором разделе «Вычислительная система» перечисляются требования к аппаратному обеспечению и конфигурации компьютера, проводится характеристика операционной системы, обоснование выбранной среды для разработки приложения. Описываются новые возможности программы, а также ее отличия от предыдущих версий.

В третьем разделе «Проектирование задачи» проводится объектно-ориентированный анализ задачи, строится алгоритм в виде математической модели или в виде блок-схемы строится концептуальный прототип системы меню, диалоговых окон и элементов управления в рамках среды разработки приложения.

В четвертом разделе «Описание программного средства» содержится информация об обозначении и наименовании приложения, о программном обеспечении, необходимом для функционирования программы, о классах решаемых задач и назначении программы.

В пятом разделе «Методика испытаний» описываются требования к техническим средствам для проведения испытаний, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Описывается проверка каждого пункта меню, каждой операции, которая выполняет приложение. Моделируются все возможные действия пользователя при работе с программой. Представляются результаты тестирования.

Шестой раздел «Применение» предназначен для описания сведений о назначении программного средства в классе решаемых задач, ограничениях, накладываемых на область применения, требования к необходимым для данного программного средства техническим и программным ресурсам, структуре справочной системы, методике использования справочной системы и области ее применения.

В заключении будет проанализировано выполнение поставленной задачи и выполненной работы, степень ее соответствия проектным решениям, причины несоответствия (если имеются).

1. Объектно-ориентированный анализ и проектирование системы.
   1. Сущность задачи.

Разработать программную реализацию на языке С++ в среде Visual Studio. Она должна содержать реализацию 2D-платформера «Колобок». Необходимо обеспечить: движение персонажей; уровни сложности; повышение характеристик персонажа; начисление бонусов; реализация жизней; сбор предметов.

Обеспечить организацию лаконичного интерфейса программы средствами создания меню, кнопочных форм, панелей инструментов. Создать справочную систему приложения и установочный диск.

* 1. Проектирование модели.

Согласно общим требованиям стандартный графический интерфейс разрабатываемого приложения должен:

* содержать привычные и понятные пользователю пункты меню, соответствующие функциям обработки;
* сохранять стандартное назначение и местоположение на экране графических объектов, работающих в среде Windows;
* удовлетворять правилам создания элементов интерфейса (командных кнопок, текстовых надписей, меню, списков и так далее).

Состав для диалоговых систем представляет собой:

* меню, где пользователь предлагается выйти из программы, либо просмотреть сведенья о программе.

Главное меню будет содержать следующие пункты: «Начать игру», «Помощь». Пункт меню «Начать игру» будет содержать подпункты выбора уровня – «1, 2, 3, 4, 5». Пункт меню «Помощь» будет содержать информацию, которая поможет ознакомиться пользователю с интерфейсом программы (клавиши управления и т.д.).

Все элементы управления должны быть расположены в оптимальном порядке использования.

В приложении предусмотрена справочная система, которая призвана обеспечить пользователя максимально точной информацией по всем интересующим его темам, касающихся программного средства.

Справочная система обладает удобными средствами навигации и поиска информации.

Начало

Выход

Начать заново

Проиграл

Исчерпаны жизни

Выбор уровня

Уровень++

Меню победы

Да

Нет

Уровень пройден?

Нет

Продолжить

Выход

Меню паузы

Да

Esc?

Загрузка уровня

Выбран любой уровень

Esc

Выбор уровня

Нет

Да

Выход

Отображение

Помощь

Начать игру

Esc?

Выход

Какой пункт меню выбран?

1. Вычислительная система.
   1. Требования к аппаратным и операционным ресурсам.

Для функционирования игры необходим компьютер IBM PC совместимый и имеющий следующие технические средства:

* процессор Intel Pentium 4 или совместимый;
* объем свободной оперативной памяти 256 Мб;
* объем необходимой памяти на жестком диске 100 Кб;
* стандартный VGA-монитор или совместимый;
* мышь и клавиатура;
  1. Инструменты разработки.

Язык программирования С++ в среде Visual Studio.

Microsoft Visual Studio — линейка продуктов компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), включающих [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) программного обеспечения и ряд других инструментальных средств. Visual Studio включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Visual Studio позволяет создавать и подключать сторонние дополнения ([плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD)) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем [контроля версий исходного кода](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8) (как, например, [Subversion](https://ru.wikipedia.org/wiki/Subversion) и [Visual SourceSafe](https://ru.wikipedia.org/wiki/Microsoft_Visual_SourceSafe)), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на [предметно-ориентированных языках программирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)) или инструментов для прочих аспектов [процесса разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) (например, клиент Team Explorer для работы с [Team Foundation Server](https://ru.wikipedia.org/wiki/Team_Foundation_Server)).

SFML ([англ](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA). *Simple and Fast Multimedia Library* — простая и быстрая мультимедийная библиотека) — свободная [кроссплатформенная](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) мультимедийная [библиотека](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Написана на [C++](https://ru.wikipedia.org/wiki/C%2B%2B), но доступна также для [C](https://ru.wikipedia.org/wiki/C_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [D](https://ru.wikipedia.org/wiki/D_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [Java](https://ru.wikipedia.org/wiki/Java), [Python](https://ru.wikipedia.org/wiki/Python), [Ruby](https://ru.wikipedia.org/wiki/Ruby), [OCaml](https://ru.wikipedia.org/wiki/OCaml), [.Net](https://ru.wikipedia.org/wiki/.NET_Framework) и [Go](https://ru.wikipedia.org/wiki/Go). Представляет собой объектно-ориентированный аналог [SDL](https://ru.wikipedia.org/wiki/Simple_DirectMedia_Layer).SFML содержит ряд модулей для простого программирования игр и мультимедиа приложений.

В настоящее время доступны следующие модули:

* System — управление временем и потоками, он является обязательным, так как все модули зависят от него.
* Window — управление окнами и взаимодействием с пользователем.
* Graphics — делает простым отображение графических примитивов и изображений, для своей работы требует модуль Window.
* Audio — предоставляет интерфейс для управления звуком.
* Network — для сетевых приложений.

Box2D — [компьютерная программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), [свободный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [открытый](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [физический движок](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA). Box2D является физическим движком реального времени и предназначен для работы с двумерными физическими объектами. Движок разработан Эрином Катто ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Erin Catto*), написан на [языке программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [C++](https://ru.wikipedia.org/wiki/C%2B%2B). Движок используется в двумерных компьютерных играх.

1. Проектирование задачи.
   1. Требования к приложению.

При проектировании приложения необходимо учитывать тот факт, что программное средство должно удовлетворять комплексу требований. Эти требования следующие:

* накладываемые ограничения (на использование приложения). В данном случае для разрабатываемого приложения ограничений нет.
* движение персонажей;
* уровни сложности;
* повышение характеристик персонажа;
* начисление бонусов;
* реализация жизней;
* сбор предметов.
  1. Концептуальный прототип.

Концептуальный прототип состоит из описания внешнего пользовательского интерфейса – системы меню, диалоговых окон и элементов управления. Кнопочная навигация дублирует часто используемые пункты основного меню.

В приложении будет разработано главное меню, обеспечивающее быстрый и удобный доступ к функциям приложения.

В таблице 3.1 представлена структура главного меню и соответствующее действие при выборе определенного пункта.

Таблица 3.2 – Структура главного меню

|  |  |  |
| --- | --- | --- |
| Уровень | Название | Действие |
| 1 | Начать игру | Вызов подменю выбора уровня. |
| 1 | Помощь | Вызов подменю, описывающего интерфейс взаимодействия с программой в игровом процессе. |
| 1 | Выход | Выход из программы. |

* 1. Организация данных.

Для удобства работы с программным средством имеется главное меню, состоящее из наиболее часто используемых пунктов. Разработанное меню будет иметь структуру, представленную в таблице 3.2.

* + 1. Функции: логическая и физическая организация

Рассмотрим основные функции программы:

Основная функция, отвечающая за определение основных данных и игровой процесс:   
bool startGame(){  
soundBonusOff = true; // звук при активации бонуса

b2Vec2 Gravity(0.f, 9.8f); // величина гравитации

b2World World(Gravity); // переменная мира

fromSLMtoMM: // Из меню выбора уровня возвращаемся в главное меню

fromSLMtoMM = false;

if(toNextLvl == 0 && Restart == false) setMenu(window); // Если нам не нужно на следующий уровень

// и мы не перезапускаем его, то только тогда можем установить главное меню

else if (toNextLvl == 1) SelectLvl = true; // иначе если мы переходим на слеюущий уровень

if (exfromGame == false) return false; // выходим из игры

if (SelectLvl || Restart == true || toNextLvl == 2) {

if ((toNextLvl == 0 || toNextLvl == 2) && Restart == false) // если нам нужно на след. лвл и не перезапускаем текущий лвл

{ nlvl = setSLMenu(window); // номер уровня определяется возвращаемым значением из меню выбора уровня

if (fromSLMtoMM == true) goto fromSLMtoMM;} // если false, переходим по метке из меню выбора уровня в главное

else if (Restart == false) { if (nlvl < 5)nlvl++; }; // иначе если не перезапускаем уровень, тогда если лвл меньше 5

// инкрементируем переменную nlvl

switch (nlvl) // в зависимости от значения переменой, устанавливаем данные определённого уровня

{

case 1:lvl.setLevel1(World); break;

case 2:lvl.setLevel2(World); break;

case 3:lvl.setLevel3(World); break;

case 4:lvl.setLevel4(World); break;

case 5:lvl.setLevel5(World); break;

}}

SelectLvl = false; Restart = false; // сбрасываем переменные

toNextLvl = 0;

window.setFramerateLimit(70); // устанавлиаваем лимит кадров в секунду(70)

view.reset(FloatRect(0, 0, 683 + 341.5f, 384 + 192.0f)); // устанавливаем вид камеры

Clock clock; //привязка ко времени(а не загруженности/мощности процессора).

Player MainPlayer("player.png", 17, 0, 160, 33, 33, 22, 12, World); // создаём персонажа

float CurrentFrame = 0; // Хранит текущий кадр

float time; // перменная времени  
while (window.isOpen()){…} Цикл, отражающий игровую логику за кадр, попутно отображая необходимые элементы игры.}

Функция, которая реализует рестарт игры (деструктор автоматически вызывается компилятором в тот момент, когда объект выходит из области видимости):

void gameRunning() { if (startGame() && exfromGame) { gameRunning(); }}

Данная функция проверяет булевые значения функции startGame() и переменной exfromGame. Если они обе равны значению true, тогда эта же функция вызывается еще раз и таким образом, пока одно из проверяемых значений не будет равно false, функция будет продолжать проверку, иначе её работа будет прекращена, тем самым завершив работу программы.

Функция void main(){// Чтение из файл номер пройденного уровня для дальнейшего доступа к предшествующим уровня до данного

int lastComplLvl = 0;

std::ifstream save1("savelvl.txt");

save1 >> lastComplLvl;

nlvl = lastComplLvl;

save1.close();

for (int i = 0; i < lastComplLvl; i++) access[i] = true; // Отрытие доступа пройденным ранее уровням

gameRunning(); // функция работы игры

// Запись в файл номер пройденного уровня для дальнейшего доступа к предшествующим уровня до данного

for (int i = 4; i >= 0; i--) { if (access[i] == true) { lastComplLvl = i+1; break; } }

std::ofstream save("savelvl.txt");

save << lastComplLvl;

save.close();}   
вызывает вышеуказанную функцию (Запуск игры) и также записывает и считывает данные из файла, где содержится номер последнего пройденного пользователем уровня.

* 1. Проектирование справочной системы приложения

В справочной системе содержится подробное описание работы программы с описанием необходимых для корректной работы программы действий. Данная система позволит пользователю получить конкретную информацию по вопросу работы с программой.

Справочная система выполнена в виде скомпилированного chm-файла и разработана в пакете «Dr.Explain».

1. Описание программного средства.
   1. Общие сведения.

Программное приложение представляет собой самостоятельно разрабатываемый файл «Kolobok.exe». Его объем занимает 142 МБ (148 914 176 байт) дисковой памяти.

Состав программы:

* «Box2D(Папка с файлами)» - Box2D является физическим движком реального времени и предназначен для работы с двумерными физическими объектами;
* «images(Папка с файлами)» - содержит файлы изображений, которые используются в данном программной приложении;
* «lib(Папка с файлами)» - папка хранения двух файлов Box2D.ofl и Box2D.pdd для подключения библиотеки Box2D к данному проекту;
* «music(Папка с файлами)» - хранит файлы коротких звуковых произведений в формате .wav;
* «SFML-2.4.2(Папка с файлами)» - свободная [кроссплатформенная](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) мультимедийная [библиотека](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Написана на C++. Представляет собой объектно-ориентированный аналог [SDL](https://ru.wikipedia.org/wiki/Simple_DirectMedia_Layer).SFML содержит ряд модулей для простого программирования игр и мультимедиа приложений;

Данное программное средство представляет собой приложение, разработанное средствами [кроссплатформенной](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) мультимедийной [библиотеки](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) SFML, физического движка реального времени Box2D, предназначенного для работы с двумерными физическими объектами, с использованием среды Visual Studio 2015.

Для создания справочной системы приложения использовалась программа Dr.Explain.

Для работы с программным средством пользователю, нужна лишь операционная система Windows, необходимые минимальные параметры конфигурации системы. Структура проекта содержит все файлы и папки, необходимые компиляции объектного файла.

* 1. Функциональное назначение.

Игра “Колобок” является средством развлечения и отдыха. В игре человек получает удовольствие, снимает нервное напряжение. Игра носит характер активной познавательной деятельности, становится действенным средством умственного и физического развития, нравственного и эстетического воспи­тания. С помощью игры познается мир, воспитывается творче­ская инициатива, пробуждается любознательность, активизиру­ется мышление.

* 1. Входные данные.

Выходными данными служат изменённые значение переменных после прохождения одного из уровней с определёнными условиями. В следствии этого, при переходе на следующий уровень данной игры, пользователю будут доступны некоторые дополнительные возможности, позволяющие повысить характеристики персонажа на определённый промежуток времени, а именно 10 секунд. Количество бонусов, позволяющих изменить характеристики персонажа, зависит от количества пройденных уровней с условием сбора всех 5-и звезд. Максимум доступно 3 бонуса.

* 1. Выходные данные.

В качестве входных данных служит текстовый файл, хранящий значения основных переменных, которые изменяются при использовании данного программного продукта некоторое время.

1. Методика испытаний.
   1. Технические требования

Для полноценного функционирования разработанного программного приложения, качественного выполнения всех поставленных задач, требуется ПК следующей конфигурации:

**Минимальные системные требования:**

* Операционная система: Windows XP SP3/Vista/7/8/10.
* Процессор (CPU): c двумя и более физическими ядрами.
* Оперативная память (RAM): 1.5 GB для Windows XP SP3, 2 GB для Windows Vista/7/8.
* Видеоадаптер: GeForce 6800/ ATI HD 2400 XT 256 MB, DirectX 9.0c.
* Аудио-карта: совместимая с DirectX 9.0c.
* Свободное место на жёстком диске: 25 GB.

**Рекомендуемые системные требования:**

* Операционная система: Windows XP SP3/Vista /7/8/10 — 64-bit.
* Процессор (CPU): Intel Core i5-3330
* Оперативная память (RAM): 4 GB (или более).
* Видеоадаптер: GeForce GTX660 (2GB) / Radeon HD 7850 2GB, DirectX 9.0c
* Аудио-карта: совместимая с DirectX 9.0c.
* Свободное место на жёстком диске: 36 GB.
  1. Порядок проведение испытаний.
     1. Функциональное тестирование.

При функциональном тестировании осуществляется проверка каждого пункта меню, каждой операции, с целью проверки выполнения всех функций, определенных на этапе объектно-ориентированного анализа и проектирования. Функциональное тестирование должно гарантировать работу всех элементов управления в автономном режиме.

Последовательность действий по процессу функционального тестирования приведена на рисунках ниже:



Рисунок 1 - Главное меню

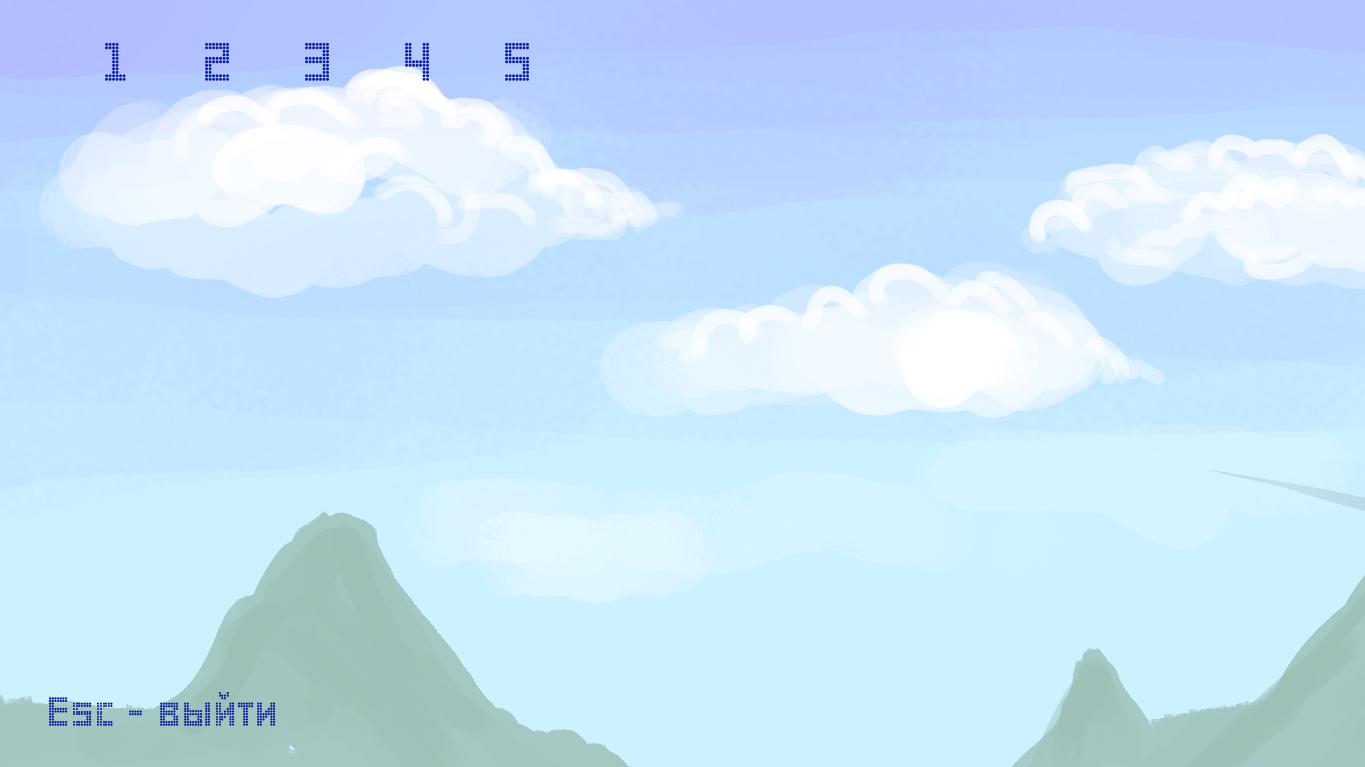


Рисунок 2 - Меню выбора уровня



Рисунок 3 - Меню поражения

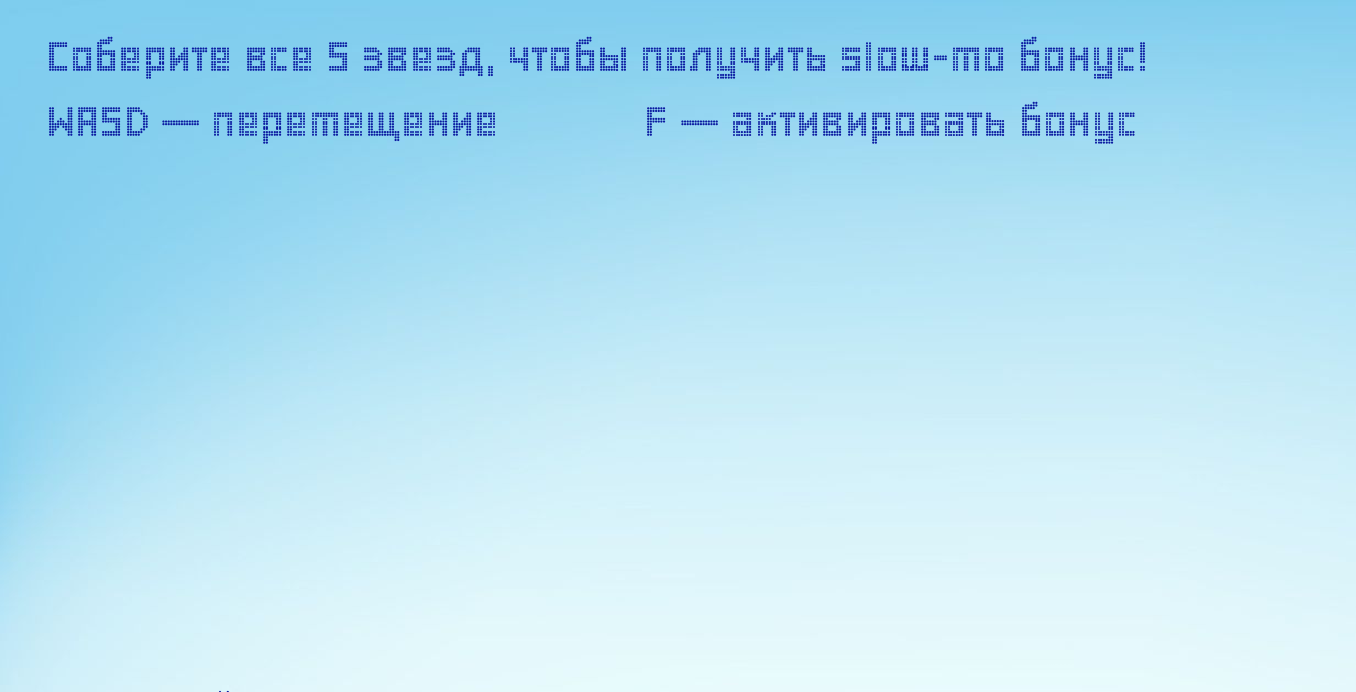


Рисунок 4 - Меню помощи



Рисунок 5 - Меню победы

* + 1. Полное тестирование.

Полное тестирование выполняется после успешного завершения функционального тестирования, продемонстрировавшего корректное выполнение всех функций, определенных на этапе проектирования.

Полное тестирование показывает работу программного продукта с момента запуска до завершения с иллюстрацией реакций программы на некорректные действия пользователя.

Последовательность действий по процессу полного тестирования приведена на рисунках ниже:



Рисунок 6 - Главное меню

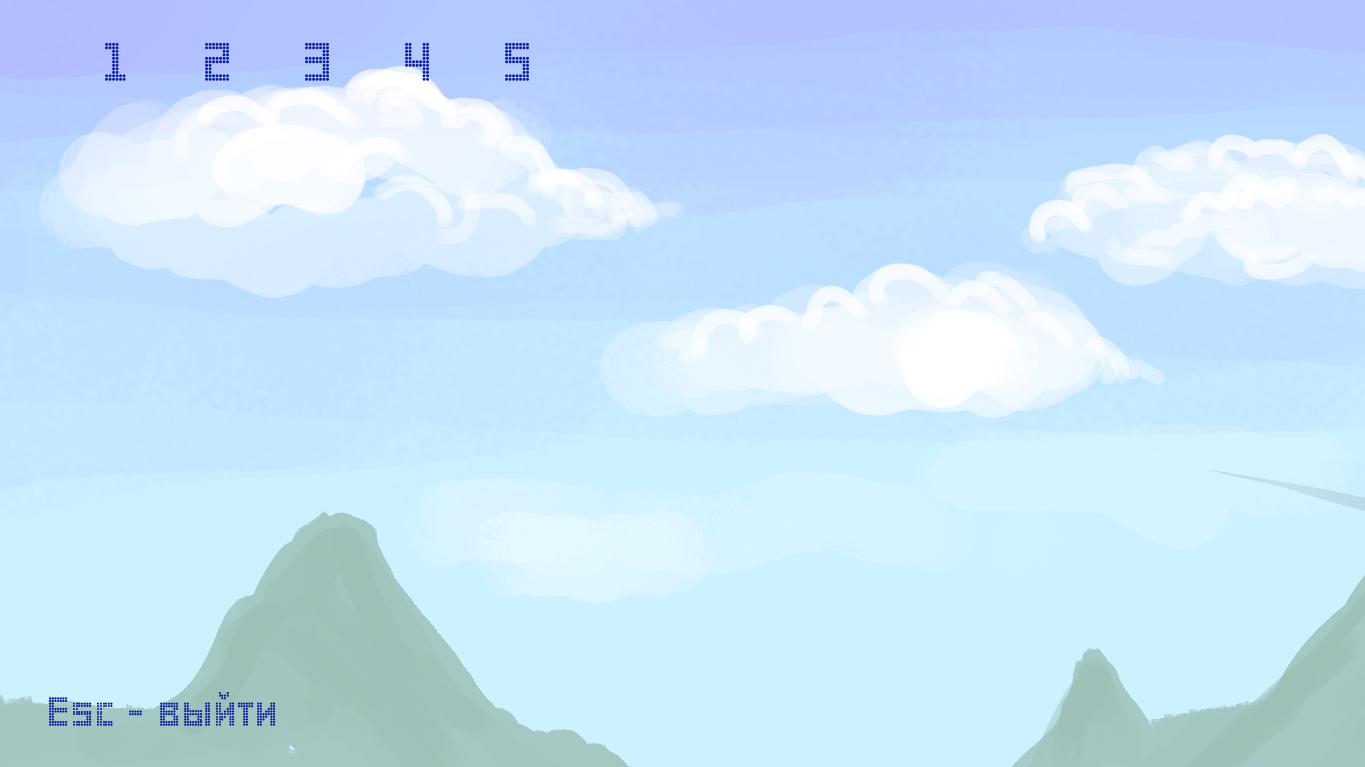


Рисунок 7 - Меню выбора уровня

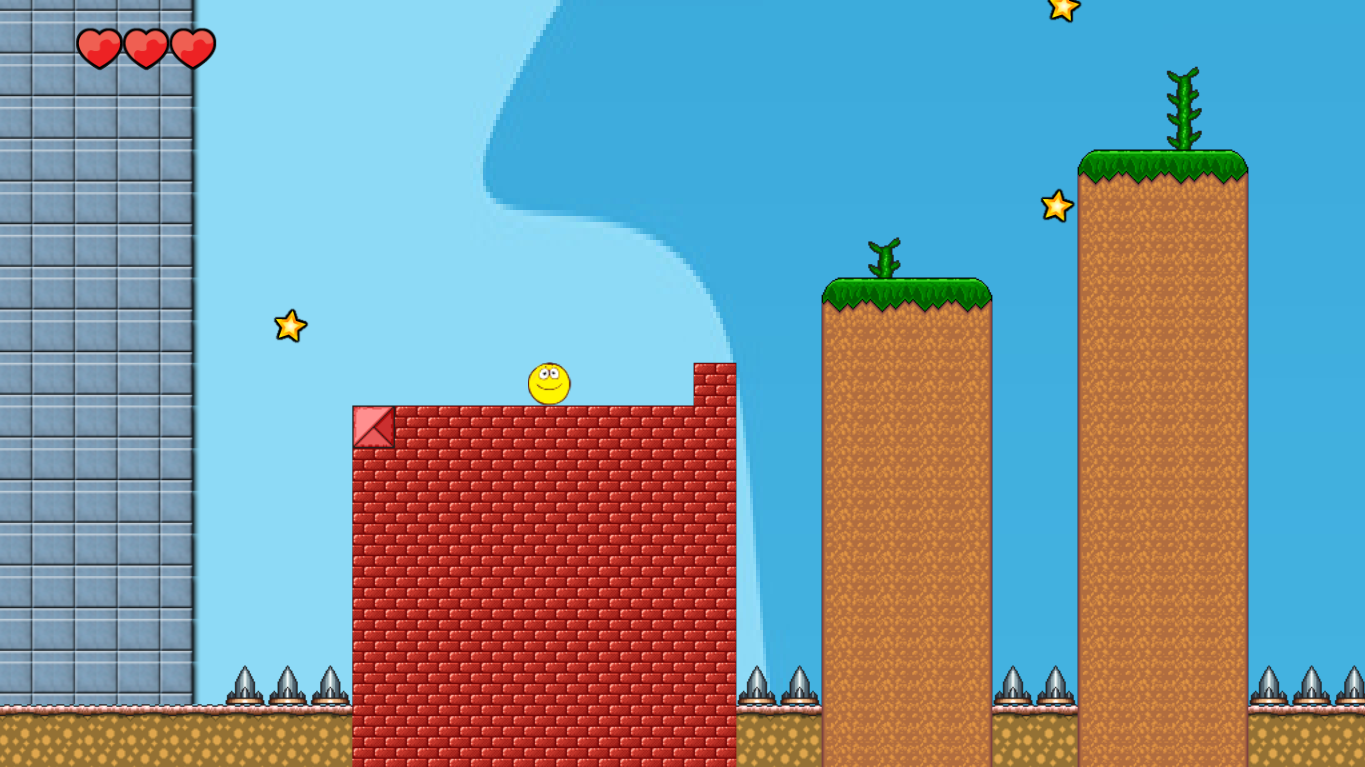


Рисунок 8 - Игровой процесс

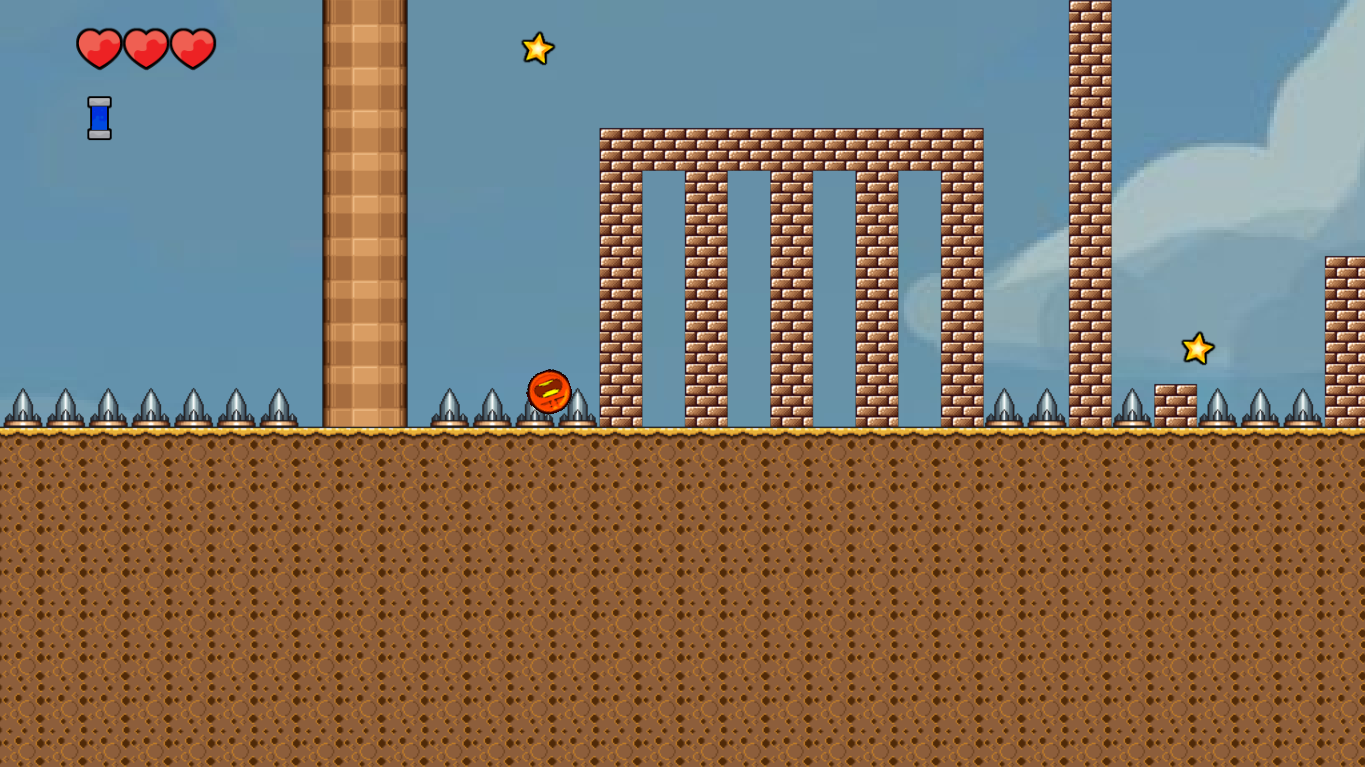


Рисунок 9 - Игровой процесс (смерть персонажа)



Рисунок 10 - Меню победы

При проведении полного тестирования был выполнен конкретный пример действий пользователя. Были описаны все действия пользователя и реакции программного продукта, которые были ожидаемы и соответствовали запланированным в процессе проектирования.

1. Применение.
   1. Назначение программы.

Данное приложение разработано для развлечения и ориентируются на развитие у игроков определенных знаний, навыков, способностей. Как правило, в такого рода играх от игрока требуется:

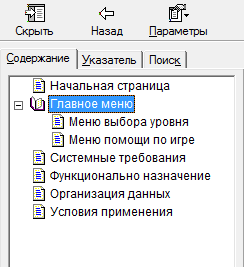
* владение средствами управления, быстрота и точность манипуляций;
* быстрая и правильная реакция на происходящие события;
* чувство времени, умение выдерживать заданные временные интервалы;
* способность следить за несколькими объектами одновременно;
* знание географии игрового поля, законов игрового мира;
* знание конкретной предметной области, которая моделируется в игре;
* умение искать закономерности;
* знание алгоритма и стратегии выигрыша;
* способность к быстрому и максимально полному перебору основных вариантов;
* память на текущие события;
* использование прошлого опыта, что происходило в предыдущих сеансах игры;
* способность интенсивно работать в течение всего сеанса игры.
  1. Условия применения.

Необходимо отметить, что данное программное средство не обладает средствами, ограничивающими доступ к нему, а также средствами, препятствующими свободному распространения программы. Для работы с приложением не требуется установка дополнительных компонентов и специальных технических и программных ресурсов.

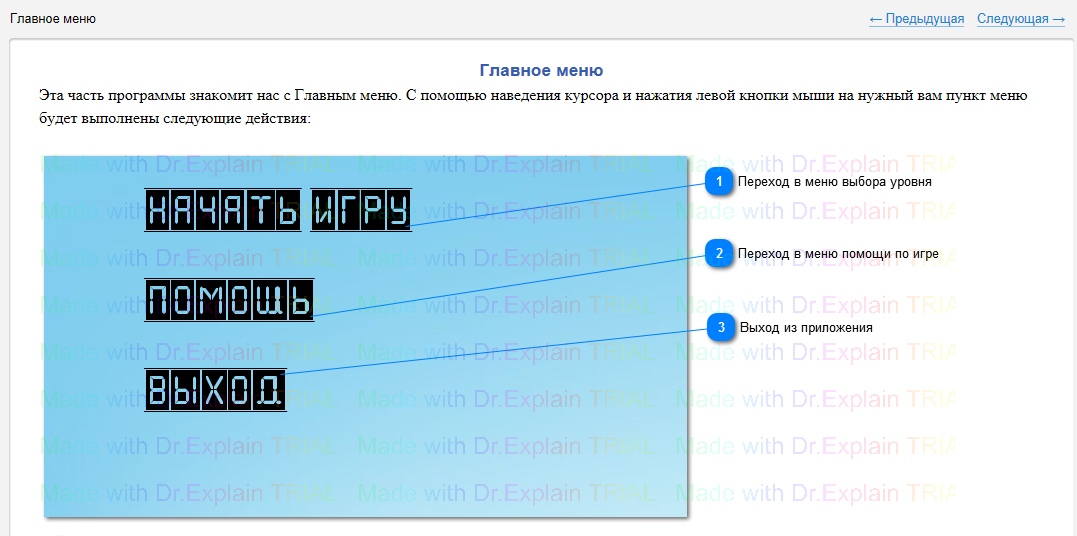
* 1. Справочная система.

Справочная система для данного программного средства довольно проста. Она представляет собой описание возможных действий для работы с программой. Вся справочная информация расположена в пункте главного меню «Справка».

Данная система позволяет пользователю получить конкретную информацию по интересующему вопросу, а также способам работы с программой. Каталог разделов справки отображен на рисунке Каталог разделов. Раздел справки «Главное меню» отображен на рисунке Раздел главного меню.



Каталог разделов



Раздел главного меню

Заключение

В рамках курсового проектирования по предмету «Основы алгоритмизации и программирование» было разработано программное приложение «Kolobok.exe». При помощи данной программы пользователь имеет возможность проверки своих навыков в игре «Kolobok».

Программное средство имеет ряд достоинств: простой и понятный интерфейс, небольшой объем памяти, занимаемый приложением на различных носителях информации, удобно для быстрой обработки информации.

В программе реализованы такие задачи, как:

* реализация 2D-платформера “Колобок”;
* чтение и запись данных файл;
* движение персонажа;
* уровни сложности;
* повышение характеристик персонажа;
* начисление бонусов;
* реализация жизней;
* сбор предметов.

В процессе курсового проектирования программное приложение прошло функциональное и полное тестирование, которые доказали корректность выполнения всех определенных функций, отсутствие сбоев и неадекватных реакций со стороны программы.

В процессе разработки программы использовался в большом объеме тематический материал и исследована предметная область разработки аналогичных программных средств.

Также использовался материал по программированию, что способствовало закреплению наработанных навыков и умений в этих интересных областях знаний.

Литература

1. Artur Moreira, Jan Haller, Henrik Vogelius Hansson. SFML Game Development. — Packt Publishing, 2013.
2. David Gouveia. 2. Audio Playback, Project 1 – building a simple audio manager // Getting Started with C++ Audio Programming for Game Development. — Packt Publishing, 2013.
3. Milcho G. Milchev. SFML Essentials. — Packt Publishing, 2015.
4. ГОСТ 2.106-96. ЕСКД. Текстовые документы.
5. ГОСТ 19.401-2000. ЕСПД. Текст программы.
6. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем.

Приложение А

(обязательное)

Текст программы

// файл исходного кода

Levels lvl; float timeb2 = 1.0f; SoundGame sound;

bool startGame()

{

soundBonusOff = true; // звук при активации бонуса

b2Vec2 Gravity(0.f, 9.8f); // величина гравитации

b2World World(Gravity); // переменная мира

fromSLMtoMM: // Из меню выбора уровня возвращаемся в главное меню

fromSLMtoMM = false;

if(toNextLvl == 0 && Restart == false) setMenu(window); // Если нам не нужно на следующий уровень

// и мы не перезапускаем его, то только тогда можем установить главное меню

else if (toNextLvl == 1) SelectLvl = true; // иначе если мы переходим на слеюущий уровень

if (exfromGame == false) return false; // выходим из игры

if (SelectLvl || Restart == true || toNextLvl == 2) {

if ((toNextLvl == 0 || toNextLvl == 2) && Restart == false) // если нам нужно на след. лвл и не перезапускаем текущий лвл

{ nlvl = setSLMenu(window); // номер уровня определяется возвращаемым значением из меню выбора уровня

if (fromSLMtoMM == true) goto fromSLMtoMM;} // если false, переходим по метке из меню выбора уровня в главное

else if (Restart == false) { if (nlvl < 5)nlvl++; }; // иначе если не перезапускаем уровень, тогда если лвл меньше 5

// инкрементируем переменную nlvl

switch (nlvl) // в зависимости от значения переменой, устанавливаем данные определённого уровня

{

case 1:lvl.setLevel1(World); break;

case 2:lvl.setLevel2(World); break;

case 3:lvl.setLevel3(World); break;

case 4:lvl.setLevel4(World); break;

case 5:lvl.setLevel5(World); break;

}

}

SelectLvl = false; Restart = false; // сбрасываем переменные

toNextLvl = 0;

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

window.setFramerateLimit(70); // устанавлиаваем лимит кадров в секунду(70)

view.reset(FloatRect(0, 0, 683 + 341.5f, 384 + 192.0f)); // устанавливаем вид камеры

Clock clock; //привязка ко времени(а не загруженности/мощности процессора).

Player MainPlayer("player.png", 17, 0, 160, 33, 33, 22, 12, World); // создаём главного персонажа

float CurrentFrame = 0; // Хранит текущий кадр

float time; // перменная времени

while (window.isOpen())

{

time = (float)clock.getElapsedTime().asMicroseconds(); //дать прошедшее время в микросекундах

if(slowmo == false)clock.restart(); // если бонус не активирован, перезапускаем время

//Чем больше делитель (сейчас это 800) тем больше действий и плавности будет в игре,

// но тем требовательнее она будет к ресурсам ПК.

time = time / 1; // (800)

Event e; // некоторое событие

while (window.pollEvent(e)) // пока приложение открыто

{

if (e.type == Event::Closed) // если тип события закрытый - выходим

window.close();

}

if (slowmo == true && time < 10000000.0f) // если активирован бонус, меняем значение переменной для замедления физики в игре

{

timeb2 = 0.5f;

}

if (time > 10000000.0f || slowmo == false) // если прошло 10 сек, сбрасываем данные для активации бонуса

{

slowmo = false;

timeb2 = 1.0f;

power = 0;

}

World.Step(timeb2 / 60.f, 8, 3); // Шаг в симуляции Box2D (действия физики за 1/60 секунды + параметры точности(по умолчанию))

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

if (Keyboard::isKeyPressed(Keyboard::Escape) || (toMainMenuisone == 1))// если нажата клаваша Esc

{ //или переменная для перехода в главное меню равна 1

if (toMainMenuisone == 1)// для перехода в главное меню путем установки определенных значений и выхода из внешнего тела

{

toMainMenu = 1; toMainMenuisone = 0;

window.clear(); break;

}

sound.soundButton(); // звук нажатия кнопки

setPauseMenu(window); // установка меню паузы

if ((toMainMenu == 1)) // для перехода в главное меню из меню паузы

{

toMainMenu = 1; toMainMenuisone = 0;

window.clear(); break;

}

}

// получаем кординаты нашего персонажа для камеры

getplayercoordinateforview(MainPlayer.sprite.getPosition().x, MainPlayer.sprite.getPosition().y);

window.setView(view);//"оживляем" камеру в окне sfml

window.clear(); // чистим окно

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

switch (nlvl) // отрисовываем уровень по дефолту

{

case 1:;

default:lvl.Drawlvl(World);

}

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

MainPlayer.Control(window, World); // главная функция персонажа, устанавливающая управление и взаимодействие с миром

if (toNextLvl == 1 || toNextLvl == 2) { window.clear(); break; } // если переходим на следущий уровень, прерываем

if (Restart == true) { window.clear(); break; } // fail menu RESTART

window.display(); // отображение отрисованых обьектов

}

return true; // по умолчанию возвращаем true

}

void gameRunning() {

if (startGame() && exfromGame) { gameRunning(); } // пока работает игровой цикл и не выходим из игры (exfromGame == true), продолжаем игру

}

void main()

{

gameRunning(); // функция работы игры

}

/\* Time SFML

Bремя обновляет игровую логику игры

и само оно так же постоянно обновляется (а если быть точнее перезапускается) в бесконечном цикле.

\*/

/\*

деструктор автоматически

вызывается компилятором в тот момент, когда объект выходит из области видимости.

\*/

//////////////////////////////////////////////////// реализация ИНТЕРФЕЙСА ЗВУКОВ

class SoundGame {

private:

Sound sound;

SoundBuffer buffer;

public:

void soundJump()

{

buffer.loadFromFile("music/jump.wav"); // загружаем в буфер sound

sound.setBuffer(buffer); //создаем звук и загружаем в него звук из буфера

sound.setVolume(50);

sound.play();

}

void soundWin()

{

buffer.loadFromFile("music/win.wav"); // загружаем в буфер sound

sound.setBuffer(buffer); //создаем звук и загружаем в него звук из буфера

sound.setVolume(50);

sound.play();

}

void soundDeath()

{

buffer.loadFromFile("music/death.wav"); // загружаем в буфер sound

sound.setBuffer(buffer); //создаем звук и загружаем в него звук из буфера

sound.setVolume(50);

sound.play();

}

void soundStar()

{

buffer.loadFromFile("music/star.wav"); // загружаем в буфер sound

sound.setBuffer(buffer); //создаем звук и загружаем в него звук из буфера

sound.setVolume(50);

sound.play();

}

void soundBonusOn()

{

buffer.loadFromFile("music/slowmo\_on.wav"); // загружаем в буфер sound

sound.setBuffer(buffer); //создаем звук и загружаем в него звук из буфера

sound.setVolume(50);

sound.play();

}

void soundButton()

{

buffer.loadFromFile("music/button.wav"); // загружаем в буфер sound

sound.setBuffer(buffer); //создаем звук и загружаем в него звук из буфера

sound.setVolume(60);

sound.play();

}

};

////////////////////////////////////////////////////КЛАСС ИГРОКА////////////////////////

class Player {

public:

Color color;

Texture texture, tHeart, tBonus;

Sprite sprite, sHeart, sBonus;

b2BodyDef bdef; // Определение тела блока в world

b2CircleShape circle; // ФОРМА ТЕЛА, тк перс - шар

b2Body \*pBody; // для создания тела в игровом мире

bool onGround; // для проверки находится ли персонаж на земле

b2Vec2 pos; // для получения кординат

b2Body\* it; // указатель не тело Box2D

b2Fixture \*f;

bool awake; // для корректной работы функций управления

b2Vec2 vel; // для считывания угла

float angVel;

b2Vec2 posP; // Считываем позицию

float angleP; // Считываем угол

b2Body\* itp; // для пробежки по телам

SoundGame soundg; // for sounds

Player (String F, int r, int rectLeft, int rectTop, int rectWidth, int rectHeight, float posX, float posY, b2World & World) {

texture.loadFromFile("images/" + F); // загружаем изображение

texture.setSmooth(true); // сглаживание

sprite.setTexture(texture); // установливаем спрайту изображение

sprite.setTextureRect(IntRect(rectLeft, rectTop, rectWidth, rectHeight));

sprite.setOrigin((float)rectWidth / 2, (float)rectHeight / 2); // переносим точку отсчёта в центр картинки

bdef.type = b2\_dynamicBody; // тип тела - динамическое

bdef.position.Set(posX, posY); // позиция персонажа в мире

circle.m\_radius = r / 30.f; // для масштабирования, ибо по умолчанию счёт в метрах, переводим в пиксели

pBody = World.CreateBody(&bdef); // Тело в мир

pBody->CreateFixture(&circle, 2); // СОЗДАТЬ ПРИВЯЗКУ !!! (2 параметр - плотность)

pBody->SetUserData("player"); // идентифицировать объект

///////////////////////////////////////////////////////////////////////////////////////////////////// Hearts

tHeart.loadFromFile("images/TileSet/hearts.png");

tHeart.setSmooth(true);

sHeart.setTexture(tHeart);

sHeart.setScale(0.7f, 0.7f); // устанавливаем масштаб спрайта

tBonus.loadFromFile("images/TileSet/hearts.png");

tBonus.setSmooth(true);

sBonus.setTexture(tBonus);

sBonus.setScale(0.7f, 0.7f);

}

void Control(RenderWindow & window, b2World & World) // for Draw

{

/////check if onGround//////

onGround = 0;

pos = pBody->GetPosition(); // Считываем позицию персонажа

pos.y += 18.5f / SCALE; // ТОЧКА прям под персонажем

for (it = World.GetBodyList(); it != 0; it = it->GetNext()) // Проходим в цикле по всем телам

for (f = it->GetFixtureList(); f != 0; f = f->GetNext()) // Для каждого тела проходим по всем прикреплённым к нему фигурам

if (f->TestPoint(pos)) onGround = true; // Если эта ТОЧКА попала внутрь

///////////////////////////// CONTROL ///////////////////////////////////////////////////////////////////////////////

awake = true;

vel = pBody->GetLinearVelocity(); // СЧИТЫВАНИЕ СКОРОСТИ

angVel = pBody->GetAngularVelocity(); // СЧИТЫВАНИЕ УГЛОВОЙ СКОРОСТИ

// ОГРАНИЧЕНИЕ СКОРОСТИ(+-20) // ПРИДАЁМ УСКОРЕНИЕ ПО X и Y (Torque) // Вращение персонажа с ограничением угловой скорости (Torque - вращать)

if (Keyboard::isKeyPressed(Keyboard::Right) || Keyboard::isKeyPressed(Keyboard::D))

if (vel.x < 5) {

pBody->ApplyForceToCenter(b2Vec2(80+(power/8), 0), awake);

if (angVel < 10 / 2) pBody->ApplyTorque(10, NULL);

}

if (Keyboard::isKeyPressed(Keyboard::Left) || Keyboard::isKeyPressed(Keyboard::A))

if (vel.x > -5) {

pBody->ApplyForceToCenter(b2Vec2(-80-(power/8), 0), awake);

if (angVel > -10 / 2) pBody->ApplyTorque(-10, NULL);

}

if (Keyboard::isKeyPressed(Keyboard::Up) || Keyboard::isKeyPressed(Keyboard::W) || Keyboard::isKeyPressed(Keyboard::Space))// JUMP

{

if (onGround) {

soundg.soundJump();

pBody->ApplyForceToCenter(b2Vec2(0, -1000-(power/2+(power/4))), awake);

onGround = false;

}

}

if (Keyboard::isKeyPressed(Keyboard::Down) || Keyboard::isKeyPressed(Keyboard::S))

{

pBody->SetAwake(true);

pBody->ApplyForceToCenter(b2Vec2(0, 20+(power/9)), awake);

}

//////////////////////////////////////// DRAW ///////////////////////////////////////////////////////////////////////////////////

for (this->itp = World.GetBodyList(); this->itp != 0; this->itp = this->itp->GetNext()) // пробегаем по всем телам (GetBodyList())

{

this->posP = this->itp->GetPosition(); // Считываем позицию

this->angleP = this->itp->GetAngle(); // Считываем угол

if (this->itp->GetUserData() == "player")

{

this->sprite.setPosition(this->posP.x\*SCALE, this->posP.y\*SCALE);

this->sprite.setRotation(this->angleP\*DEG);

window.draw(this->sprite);

}

}

//////////////////////////////////////////////////// WIN ////////////////////////////////////////////////////////////////////////////

switch (nlvl) // проверка в зависимости от номера уровня. Если спрайт персонажа расположжен дальше определенной позиции,

{ // устанавливаем меню пройденного уровня и проигрываем звук

case 2:

if (this->sprite.getPosition().x > 3564)

{

soundg.soundWin();

setWinMenu(window);

}

; break;

case 1:

if (this->sprite.getPosition().x > 3375)

{

soundg.soundWin();

setWinMenu(window);

}

; break;

case 3:

if (this->sprite.getPosition().x > 3354)

{

soundg.soundWin();

setWinMenu(window);

}

; break;

case 4:

if (this->sprite.getPosition().x > 3356)

{

soundg.soundWin();

setWinMenu(window);

}

; break;

case 5:

if (this->sprite.getPosition().x > 3545)

{

soundg.soundWin();

setWinMenu(window);

}

; break;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// DEATH

switch (nlvl) // реализация гибели персонажа, жизней, активации бонуса.

{

case 2:

if ((this->sprite.getPosition().y > 777) || (this->sprite.getPosition().x > 1412 && this->sprite.getPosition().x < 1435 && this->sprite.getPosition().y>649) || (this->sprite.getPosition().x > 1540 && this->sprite.getPosition().x < 1562 && this->sprite.getPosition().y>649) || (this->sprite.getPosition().x > 2820 && this->sprite.getPosition().x < 2843 && this->sprite.getPosition().y>678) || (this->sprite.getPosition().x > 3428 && this->sprite.getPosition().x < 3451 && this->sprite.getPosition().y>681))

{

soundg.soundDeath();

for (int i = 0; i < 10; i++) {

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Red); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Blue); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Cyan); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Yellow); window.draw(this->sprite); window.display();

}

switch (numh)

{

case 0:heart3 = false; Restart = true; break;

case 1:heart2 = false; Restart = true; break;

case 2:heart1 = false; break;

}

numh++;

slowmo = false;

}

; break;

case 1:

if ((this->sprite.getPosition().x > 1310 && this->sprite.getPosition().x < 1344 && this->sprite.getPosition().y>426) || (this->sprite.getPosition().x > 3200 && this->sprite.getPosition().x < 3260 && this->sprite.getPosition().y>615) || (this->sprite.getPosition().x > 3011 && this->sprite.getPosition().x < 3068 && this->sprite.getPosition().y>615) || (this->sprite.getPosition().x > 2852 && this->sprite.getPosition().x < 2908 && this->sprite.getPosition().y>615) || (this->sprite.getPosition().x > 2756 && this->sprite.getPosition().x < 2780 && this->sprite.getPosition().y>615) || (this->sprite.getPosition().y > 776))

{

soundg.soundDeath();

for (int i = 0; i < 10; i++) {

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Red); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Blue); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Cyan); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Yellow); window.draw(this->sprite); window.display();

}

switch (numh)

{

case 0:heart3 = false; Restart = true; break;

case 1:heart2 = false; Restart = true; break;

case 2:heart1 = false; break;

}

numh++;

slowmo = false;

}

; break;

case 3:

if ((this->sprite.getPosition().x > 705 && this->sprite.getPosition().x < 732 && this->sprite.getPosition().y>550) || (this->sprite.getPosition().x > 993 && this->sprite.getPosition().x < (993 + 32) && this->sprite.getPosition().y < 441) || (this->sprite.getPosition().x > 1154 && this->sprite.getPosition().x < (1154 + 32) && this->sprite.getPosition().y < 441) || (this->sprite.getPosition().x > 1313 && this->sprite.getPosition().x < (1313 + 32) && this->sprite.getPosition().y < 441) || (this->sprite.getPosition().x > 1058 && this->sprite.getPosition().x < (1058 + 30) && this->sprite.getPosition().y>550) || (this->sprite.getPosition().y > 776) || (this->sprite.getPosition().x > 1155 && this->sprite.getPosition().x < (1155 + (31 \* 3)) && this->sprite.getPosition().y>550) || (this->sprite.getPosition().x > 1346 && this->sprite.getPosition().x < (1346 + 30) && this->sprite.getPosition().y>550) || (this->sprite.getPosition().x > 1858 && this->sprite.getPosition().x < (1858 + 30) && this->sprite.getPosition().y>550) || (this->sprite.getPosition().x > 2020 && this->sprite.getPosition().x < (2020 + (31 \* 3)) && this->sprite.getPosition().y < (513 + 31)) || (this->sprite.getPosition().x > 2049 && this->sprite.getPosition().x < (2049 + 30) && this->sprite.getPosition().y < 572) || (this->sprite.getPosition().x > 3013 && this->sprite.getPosition().x < (3013 + 30) && this->sprite.getPosition().y < 510))

{

soundg.soundDeath();

for (int i = 0; i < 10; i++) {

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Red); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Blue); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Cyan); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Yellow); window.draw(this->sprite); window.display();

}

switch (numh)

{

case 0:heart3 = false; Restart = true; break;

case 1:heart2 = false; Restart = true; break;

case 2:heart1 = false; break;

}

numh++;

slowmo = false;

}

; break;

case 4:

if ((this->sprite.getPosition().x > 2338 && this->sprite.getPosition().x < (2338 + 31) && this->sprite.getPosition().y>710) || (this->sprite.getPosition().x > 2209 && this->sprite.getPosition().x < (2209 + 31) && this->sprite.getPosition().y>710) || (this->sprite.getPosition().x > 2595 && this->sprite.getPosition().x < (2595 + 31) && this->sprite.getPosition().y>704 - 31) || (this->sprite.getPosition().x > 2529 && this->sprite.getPosition().x < (2529 + 31) && this->sprite.getPosition().y>704 - 31) || (this->sprite.getPosition().x > 994 && this->sprite.getPosition().x < (994 + 31) && this->sprite.getPosition().y>615) || (this->sprite.getPosition().x > 1090 && this->sprite.getPosition().x < (1090 + 32 \* 3) && this->sprite.getPosition().y>615 && this->sprite.getPosition().y < 640) || (this->sprite.getPosition().x > 1347 && this->sprite.getPosition().x < (1347 + 31 \* 2) && this->sprite.getPosition().y>710) || (this->sprite.getPosition().x > 1475 - 31 && this->sprite.getPosition().x < (1475 + 31) && this->sprite.getPosition().y>710) || (this->sprite.getPosition().x > 1522 && this->sprite.getPosition().x < (1522 + 31) && this->sprite.getPosition().y>326 && this->sprite.getPosition().y < 366) || (this->sprite.getPosition().y > 776) || (this->sprite.getPosition().x > 1954 && this->sprite.getPosition().x < (1954 + 31 \* 2) && this->sprite.getPosition().y>358) || (this->sprite.getPosition().x > 2015 && this->sprite.getPosition().x < (2015 + 31) && this->sprite.getPosition().y>386) || (this->sprite.getPosition().x > 2117 && this->sprite.getPosition().x < (2117 + 30) && this->sprite.getPosition().y>450 && this->sprite.getPosition().y < 511) || (this->sprite.getPosition().x > 2146 && this->sprite.getPosition().x < (2146 + (31 \* 3)) && this->sprite.getPosition().y>420 && this->sprite.getPosition().y < 450) || (this->sprite.getPosition().x > 2210 && this->sprite.getPosition().x < (2210 + 30) && this->sprite.getPosition().y < 440) || (this->sprite.getPosition().x > 2274 && this->sprite.getPosition().x < (2274 + 31) && this->sprite.getPosition().y>710) || (this->sprite.getPosition().x > 2914 && this->sprite.getPosition().x < (2914 + 31 \* 2) && this->sprite.getPosition().y>675) || (this->sprite.getPosition().x > 3040 && this->sprite.getPosition().x < (3040 + 31 \* 2) && this->sprite.getPosition().y>675) || (this->sprite.getPosition().x > 3170 && this->sprite.getPosition().x < (3170 + 31 \* 2) && this->sprite.getPosition().y>675) || (this->sprite.getPosition().x > 3266 && this->sprite.getPosition().x < (3266 + 31) && this->sprite.getPosition().y>675))

{

soundg.soundDeath();

for (int i = 0; i < 10; i++) {

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Red); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Blue); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Cyan); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Yellow); window.draw(this->sprite); window.display();

}

switch (numh)

{

case 0:heart3 = false; Restart = true; break;

case 1:heart2 = false; Restart = true; break;

case 2:heart1 = false; break;

}

numh++; slowmo = false;

}

; break;

case 5:

if ((this->sprite.getPosition().x < 541 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 898 && this->sprite.getPosition().x < 957 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 994 && this->sprite.getPosition().x < 1117 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 1218 && this->sprite.getPosition().x < 1341 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 1410 && this->sprite.getPosition().x < 1502 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 1538 && this->sprite.getPosition().x < 1597 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 1666 && this->sprite.getPosition().x < 1694 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 1730 && this->sprite.getPosition().x < 1821 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 1858 && this->sprite.getPosition().x < 1886 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 1922 && this->sprite.getPosition().x < 2014 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 2082 && this->sprite.getPosition().x < 2238 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 2306 && this->sprite.getPosition().x < 2430 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 2467 && this->sprite.getPosition().x < 2589 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 2627 && this->sprite.getPosition().x < 2686 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 2723 && this->sprite.getPosition().x < 2844 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 2883 && this->sprite.getPosition().x < 3005 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 3075 && this->sprite.getPosition().x < 3197 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x > 3266 && this->sprite.getPosition().x < 3357 && this->sprite.getPosition().y>772) || (this->sprite.getPosition().x < 3545 && this->sprite.getPosition().y>997))

{

soundg.soundDeath();

for (int i = 0; i < 10; i++) {

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Red); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Blue); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(32, 63, 33, 33)); this->sprite.setColor(Color::Cyan); window.draw(this->sprite); window.display();

this->sprite.setTextureRect(IntRect(198, 64, 33, 33)); this->sprite.setColor(Color::Yellow); window.draw(this->sprite); window.display();

}

switch (numh)

{

case 0:heart3 = false; Restart = true; break;

case 1:heart2 = false; Restart = true; break;

case 2:heart1 = false; break;

}

numh++; slowmo = false;

}

; break;

}

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

//////////////////////////////////////// DRAW HEARTS ///////////////////////////////////////////////////////////////////////////////////

// отрисовка сердце относительно камеры при условии утрачено оно или нет

if (heart1 == true) {

sHeart.setTextureRect(IntRect(320, 70, 60, 70));

this->sHeart.setPosition(view.getCenter().x-460, view.getCenter().y-270);

window.draw(this->sHeart);

}

else

{

sHeart.setTextureRect(IntRect(240, 67, 80, 125 - 68));

this->sHeart.setPosition(view.getCenter().x-460, view.getCenter().y-275);

window.draw(this->sHeart);

}

if (heart2 == true) {

sHeart.setTextureRect(IntRect(320, 70, 60, 70));

this->sHeart.setPosition(view.getCenter().x - 460+35, view.getCenter().y - 270);

window.draw(this->sHeart);

}

else

{

sHeart.setTextureRect(IntRect(240, 67, 80, 125 - 68));

this->sHeart.setPosition(view.getCenter().x - 460+35, view.getCenter().y - 275);

window.draw(this->sHeart);

}

if (heart3 == true) {

sHeart.setTextureRect(IntRect(320, 70, 60, 70));

this->sHeart.setPosition(view.getCenter().x - 460+70, view.getCenter().y - 270);

window.draw(this->sHeart);

}

else

{

sHeart.setTextureRect(IntRect(240, 67, 80, 125 - 68));

this->sHeart.setPosition(view.getCenter().x - 460+70, view.getCenter().y - 275);

window.draw(this->sHeart);

}

if (heart1 == false && heart2 == false && heart3 == false) {

setFailMenu(window);

heart1 = true; heart2 = true; heart3 = true; numh = 0;

}

//////////////////////////////////////// STARS ///////////////////////////////////////////////////////////////////////////////////

// Установка звезд в зависимости от уровня

switch (nlvl)

{

case 1:

if (this->sprite.getPosition().x > 440 && this->sprite.getPosition().x < 450 + 60 && this->sprite.getPosition().y > 490 && this->sprite.getPosition().y < 490 + 40) Stars[0] = false;

if (this->sprite.getPosition().x > 1015 && this->sprite.getPosition().x < 1025 + 60 && this->sprite.getPosition().y > 400 && this->sprite.getPosition().y < 400 + 40) Stars[1] = false;

if (this->sprite.getPosition().x > 1020 && this->sprite.getPosition().x < 1030 + 60 && this->sprite.getPosition().y > 250 && this->sprite.getPosition().y < 250 + 40) Stars[2] = false;

if (this->sprite.getPosition().x > 1740 && this->sprite.getPosition().x < 1740 + 60 && this->sprite.getPosition().y > 465 && this->sprite.getPosition().y < 465 + 40) Stars[3] = false;

if (this->sprite.getPosition().x > 2690 && this->sprite.getPosition().x < 2690 + 60 && this->sprite.getPosition().y > 535 && this->sprite.getPosition().y < 535 + 40) Stars[4] = false;

break;

case 2:

if (this->sprite.getPosition().x > 440 && this->sprite.getPosition().x < 450 + 60 && this->sprite.getPosition().y > 490 && this->sprite.getPosition().y < 490 + 40) Stars[0] = false;

if (this->sprite.getPosition().x > 935 && this->sprite.getPosition().x < 945 + 60 && this->sprite.getPosition().y > 715 && this->sprite.getPosition().y < 715 + 40) Stars[1] = false;

if (this->sprite.getPosition().x > 1645 && this->sprite.getPosition().x < 1655 + 60 && this->sprite.getPosition().y > 600 && this->sprite.getPosition().y < 600 + 40) Stars[2] = false;

if (this->sprite.getPosition().x > 2565 && this->sprite.getPosition().x < 2575 + 60 && this->sprite.getPosition().y > 575 && this->sprite.getPosition().y < 575 + 40) Stars[3] = false;

if (this->sprite.getPosition().x > 3075 && this->sprite.getPosition().x < 3085 + 60 && this->sprite.getPosition().y > 665 && this->sprite.getPosition().y < 665 + 40) Stars[4] = false;

break;

case 3:

if (this->sprite.getPosition().x > 605 && this->sprite.getPosition().x < 615 + 60 && this->sprite.getPosition().y > 630 && this->sprite.getPosition().y < 630 + 40) Stars[0] = false;

if (this->sprite.getPosition().x > 515 && this->sprite.getPosition().x < 525 + 60 && this->sprite.getPosition().y > 530 && this->sprite.getPosition().y < 530 + 40) Stars[1] = false;

if (this->sprite.getPosition().x > 1190 && this->sprite.getPosition().x < 1200 + 60 && this->sprite.getPosition().y > 470 && this->sprite.getPosition().y < 470 + 40) Stars[2] = false;

if (this->sprite.getPosition().x > 1740 && this->sprite.getPosition().x < 1750 + 60 && this->sprite.getPosition().y > 465 && this->sprite.getPosition().y < 465 + 40) Stars[3] = false;

if (this->sprite.getPosition().x > 1706 && this->sprite.getPosition().x < 1716 + 60 && this->sprite.getPosition().y > 475 && this->sprite.getPosition().y < 475 + 40) Stars[4] = false;

break;

case 4:

if (this->sprite.getPosition().x > 1150 && this->sprite.getPosition().x < 1160 + 60 && this->sprite.getPosition().y > 690 && this->sprite.getPosition().y < 690 + 40) Stars[0] = false;

if (this->sprite.getPosition().x > 1415 && this->sprite.getPosition().x < 1425 + 60 && this->sprite.getPosition().y > 650 && this->sprite.getPosition().y < 650 + 40) Stars[1] = false;

if (this->sprite.getPosition().x > 1870 && this->sprite.getPosition().x < 1880 + 60 && this->sprite.getPosition().y > 300 && this->sprite.getPosition().y < 300 + 40) Stars[2] = false;

if (this->sprite.getPosition().x > 2305 && this->sprite.getPosition().x < 2315 + 60 && this->sprite.getPosition().y > 670 && this->sprite.getPosition().y < 670 + 40) Stars[3] = false;

if (this->sprite.getPosition().x > 3233 && this->sprite.getPosition().x < 3243 + 60 && this->sprite.getPosition().y > 660 && this->sprite.getPosition().y < 660 + 40) Stars[4] = false;

break;

}

for (int i = 0; i < 5; i++) // берём звезду - проигрывается звук

{

if (Stars[i] == false && soundStars[i] == true)

{

soundg.soundStar(); soundStars[i] = false;

}

}

//////////////////////////////////////// BONUSES ///////////////////////////////////////////////////////////////////////////////////

// Установка, отрисовка бонусов в зависимости от их количества

if (bonuses == 1 || bonuses == 2 || bonuses == 3 || bonuses == 4) {

sBonus.setTextureRect(IntRect(320, 388, 60, 70));

this->sBonus.setPosition(view.getCenter().x - 460, view.getCenter().y - 220);

window.draw(this->sBonus);

}

if (bonuses == 2 || bonuses == 3 || bonuses == 4) {

sBonus.setTextureRect(IntRect(320, 388, 60, 70));

this->sBonus.setPosition(view.getCenter().x - 430, view.getCenter().y - 220);

window.draw(this->sBonus);

}

if (bonuses == 3 || bonuses == 4) {

sBonus.setTextureRect(IntRect(320, 388, 60, 70));

this->sBonus.setPosition(view.getCenter().x - 400, view.getCenter().y - 220);

window.draw(this->sBonus);

}

if (bonuses == 4) {

sBonus.setTextureRect(IntRect(320, 388, 60, 70));

this->sBonus.setPosition(view.getCenter().x - 370, view.getCenter().y - 220);

window.draw(this->sBonus);

}

if(bonuses > 0){ // активация бонуса

if (Keyboard::isKeyPressed(Keyboard::F))

{

bonuses--; slowmo = true; power = 2000;

soundg.soundBonusOn();

}

}

}

};

// Реализация главного меню

void setMenu(RenderWindow & window)

{

slowmo = false;

for (int i = 0; i < 5; i++)

{

Stars[i] = true;

soundStars[i] = true;

}

Texture menuTexture1, menuTexture2, menuTexture3, aboutTexture, menuBackground, tcontrolhelp, tesc;

menuTexture1.loadFromFile("images/Menu/StartGame.png");

menuTexture2.loadFromFile("images/Menu/Help.png");

menuTexture3.loadFromFile("images/Menu/Exit.png");

aboutTexture.loadFromFile("images/Menu/helpmenu.png");

menuBackground.loadFromFile("images/Menu/BackgrMenu.jpg");

tcontrolhelp.loadFromFile("images/Menu/controlhelp.png");

tesc.loadFromFile("images/Menu/esc\_exit.png");

Sprite menu1(menuTexture1), menu2(menuTexture2), menu3(menuTexture3), about(aboutTexture), menuBg(menuBackground), scontrolhelp(tcontrolhelp), sesc(tesc);

bool isMenu = 1;

int menuNum = 0;

menuTexture1.setSmooth(true);

menuTexture2.setSmooth(true);

menuTexture3.setSmooth(true);

menuBg.setPosition(0, 0);

menuBg.setOrigin(0, 0);

menuBg.setScale(1.1f, 1.1f); // !!!!!!!!!

View view;

//////////////////////////////МЕНЮ///////////////////

while (isMenu)

{

menu1.setPosition(Vector2f(100, 30));

menu2.setPosition(Vector2f(100, 120));

menu3.setPosition(Vector2f(100, 210));

view.reset(FloatRect(0, 0, 1366, 768)); // без бага с съезжающей камерой

window.setView(view);

menuNum = 0;

window.clear();//Color(129, 181, 221)

if (IntRect(100, 30, 280, 50).contains(Mouse::getPosition(window)))

{

menuNum = 1;

menu1.setPosition(100 + 8, 30 + 8);

}

if (IntRect(100, 120, 205, 50).contains(Mouse::getPosition(window)))

{

menuNum = 2;

menu2.setPosition(100 + 8, 120 + 8);

}

if (IntRect(100, 210, 150, 50).contains(Mouse::getPosition(window))) // 100, 210, 150, 50

{

menuNum = 3;

menu3.setPosition(100 + 8, 210 + 8);

}

if (Mouse::isButtonPressed(Mouse::Left))

{

if (menuNum == 1) { isMenu = false; SelectLvl = true; }// выходим из меню

if (menuNum == 2) { // вызов меню ознаколения с управлением

window.draw(menuBg);

about.setPosition(45, 35);

window.draw(about);

scontrolhelp.setPosition(45, 100);

window.draw(scontrolhelp);

sesc.setPosition(45, 690);

window.draw(sesc);

window.display();

while (!Keyboard::isKeyPressed(Keyboard::Escape));

}

if (menuNum == 3) { isMenu = false; exfromGame = false; window.close(); } // выход из приложения

}

window.draw(menuBg);

window.draw(menu1);

window.draw(menu2);

window.draw(menu3);

window.display();

}

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

}

// Установка статического обьекта

void setRTWall(int x, int y, int w, int h, b2World & World)

{

x = x + (w/2) + 1;

y = y + 11 + h;

w = w / 2;

h = h + 10;

b2PolygonShape gr; // Многоугольник

gr.SetAsBox(w / SCALE, h / SCALE);

b2BodyDef bdef; // Создание тела

bdef.position.Set(x / SCALE, y / SCALE);

b2Body \*b\_ground = World.CreateBody(&bdef); // Само тело, отображаемое в мире

b\_ground->CreateFixture(&gr, 1); // прикрепляем форму(shape)

}

void getplayercoordinateforview(float x, float y) { // для считывания координат игрока

view.setCenter(x+100, y); // передавая его координаты камере.

}

RenderWindow window(VideoMode(1366, 768, 32), "Kolobok", Style::Fullscreen); // Создание окна с именем, разрешением и режимом

// Реализация меню победы

void setWinMenu(RenderWindow & window) {

//// bonuses ////

if(nlvl<5)

access[nlvl] = true;

slowmo = false;

if (Stars[0] == false && Stars[1] == false && Stars[2] == false && Stars[3] == false && Stars[4] == false)

if (bonuses < 4) bonuses++;

for (int i = 0; i < 5; i++) {

Stars[i] = true;

soundStars[i] = true;

}

heart1 = true; heart2 = true; heart3 = true;

hearts[0] = true; hearts[1] = true; hearts[2] = true; numh = 0;

Texture menuTexture1, menuTexture2, menuBackground, menulvlcom, texit;

texit.loadFromFile("images/Menu/Exit2.png");

menuTexture1.loadFromFile("images/Menu/nextlevel.png");

menuTexture2.loadFromFile("images/Menu/SelectLvl.png");

menulvlcom.loadFromFile("images/Menu/lvlcomplete.png");

menuBackground.loadFromFile("images/Menu/BackgrSelection.png");

Sprite menu1(menuTexture1), menu2(menuTexture2), menuBg(menuBackground), lvlcom(menulvlcom);

bool isMenu = 1;

int menuNum = 0;

menuTexture1.setSmooth(true);

menuTexture2.setSmooth(true);

menuBackground.setSmooth(true);

menulvlcom.setSmooth(true);

menuBg.setPosition(0, 0);

View view;

while (isMenu)

{

menu1.setScale(0.9f, 0.9f); menu1.setPosition(100, 60);

menu2.setScale(0.9f, 0.9f); menu2.setPosition(100, 170);

lvlcom.setPosition(500, 580);

view.reset(FloatRect(0, 0, 1366, 768));

window.setView(view);

menuNum = 0;

window.clear();

if (IntRect(100, 60, 340, 54).contains(Mouse::getPosition(window)))

{

menuNum = 1;

menu1.setPosition(100 + 8, 60 + 8);

}

if (IntRect(100, 170, 386, 64).contains(Mouse::getPosition(window)))

{

menuNum = 2;

menu2.setPosition(100 + 8, 170 + 8);

}

if (Mouse::isButtonPressed(Mouse::Left))

{

if (menuNum == 1) { isMenu = false; toNextLvl = 1; }// переход на следующий лвл

if (menuNum == 2) { isMenu = false; toNextLvl = 2; break; } // выход в главное меню

}

window.draw(menuBg);

window.draw(menu1);

window.draw(menu2);

window.draw(lvlcom);

window.display();

}

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

}

// Реализация меню выбора уровня

int setSLMenu(RenderWindow & window) {

Texture one, two, three, four, five, menuBackground, tesc;

one.loadFromFile("images/Menu/one.png");

two.loadFromFile("images/Menu/two.png");

three.loadFromFile("images/Menu/three.png");

four.loadFromFile("images/Menu/four.png");

five.loadFromFile("images/Menu/five.png");

tesc.loadFromFile("images/Menu/esc\_exit.png");

menuBackground.loadFromFile("images/Menu/BackgrSelection.png");

Sprite menu1(one), menu2(two), menu3(three), menu4(four), menu5(five), menuBg(menuBackground), sesc(tesc);

bool isMenu = 1;

int menuNum = 0;

one.setSmooth(true);

two.setSmooth(true);

three.setSmooth(true);

four.setSmooth(true);

five.setSmooth(true);

menuBg.setPosition(0, 0);

menuBg.setOrigin(0, 0);

menuBg.setScale(1.0f, 1.0f); // !!!!!!!!!

View view;

//////////////////////////////МЕНЮ///////////////////

while (isMenu)

{

menu1.setPosition(100, 40);

menu2.setPosition(200, 40);

menu3.setPosition(300, 40);

menu4.setPosition(400, 40);

menu5.setPosition(500, 40);

sesc.setPosition(45, 690);

view.reset(FloatRect(0, 0, 1366, 768));

window.setView(view);

menuNum = 0;

window.clear();

if (IntRect(100, 40, 40, 60).contains(Mouse::getPosition(window)))

{

menuNum = 1;

menu1.setPosition(100 + 8, 40 + 8);

}

if (IntRect(200, 40, 40, 60).contains(Mouse::getPosition(window)))

{

menuNum = 2;

menu2.setPosition(200 + 8, 40 + 8);

}

if (IntRect(300, 40, 40, 60).contains(Mouse::getPosition(window)))

{

menuNum = 3;

menu3.setPosition(300 + 8, 40 + 8);

}

if (IntRect(400, 40, 40, 60).contains(Mouse::getPosition(window)))

{

menuNum = 4;

menu4.setPosition(400 + 8, 40 + 8);

}

if (IntRect(500, 40, 40, 60).contains(Mouse::getPosition(window)))

{

menuNum = 5;

menu5.setPosition(500 + 8, 40 + 8);

}

if (Mouse::isButtonPressed(Mouse::Left)) // выбор уровня

{

switch (menuNum)

{

case 1: if (access[0] == true) return 1; break;

case 2: if (access[1] == true)return 2; break;

case 3: if (access[2] == true)return 3; break;

case 4: if (access[3] == true)return 4; break;

case 5: if (access[4] == true)return 5; break;

}

}

if (Keyboard::isKeyPressed(Keyboard::Escape)) { isMenu = false; fromSLMtoMM = true; break; } // выход в главное меню

window.draw(menuBg);

window.draw(menu1);

window.draw(menu2);

window.draw(menu3);

window.draw(menu4);

window.draw(menu5);

window.draw(sesc);

window.display();

}

return 1;

}

// Реализация меню паузы

void setPauseMenu(RenderWindow & window) {

toMainMenu = 0;

Texture menuTexture1, menuTexture2, menuBackground;

menuTexture1.loadFromFile("images/Menu/Continue.png");

menuTexture2.loadFromFile("images/Menu/ExitPM.png");

menuBackground.loadFromFile("images/Menu/BackgrSelection.png");

Sprite menu1(menuTexture1), menu2(menuTexture2), menuBg(menuBackground);

bool isMenu = 1;

int menuNum = 0;

menuTexture1.setSmooth(true);

menuTexture2.setSmooth(true);

menuBackground.setSmooth(true);

menuBg.setPosition(0, 0);

View view;

//////////////////////////////МЕНЮ///////////////////

while (isMenu)

{

menu1.setPosition(Vector2f(100, 60));

menu2.setPosition(Vector2f(100, 140));

view.reset(FloatRect(0, 0, 1366, 768)); // без бага с съезжающей камерой

window.setView(view);

menuNum = 0;

window.clear();//Color(129, 181, 221)

if (IntRect(100, 60, 270, 50).contains(Mouse::getPosition(window)))

{

menuNum = 1;

for (float i = 0; i < 8; i++) menu1.setPosition(100 + i, 60 + i);

}

if (IntRect(100, 140, 160, 50).contains(Mouse::getPosition(window)))

{

menuNum = 2;

for (float i = 0; i < 8; i++) menu2.setPosition(100 + i, 140 + i);

}

if (Mouse::isButtonPressed(Mouse::Left))

{

if (menuNum == 1) { isMenu = false;}// выходим из меню

if (menuNum == 2) {

isMenu = false; toMainMenu = 1; numh = 0;

for (int i = 0; i < 5; i++) { // сброс параметров

Stars[i] = true;

soundStars[i] = true;

}

heart1 = true; heart2 = true; heart3 = true; // сброс сердец

}

}

window.draw(menuBg);

window.draw(menu1);

window.draw(menu2);

window.display();

}

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

}

// Реализация меню поражения

void setFailMenu(RenderWindow & window) {

slowmo = false;

for (int i = 0; i < 5; i++) Stars[i] = true; // сбрасывание звёзд (теперь их снова можно собрать)

numh = 0;

// загразука текстур и установка спрайтов, масштаба, сглаживания

Texture menuTexture1, menuTexture2, menuBackground, menuBg2;

menuTexture1.loadFromFile("images/Menu/Exit2.png");

menuTexture2.loadFromFile("images/Menu/Restart.png");

menuBackground.loadFromFile("images/Menu/BackgrSelection.png");

Sprite menu1(menuTexture1), menu2(menuTexture2), menuBg(menuBackground);

bool isMenu = 1;

int menuNum = 0;

menuTexture1.setSmooth(true);

menuTexture2.setSmooth(true);

menuBackground.setSmooth(true);

menuBg2.setSmooth(true);

menu1.setScale(1.3f,1.3f);

menuBg.setPosition(0, 0);

View view;

while (isMenu) // цикл отображения меню

{

menu2.setPosition(Vector2f(100, 60)); // устанавливаем позицию спрайтов на экране

menu1.setPosition(Vector2f(100, 160));

view.reset(FloatRect(0, 0, 1366, 768)); // камера для меню

window.setView(view); // установка вида для текущего меню

menuNum = 0;

window.clear();

// реализация анимации

if (IntRect(100, 60, 379, 55).contains(Mouse::getPosition(window)))

{

menuNum = 1;

menu2.setPosition(100 + 8, 60 + 8);

}

if (IntRect(100, 160, 165, 55).contains(Mouse::getPosition(window)))

{

menuNum = 2;

menu1.setPosition(100 + 8, 160 + 8);

}

if (Mouse::isButtonPressed(Mouse::Left)) // при нажатии на определенную иконку

{

if (menuNum == 1) { isMenu = false; Restart = true; break; // выход и перезапуск

for (int i = 0; i < 5; i++) { // сбрасываем звук для сбора звезд

soundStars[i] = true;

}

}

if (menuNum == 2) { isMenu = false; toMainMenuisone = 1; break; } // выход в главное меню

}

// отрисовка и отображение

window.draw(menuBg);

window.draw(menu1);

window.draw(menu2);

window.display();

}

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

}

// реализация интерфейса с функциями для установки уровня (изображения, спрайты, расположения статических тел Box2D)

// и отрисовка с отображением

class Levels

{

public:

Texture tlevel, bglvl, tStar;

Sprite slevel, sbglvl, sStar[5];

int numlvl = 0;

//

void setLevel1(b2World & World)

{

bglvl.loadFromFile("images/Levels/BGoflvl/sky1.jpg");

sbglvl.setTexture(bglvl, false); sbglvl.setPosition(0, -300); sbglvl.setScale(2.5f, 4.5f);

tlevel.loadFromFile("images/Levels/LEVEL1.png");

slevel.setTexture(tlevel, false);

// сглаживание

tlevel.setSmooth(true);

//////////// Stars ///////////////

tStar.loadFromFile("images/TileSet/hearts.png");

tStar.setSmooth(true);

numlvl = 1;

sStar[0].setTexture(tStar);

sStar[0].setScale(0.7f, 0.7f);

sStar[0].setTextureRect(IntRect(490, 10, 60, 80));

sStar[0].setPosition(450, 500);

sStar[1].setTexture(tStar);

sStar[1].setScale(0.7f, 0.7f);

sStar[1].setTextureRect(IntRect(490, 10, 60, 80));

sStar[1].setPosition(1025, 410);

sStar[2].setTexture(tStar);

sStar[2].setScale(0.7f, 0.7f);

sStar[2].setTextureRect(IntRect(490, 10, 60, 80));

sStar[2].setPosition(1030, 260);

sStar[3].setTexture(tStar);

sStar[3].setScale(0.7f, 0.7f);

sStar[3].setTextureRect(IntRect(490, 10, 60, 80));

sStar[3].setPosition(1750, 475);

sStar[4].setTexture(tStar);

sStar[4].setScale(0.7f, 0.7f);

sStar[4].setTextureRect(IntRect(490, 10, 60, 80));

sStar[4].setPosition(2700, 545);

/////////box2d///////////

setWall(3839 / 2, 800 + 11, 3839 / 2, 10, World); //

setWall(385, 0, 10, 2000, World); // left wall

setWall(3845, 0, 10, 2000, World); // right wall

//setWall(512 + (287 / 2)+1, 576 + 11 + (100), 287/2, 10 + (100), World); // red wall

setRTWall(512, 576, 287, 100, World); // !!!!!!!!!!!!!!!!!!!!!!

setWall(768 + (32 / 2) + 1, 544 + 11 + (100), 32 / 2, 10 + (100), World); // red box

setWall(864 + (128 / 2) + 1, 481 + 11 + (100 \* 2), 128 / 2, 10 + (100 \* 2), World); // first ground

setWall(1058 + (120 / 2) + 1, 384 + 11 + (100 \* 2), 120 / 2, 10 + (100 \* 2), World); // second ground

setWall(1281 + (31 / 2) + 1, 289 + 11 + (100 \* 3), 31 / 2, 10 + (100 \* 3), World); // 3 ground

setWall(1310 + ((62 / 2) + 1), 448 + (11 + (100 \* 3)), 62 / 2, 10 + (100 \* 3), World); // 3 ground hole

setWall(1346 + ((60 / 2) + 1), 289 + (11 + (100 \* 3)), 60 / 2, 10 + (100 \* 3), World); // 3 ground

setWall(1538 + ((92 / 2) + 1), 481 + (11 + (10)), 92 / 2, 10 + (10), World); // 4 platform

setWall(1553 + ((62 / 2) + 1), 495 + (11 + (100 \* 3)), 62 / 2, 10 + (100 \* 3), World); // 4 platform

setWall(1697 + ((68 / 2) + 1), 404 + (11 + (1)), 68 / 2, 10 + (1), World); // 5 platform

setWall(1760 + ((68 / 2) + 1), 384 - 32 + (11 + (1)), 68 / 2, 10 + (1), World); // 5 platform

setWall(1794 + ((124 / 2) + 1), 257 + (11 + (100 \* 3)), 124 / 2, 10 + (100 \* 3), World); // 6 platform

setWall(1920 + ((62 / 2) + 1), 416 + (11 + (100 \* 3)), 62 / 2, 10 + (100 \* 3), World); // mini-platform 1

setWall(1920 + ((62 / 2 \* 2) + 1), 416 + (33) + (11 + (100 \* 3)), 62 / 2 \* 2, 10 + (100 \* 3), World); // mini-platform 2

setWall(1920 + ((62 / 2 \* 2 \* 2) + 1), 416 + (33 \* 2) + (11 + (100 \* 3)), 62 / 2 \* 2, 10 + (100 \* 3), World); // mini-platform 3

setWall(1920 + ((62 / 2) - 23), 416 + (33 \* 2 \* 1.5) + (11 + (100 \* 3)), 62 / 2 \* 2 \* 2 \* 2, 10 + (100 \* 3), World); // mini-platform 4

setWall(2176 + ((63 / 2) + 1), 545 + (11 + (100 \* 3)), 63 / 2, 10 + (100 \* 3), World); // mini-platform 5

setWall(2240 + ((63 / 2) + 1), 576 + (11 + (100 \* 3)), 63 / 2, 10 + (100 \* 3), World); // mini-platform 6

setWall(2304 + ((63 / 2) + 1), 608 + (11 + (100 \* 3)), 63 / 2, 10 + (100 \* 3), World); // mini-platform 7

setWall(2368 + ((63 / 2) + 1), 641 + (11 + (100 \* 3)), 63 / 2, 10 + (100 \* 3), World); // mini-platform 8

setWall(2432 + ((63 / 2) + 1), 673 + (11 + (100 \* 3)), 63 / 2, 10 + (100 \* 3), World); // mini-platform 9

setWall(2496 + ((63 / 2) + 1), 705 + (11 + (100 \* 3)), 63 / 2, 10 + (100 \* 3), World); // mini-platform 10

setWall(2560 + ((63 / 2) + 1), 736 + (11 + (100 \* 3)), 63 / 2, 10 + (100 \* 3), World); // mini-platform 11

setWall(2688 + ((1000 / 2) + 1), 641 + (11 + (100 \* 3)), 1000 / 2, 10 + (100 \* 3), World); // platform 7

}

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

void setLevel2(b2World & World)

{

bglvl.loadFromFile("images/Levels/BGoflvl/sky.jpg");

sbglvl.setTexture(bglvl, false); sbglvl.setPosition(0, -300); sbglvl.setScale(2.5f, 2.5f);

tlevel.loadFromFile("images/Levels/REALLEVEL1.png");

slevel.setTexture(tlevel, false);

// сглаживание

tlevel.setSmooth(true);

//////////// Stars ///////////////

tStar.loadFromFile("images/TileSet/hearts.png");

tStar.setSmooth(true);

numlvl = 2;

sStar[0].setTexture(tStar);

sStar[0].setScale(0.7f, 0.7f);

sStar[0].setTextureRect(IntRect(490, 10, 60, 80));

sStar[0].setPosition(450, 500);

sStar[1].setTexture(tStar);

sStar[1].setScale(0.7f, 0.7f);

sStar[1].setTextureRect(IntRect(490, 10, 60, 80));

sStar[1].setPosition(945, 725);

sStar[2].setTexture(tStar);

sStar[2].setScale(0.7f, 0.7f);

sStar[2].setTextureRect(IntRect(490, 10, 60, 80));

sStar[2].setPosition(1655, 610);

sStar[3].setTexture(tStar);

sStar[3].setScale(0.7f, 0.7f);

sStar[3].setTextureRect(IntRect(490, 10, 60, 80));

sStar[3].setPosition(2575, 585);

sStar[4].setTexture(tStar);

sStar[4].setScale(0.7f, 0.7f);

sStar[4].setTextureRect(IntRect(490, 10, 60, 80));

sStar[4].setPosition(3085, 675);

/////////box2d///////////

setRTWall(0, 801, 4640, 1000, World); // ground

setRTWall(0, 0, 364, 1000, World); // leftwall

setRTWall(3840, 0, 364, 1000, World); // rightwall

setRTWall(512, 576, (800 - 512), 1000, World); // platform 1

setRTWall(864, 480, (896 - 864 - 3), 1000, World); // platform 2

setRTWall(928, 768, (950 - 928 + 10), 1000, World); // platform 3

setRTWall(1056, 672, (1184 - 1056), 1000, World); // platform 4

setRTWall(1280, 640, (1408 - 1280 - 1), 1000, World); // platform 5

setRTWall(1408, 640, (1408 - 1408 - 1), 1000, World); // platform 6

setRTWall(1280, 673, (1568 - 1280), 1000, World); // platform 7

setRTWall(1664, 640, (1856 - 1664 - 1), 30, World); // platform 8

setRTWall(1920, 576, (2240 - 1920 - 3), 30, World); // platform 9

setRTWall(2272, 609, 32, 1000, World); // platform 10

setRTWall(2272 + 64 + 32, 609, 32, 1000, World); // platform 11

setRTWall(2272 + 64 \* 3, 609, 32, 1000, World); // platform 12

setRTWall(2272 + 64 \* 4 + 32, 609, 32, 1000, World); // platform 13

setRTWall(2272 + 64 \* 7 - 64, 609, 32, 1000, World); // platform 14

setRTWall(2689, 577, 32 \* 3, 32, World); // platform 15

setRTWall(2816, 705, 32, 32, World); // platform 16

setRTWall(2816 + 32 \* 4, 705, 32, 32, World); // platform 17

setRTWall(2816 + 32 \* 8, 705, 32, 32, World); // platform 18

setRTWall(2816 + 32 \* 12, 705, 32, 32, World); // platform 19

setRTWall(2816 + 32 \* 16, 705, 32, 32, World); // platform 20

setRTWall(3425, 705, 320, 932, World); // platform 19

}

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

void setLevel3(b2World & World)

{

bglvl.loadFromFile("images/Levels/BGoflvl/sky1.jpg");

sbglvl.setTexture(bglvl, false); sbglvl.setPosition(0, -300); sbglvl.setScale(2.5f, 2.5f);

tlevel.loadFromFile("images/Levels/LEVEL3.png");

slevel.setTexture(tlevel, false);

// сглаживание

tlevel.setSmooth(true);

//////////// Stars ///////////////

tStar.loadFromFile("images/TileSet/hearts.png");

tStar.setSmooth(true);

numlvl = 3;

sStar[0].setTexture(tStar);

sStar[0].setScale(0.7f, 0.7f);

sStar[0].setTextureRect(IntRect(490, 10, 60, 80));

sStar[0].setPosition(615, 640);

sStar[1].setTexture(tStar);

sStar[1].setScale(0.7f, 0.7f);

sStar[1].setTextureRect(IntRect(490, 10, 60, 80));

sStar[1].setPosition(525, 540);

sStar[2].setTexture(tStar);

sStar[2].setScale(0.7f, 0.7f);

sStar[2].setTextureRect(IntRect(490, 10, 60, 80));

sStar[2].setPosition(1200, 480);

sStar[3].setTexture(tStar);

sStar[3].setScale(0.7f, 0.7f);

sStar[3].setTextureRect(IntRect(490, 10, 60, 80));

sStar[3].setPosition(1750, 475);

sStar[4].setTexture(tStar);

sStar[4].setScale(0.7f, 0.7f);

sStar[4].setTextureRect(IntRect(490, 10, 60, 80));

sStar[4].setPosition(1716, 485);

/////////box2d///////////

setRTWall(0, 801, 4640, 1000, World); // ground

setRTWall(0, 0, 383, 1000, World); // leftwall

setRTWall(3840, 0, 364, 1000, World); // rightwall

setRTWall(513, 578, 31, 1000, World); // platform 1

setRTWall(608, 673, 31, 7, World); // platform 2

setRTWall(640, 578, 31, 1000, World); // platform 3

setRTWall(640+32\*2+2, 578, 29, 1000, World);

setRTWall(640 + 32 \* 4+2, 578, 29, 1000, World);

setRTWall(640 + 32 \* (4+7) + 2, 578, 32\*12-5, 1000, World);

setRTWall(1600, 578, 31\*2, 1000, World);

setRTWall(1728, 578, 31\*2, 1000, World);

setRTWall(1857, 578, 31 \* 2, 1000, World);

setRTWall(1985, 689+16, 31\*3, 5, World);

setRTWall(1985+32\*5-1, 689 + 16, 31 \* 4, 5, World);

setRTWall(1985 + 32 \* 18 - 1, 689 + 16, 31 \* 4, 5, World);

setRTWall(2048, 518, 31, 5, World);

setRTWall(2240, 577, 31, 5\*5\*2, World);

setRTWall(2369, 577, 31, 5 \* 5 \* 2, World);

setRTWall(2369, 706, 31\*4, 7, World);

setRTWall(2753, 385, 31, 5 \* 9 \* 2, World);

setRTWall(2720, 449, 33, 6, World);

setRTWall(2689, 577, 31\*4, 6, World);

setRTWall(3040, 705, 31 \* 44, 6, World);

}

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

void setLevel4(b2World & World)

{

bglvl.loadFromFile("images/Levels/BGoflvl/sky1.jpg");

sbglvl.setTexture(bglvl, false); sbglvl.setPosition(0, -300); sbglvl.setScale(2.5f, 2.5f);

tlevel.loadFromFile("images/Levels/LEVEL4.png");

slevel.setTexture(tlevel, false);

// сглаживание

tlevel.setSmooth(true);

//////////// Stars ///////////////

tStar.loadFromFile("images/TileSet/hearts.png");

tStar.setSmooth(true);

numlvl = 4;

sStar[0].setTexture(tStar);

sStar[0].setScale(0.7f, 0.7f);

sStar[0].setTextureRect(IntRect(490, 10, 60, 80));

sStar[0].setPosition(1160, 700);

sStar[1].setTexture(tStar);

sStar[1].setScale(0.7f, 0.7f);

sStar[1].setTextureRect(IntRect(490, 10, 60, 80));

sStar[1].setPosition(1425, 660);

sStar[2].setTexture(tStar);

sStar[2].setScale(0.7f, 0.7f);

sStar[2].setTextureRect(IntRect(490, 10, 60, 80));

sStar[2].setPosition(1880, 310);

sStar[3].setTexture(tStar);

sStar[3].setScale(0.7f, 0.7f);

sStar[3].setTextureRect(IntRect(490, 10, 60, 80));

sStar[3].setPosition(2315, 680);

sStar[4].setTexture(tStar);

sStar[4].setScale(0.7f, 0.7f);

sStar[4].setTextureRect(IntRect(490, 10, 60, 80));

sStar[4].setPosition(3243, 670);

/////////box2d///////////

setRTWall(0, 801, 4640, 1000, World); // ground

setRTWall(0, 0, 383, 1000, World); // leftwall

setRTWall(3840, 0, 364, 1000, World); // rightwall

setRTWall(512, 577, (800 - 512), 2, World); // platform 1

setRTWall(992, 641, (1184-992), 2, World); // platform 2

setRTWall(1152, 737, (1216 - 1152), 2, World); // platform 3

setRTWall(1344, 737, (1504 - 1344), 2, World); // platform 4

setRTWall(1568, 641, (1600-1568), 2, World); // platform 5

setRTWall(1728, 545, 32, 2, World); // platform 6

setRTWall(1600, 417, 32, 2, World); // platform 7

setRTWall(1728, 353, 32, 2, World); // platform 8

setRTWall(1856, 353, 32\*3, 2, World); // platform 9

setRTWall(2048, 577, 32 \* 2, 2, World); // platform 10

setRTWall(2208, 737, 32 \* 5, 2, World); // platform 11

setRTWall(2528, 705, 32 \* 3, 2, World); // platform 12

setRTWall(2816, 705, 32 \* 30, 2, World); // platform 13

}

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

void setLevel5(b2World & World)

{

bglvl.loadFromFile("images/Levels/BGoflvl/sky1.jpg");

sbglvl.setTexture(bglvl, false); sbglvl.setPosition(0, -300); sbglvl.setScale(2.5f, 2.5f);

numlvl = 0;

tlevel.loadFromFile("images/Levels/LEVEL5.png");

slevel.setTexture(tlevel, false);

// сглаживание

tlevel.setSmooth(true);

//////////// Stars ///////////////

// ЧЕМ БОЛЬШЕ СОБРАНО ЗВЁЗД НА ПРЕДЫДУЩИХ УРОВНЯХ, ТЕМ БОЛЬШЕ БОНУСОВ НА ПОСЛЕДНЕМ

// В УРОВНЯХ ТАК ЖЕ ИМЕЮТСЯ БОНУСЫ

/////////box2d///////////

setRTWall(0, 801, 4640, 1000, World); // ground

setRTWall(0, 0, 383, 1000, World); // leftwall

setRTWall(3840, 0, 364, 1000, World); // rightwall

}

void Drawlvl(b2World & World)

{

//////////Draw///////////////

window.draw(sbglvl);

window.draw(slevel);

switch (numlvl)

{

case 1: for (int i = 0; i < 5; i++) { if (Stars[i] == true) window.draw(sStar[i]); }break;

case 2: for (int i = 0; i < 5; i++) { if (Stars[i] == true) window.draw(sStar[i]); } break;

case 3: for (int i = 0; i < 5; i++) { if (Stars[i] == true) window.draw(sStar[i]); } break;

case 4: for (int i = 0; i < 5; i++) { if (Stars[i] == true) window.draw(sStar[i]); } break;

case 5: for (int i = 0; i < 5; i++) { if (Stars[i] == true) window.draw(sStar[i]); } break;

}

}

};

Приложение Б

(справочное)

Экранные формы

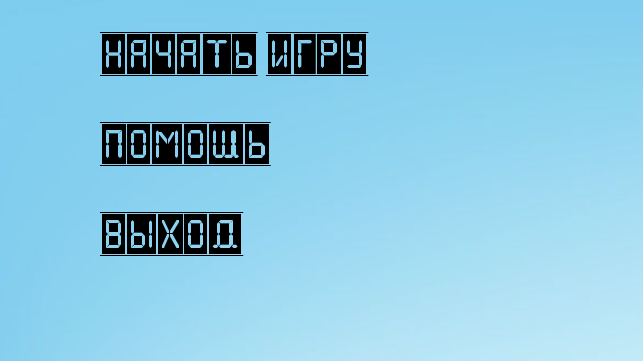


Рисунок Б.1

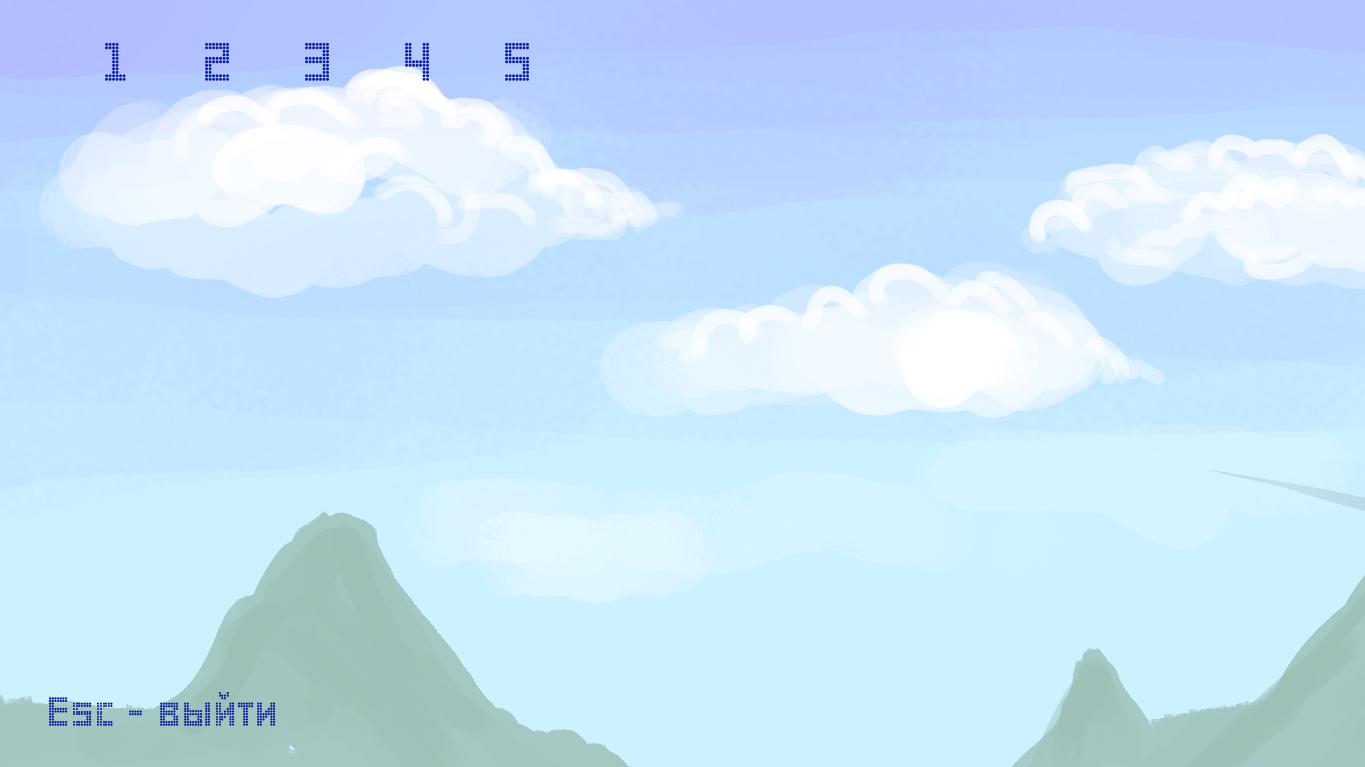


Рисунок Б.2

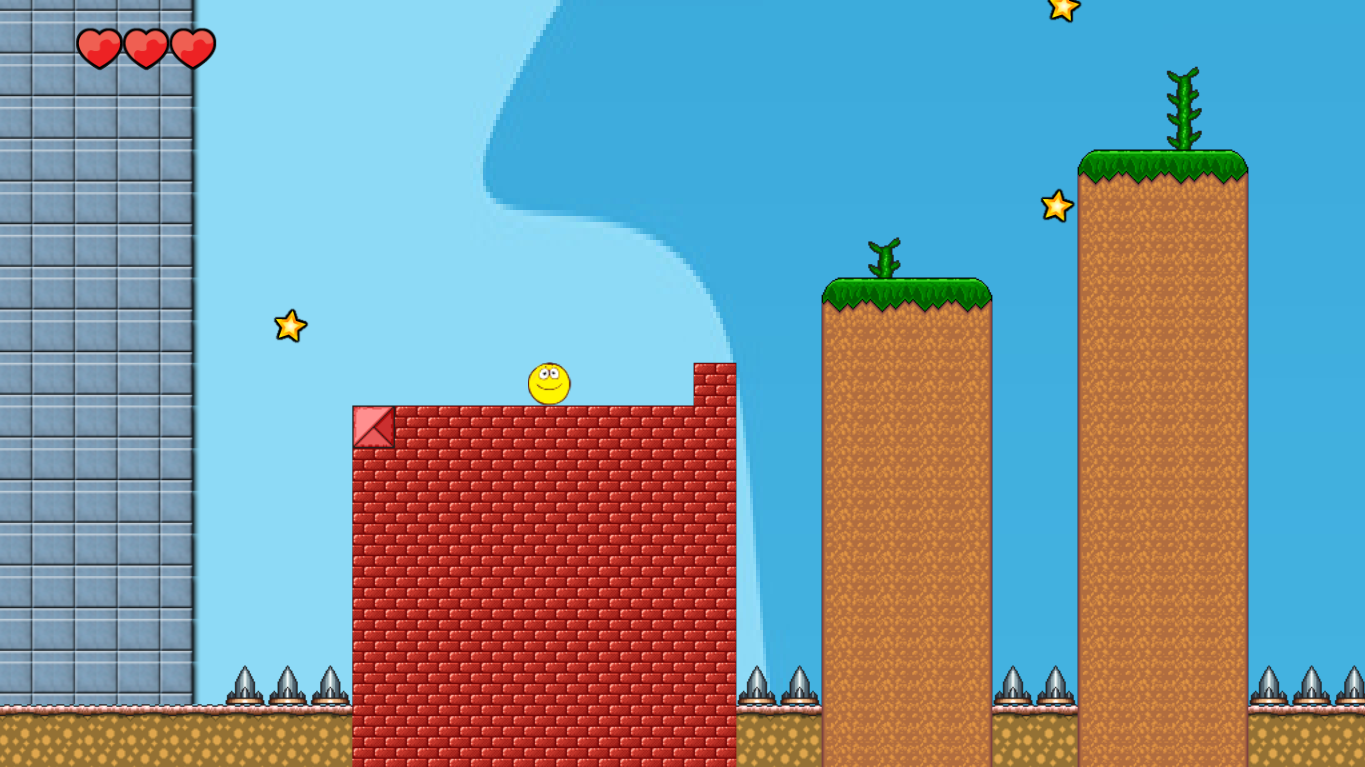


Рисунок Б.3

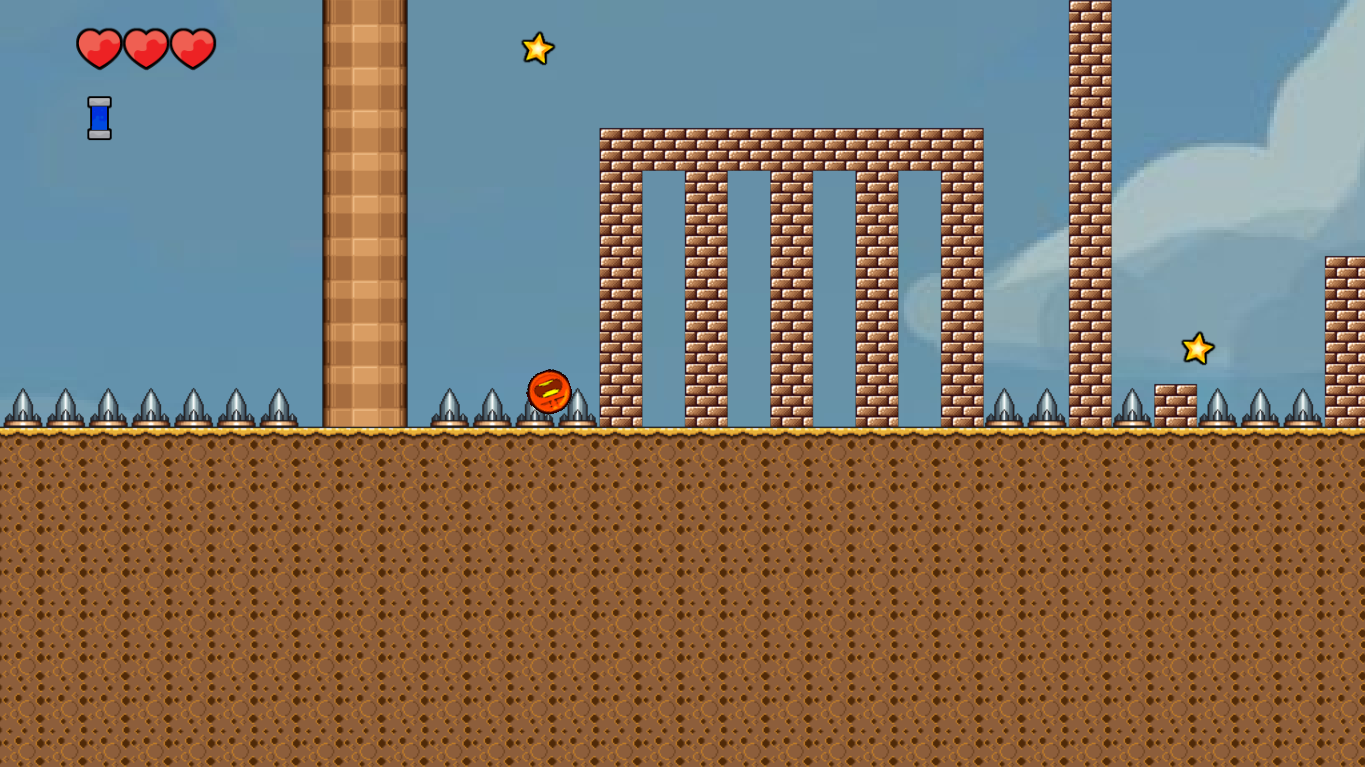


Рисунок Б.4



Рисунок Б.5

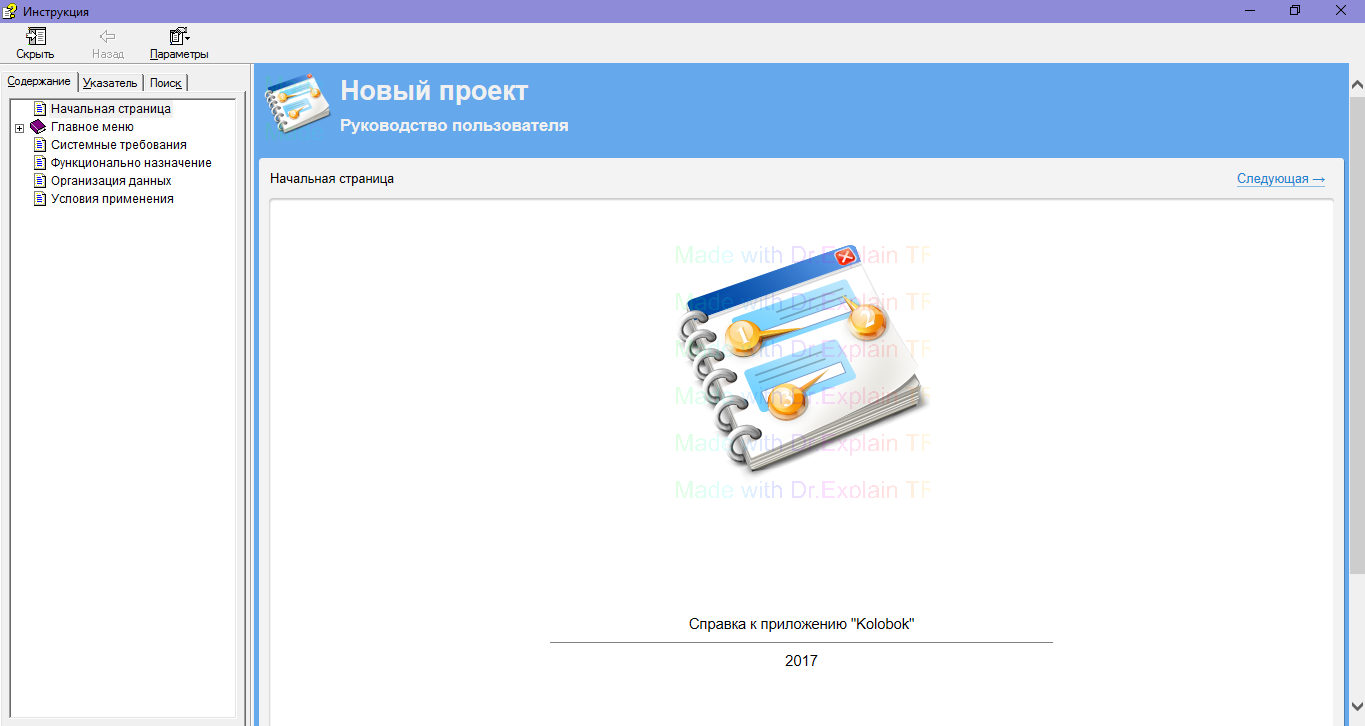


Рисунок Б.6