МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И.ВЕРНАДСКОГО»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

**КОМПЬЮТЕРНАЯ ИГРА RUN DF**

**С БИБЛИОТЕКОЙ SFML**

Курсовая работа

по дисциплине «Программирование»

студента 1 курса группы ИВТ-б-о-202

Маслова Дмитрия Дмитриевича

направления подготовки 09.03.01 «Информатика и вычислительная техника»

|  |  |  |
| --- | --- | --- |
| Научный руководитель  Старший преподаватель кафедры компьютерной инженерии и моделирования | \_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Чабанов В.В. |

Симферополь, 2021

**РЕФЕРАТ**

Создание игры RUN DF c библиотекой SFML. – Симферополь: ФТИ КФУ им. Вернадского, 2021.

Курсовой проект выполнен на 27 листах, состоит из 3-х глав, в проекте 5 иллюстрации, 1 приложение и 8 литературных источников.

**Жанр создания игры:** платфомер.

**Предметы в игре:** персонаж, беговые дороги, препятствия (стенки).

**Целью** курсовой работы является создание игры RUN DF. Смысл игры в том, что нужно бегать по трем дорожкам, избегая препятствия (стенки). Для создания игры был использован один из основных языков программирования C++ и библиотеки SFML. C++ - компилируемый язык программирования общего назначения, сочетает свойства как высокоуровневых, так и низкоуровневых языков программирования.

В связи с целью курсовой работы, были поставлены следующие **задачи:**

* Изучена тема о аналогах подобной игры
* Изучены теоретические аспекты одного из основных языков программирования C++
* Изучено создание библиотеки SFML

Создать игру RUN DF с библиотекой SFML на языке программирования C++.

В результате создана игра RUN DF с библиотекой SFML. Для создания игры был использован один из основных языков программирования C++ и библиотеки SFML. C++ - компилируемый язык программирования общего назначения, сочетает свойства как высокоуровневых, так и низкоуровневых языков программирования.

Выполнены следующие **задачи:**

* Изучена тема о аналогах подобной игры
* Изучены теоретические аспекты одного из основных языков программирования C++
* Изучено создание библиотеки SFML

Достоинства игры RUN DF:

* Тренирует скорость реакции
* Помогает коротать время
* Отсутствует реклама
* Несложное управление
* Игра проста и понятна

Если добавить в данную игру уровни и миссии, она будет похожа на игру Subway Surf.

Для перспективы монетизации бесплатной игры необходимо встроить платные дополнения. В качестве дополнения может выступать такие функции, как смена карты, смена игрока и т.д.

Игра может быть использована, как и детям, так и взрослым, чтобы к примеру, скоротать время.

**Оглавление**

[**ВВЕДЕНИЕ** 5](#_Toc72811415)

[**ГЛАВА 1** 6](#_Toc72811416)

[**ПОСТАНОВКА ЗАДАЧИ** 6](#_Toc72811417)

[**1.1** **Цель проекта** 6](#_Toc72811418)

[**1.1.1** **История создания компьютерных игр** 6](#_Toc72811419)

[**1.1.2** **Что же такое компьютерная игра** 7](#_Toc72811420)

[**1.2** **Существующие аналоги** 8](#_Toc72811421)

[**1.3** **Отличия от аналогов игры** 9](#_Toc72811422)

[**1.4** **Техническое задание** 10](#_Toc72811423)

[**ГЛАВА 2** 11](#_Toc72811424)

[**ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИГРЫ** 11](#_Toc72811425)

[**2.1 Анализ инструментальных средств** 11](#_Toc72811426)

[**2.2 Установка SFML** 11](#_Toc72811427)

[**2.3 Основные пространства имен WPF** 13](#_Toc72811428)

[**ГЛАВА 3** 15](#_Toc72811429)

[**ТЕСТИРОВАНИЕ ИГРЫ** 15](#_Toc72811430)

[**3.1 Тестирование исходного кода** 15](#_Toc72811431)

[**3.2 Тестирование интерфейса игры** 15](#_Toc72811432)

[**ГЛАВА 4** 17](#_Toc72811433)

[**ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА** 17](#_Toc72811434)

[**4.1 Перспективы технического развития** 17](#_Toc72811435)

[**4.2 Перспективы монетизации** 17](#_Toc72811436)

[**ЗАКЛЮЧЕНИЕ** 18](#_Toc72811437)

[**ЛИТЕРАТУРА** 19](#_Toc72811438)

[**ПРИЛОЖЕНИЕ 1** 20](#_Toc72811439)

[**КОД ОСНОВНЫХ МОДУЛЕЙ ПРОЕКТА** 20](#_Toc72811440)

# **ВВЕДЕНИЕ**

Программирование - одновременно наука и искусство создания компьютерных программ и программного обеспечения с помощью языков программирования. Язык программирования - формальная знаковая система, предназначенная для записи компьютерных программ. Программирование сочетает в себе элементы искусства, фундаментальных наук (прежде всего информатика и математика), связана со всеми сферами науки и жизни человека, в том числе с культурой правильного питания.

**Жанр создания игры:** платфомер.

**Предметы в игре:** персонаж, беговые дороги, препятствия (стенки).

**Целью** курсовой работы является создание игры RUN DF. Смысл игры в том, что нужно бегать по трем дорожкам, избегая препятствия (стенки). Для создания игры был использован один из основных языков программирования C++ и библиотеки SFML. C++ - компилируемый язык программирования общего назначения, сочетает свойства как высокоуровневых, так и низкоуровневых языков программирования.

В связи с целью курсовой работы, были поставлены следующие **задачи:**

* Изучена тема о аналогах подобной игры
* Изучены теоретические аспекты одного из основных языков программирования C++
* Изучено создание библиотеки SFML
* Создать игру RUN DF с библиотекой SFML на языке программирование C++

# **ГЛАВА 1**

# **ПОСТАНОВКА ЗАДАЧИ**

* 1. **Цель проекта**

При создании игры RUN DF c библиотекой SFML первоначально необходимо изучить выбранную программу (Visual Studio) и создать в ней игру. При создании игры узнать историю возникновения компьютерных игр, рассмотреть технологию создание компьютерной игры. С помощью компьютерных игр можно развиваться, к примеру, тренировать скорость реакции.

В связи с этим, была поставлена цель: создать игру RUN DF с библиотекой SFML. Игра позволит тренировать скорость реакции и скоротать время.

Из цели вытекают следующие задачи, первая из которых – изучить тему о аналогах подобной игры, а изучить технологию создание компьютерной игры. После изучения темы компьютерных игр, необходимо обратиться к теоретическим аспектам одного из основных языков программирования C++, а также расширению SFML.

* + 1. **История создания компьютерных игр**

На теоретическом уровне история развития компьютерных игр началась в далеком 1951 году, когда инженер Ральф Баэр впервые предложил идею интерактивного телевидения. В следующем, 1952 году, А.Дуглас создает "ЭХО" - компьютерный вариант крестиков-ноликов. История компьютерных игр развивалась достаточно медленно. Итак, время – 1961 год. Место – Массачусетский Технологический университет. Событие – на одном из мэнфреймов впервые появилась на свет компьютерная игра Space War. Игровая составляющая была весьма проста: два маленьких кораблика перемещались по экрану и обстреливали друг друга. Примерно в это же время произошел рывок в области графики – текстовые терминалы стали постепенно заменяться векторными графическими дисплеями. Соответственно, электронный луч обрисовывал на экране заданные программой контуры объектов.

Процесс развития компьютерных игр уже нельзя остановить. Они становятся все более разнообразными и захватывающими. На смену персональным играм пришли игры браузерные, где в онлайн-режиме можно общаться, сообща решать пусть виртуальные, но важные задачи. Такие игры, помимо развлечения, дают возможность осваивать и развивать навыки общения, социализации, расширять кругозор. Мир компьютерных игр по-прежнему дает больше положительных эффектов, нежели отрицательных. Важно лишь правильно выбирать для себя подходящие игры и верно дозировать количество времени, проведенного у монитора.

В наши дни огромное количество самых разных по интересам людей частенько играют в компьютерные игры. Компьютерные игры стали настоящим культурным феноменом - возникнув как незамысловатый плод творческой мысли программистов, они с каждым годом приобретали всё большую популярность - и развились до того, что стали отдельной специфической спортивной дисциплиной – киберспортом. В настоящее время существует много жанров и поджанров игр, такие как экшн, приключения, симуляторы, стратегии, спортивные и т.д.

### **Что же такое компьютерная игра**

Компьютерная игра – это [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), которая служит для организации [игрового](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%B0) процесса ([геймплея](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%B9%D0%BC%D0%BF%D0%BB%D0%B5%D0%B9)), связи с партнёрами по игре, или сама выступает в качестве [партнёра](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B0%D1%80%D1%80%D0%B8%D0%BD%D0%B3).

В настоящее время в ряде случаев вместо "компьютерная игра" может использоваться "[видеоигра](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%B8%D0%B3%D1%80%D0%B0)", то есть данные термины могут употребляться как синонимы и быть взаимозаменяемыми. В компьютерных играх, как правило, игровая ситуация воспроизводится на экране дисплея или обычного телевизора (в этом случае компьютерные игры одновременно являются и [видеоиграми](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%B8%D0%B3%D1%80%D0%B0)), но в то же время компьютерная игра может быть [звуковой](https://ru.wikipedia.org/wiki/%D0%97%D0%B2%D1%83%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0).

Видеоигры могут создаваться на основе [фильмов](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BD%D0%BE%D1%84%D0%B8%D0%BB%D1%8C%D0%BC) [книг](https://ru.wikipedia.org/wiki/%D0%A5%D1%83%D0%B4%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5); есть и обратные случаи. С 2011 года компьютерные игры официально признаны в [США](https://ru.wikipedia.org/wiki/%D0%A1%D0%A8%D0%90) отдельным [видом искусства](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%BE).

Компьютерные игры оказали столь существенное влияние на общество, что в [информационных технологиях](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8) отмечена устойчивая тенденция к [геймификации](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%B9%D0%BC%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) для неигрового [прикладного программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%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).

Игровой процесс или геймплей ([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) gameplay) — компонент игры, отвечающий за интерактивное взаимодействие игры и игрока. Геймплей описывает, как игрок взаимодействует с игровым миром, как игровой мир реагирует на действия игрока и как определяется набор действий, который предлагает игроку игра.

* 1. **Существующие аналоги**

**Subway Surfers** - традиционный трехмерный раннер. Пользователь играет за уличного художника, убегающего от железнодорожного инспектора. Игрок должен уворачиваться от препятствий, несущихся на него поездов и собирать монетки.

**Temple Run** – это одна из оригинальных игр в формате бесконечных раннеров, в которых используются прыжки, скольжения, удары и механика наклона. В которой игроки должны будут убежать от демонических обезьян с украденной редкой реликвией.

**Crossy Road**— это мобильная пиксельная игра, в которой игроки пересекают оживленную улицу своим персонажем. Сочетание классического геймплея и добавление в него бесконечного компонента, который доказал свою успешную формулу на платформах iOS и Android делает игру ещё интереснее.

**Into The Dead** — это упрощенный бесконечный раннер, в котором вы уклоняетесь от смертоносной хватки зомби, путешествуя через преимущественно открытые поля. Решив отойти от механики обычно ярких и бесконечных игр в раннер тематике представленных на рынке, Into The Dead предлагает нечто более интенсивное, что-то, что может заставить ваше сердце биться быстрее.

**Jetpack Joyride** предлагает вам сбежать из сверхсекретной лаборатории с украденной технологией при помощи вашего реактивного ранца. Несмотря на название игры, это для игроков оказывается не самой простой задачей. Им придётся уворачиваться от электрических ворот, ракет и других разнообразных препятствий в этом приключении в формате боковой прокрутки и вдохновленной тематикой про секретных агентов.

Игра **Minion Rush** сочетает в себе геймплей бесконечного раннера с миньонами из знаменитого мультифильма “Гадкий Я”, чтобы предложить хороший мобильный опыт с более простым игровым процессом, чем в большинстве других альтернатив в этом жанре. Оставаясь верным мультику с точки зрения общей атмосферы, именно это особенно понравится молодым поклонникам этой вселенной.

**Agent, Run!**  Это 2D бесконечный раннер, который представляет интересную механику и предлагает бесплатное приложение для Android. Игра идеально подходит для поклонников этого увлекательного жанра, которые хотят более сложного и разнообразного игрового процесса, который даст настоящее ощущение прогресса. Agent, Run! заставляет игроков проходить до 4-х уровней в качестве секретного агента, пытающегося спасти мир.

**Sonic Dash** позволяет вам играть за своего любимого ежа в бесконечном раннере, который очень напоминает игру Sonic с битвами с боссами, сумасшедшим дизайном треков и всеми вашими любимыми персонажами из франшизы на вашем мобильном устройстве.Sonic Dash сводит на нет все основные элементы жанра и обладает отзывчивым управлением, приятной графикой и большим разнообразием маршрутов.

* 1. **Отличия от аналогов игры**

Основным отличием от аналогов является: игры абсолютно без рекламы, но у конкурентов есть миссии и различные задания, присвоение монет (денег) на покупку скина или чего-нибудь другого, есть и меню.

* 1. **Техническое задание**

Создать игру RUN DF с библиотекой SFML. Игра должна работать без любой возможной рекламы. Необходимо через расширение SFML сделать платформу где можно будет играть. Написать код для игры и присвоить картинки для анимации.

# **ГЛАВА 2**

# **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИГРЫ**

## **2.1 Анализ инструментальных средств**

Игра RUN DF с разрешением SFML разработана с помощью языка C++.

«С++ - язык программирования, который поддерживает такие [парадигмы программирования](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D1%8B_%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) как процедурное программирование, объектно-ориентированное программирование, обобщенное программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как [высокоуровневых](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%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), так и [низкоуровневых языков](http://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D0%B7%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%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).» [1, с.74].

Работа игры была разработана с помощью программы Visual Studio и библиотеки SFML.

Консольное приложение — это программа, которая выполняется из командной строки окна DOS или Windows и не имеет графического интерфейса. Проект консольного приложения создается пустым и предполагает добавление в него исходных файлов вручную. [2, с 13.]

Библиотека SFML предоставляет простой интерфейс для различных компонентов компьютера, чтобы облегчить разработку игр и мультимедийных приложений. Она состоит из пяти модулей: system, window, graphics, audio и network.

## **2.2 Установка SFML**

Для начала я скачал SFML SDK, самое главное, чтобы скаченный пакет соответствовал версии Visual C++.

Далее я распаковал архив с SFML в директорию. Не рекомендуется копировать заголовочные файлы и библиотеки в установку Visual Studio. Лучше держать библиотеки в отдельном месте.

Первое, я выбрал тип создаваемого проекта «Win32 application». Мастер мне предложил несколько опций для настройки проекта, я выбрал «Console application», следом выбрал «Empty project», чтобы был автоматически сгенерированный код.

Создал файл main.cpp и добавил его в проект. Далее указал компилятору, где искать заголовочные файлы (файлы с расширением .hpp) и компоновщику, где искать библиотеки SFML (файлы с расширением .lib).

Добавил в свойства проекта следующее:

* Путь до заголовочных файлов SFML в C/C++ » General » Additional Include Directories
* Путь до библиотек SFML в Linker » General » Additional Library Directories

Эти пути совпадают для конфигураций Debug и Release. Следующий мой шаг — компоновка игры с библиотеками SFML (файлы с расширением .lib). Как было упомянуто ранее, SFML состоит из пяти модулей (system, window, graphics, network и audio). Библиотеки добавил в свойства проекта в Linker » Input » Additional Dependencies. Указал библиотеки, соответствующие конфигурации: «sfml-xxx-d.lib» для Debug и «sfml-xxx.lib» для Release, чтобы не возникли ошибки.

Также я определил макрос SFML\_STATIC в опциях препроцессора моего проекта.

При статической компоновке я также скомпоновал все зависимости SFML.

## **2.3 Microsoft 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 инструментов. Данные продукты позволяют разрабатывать как [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%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), так и игры и приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в том числе с поддержкой технологии [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), а также [веб-сайты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82), [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), [веб-службы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) как в [родном](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), так и в [управляемом](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) кодах для всех платформ, поддерживаемых [Windows](https://ru.wikipedia.org/wiki/Windows), [Windows Mobile](https://ru.wikipedia.org/wiki/Windows_Mobile), [Windows CE](https://ru.wikipedia.org/wiki/Windows_CE), [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), [Xbox](https://ru.wikipedia.org/wiki/Xbox), [Windows Phone](https://ru.wikipedia.org/wiki/Windows_Phone) [.NET Compact Framework](https://ru.wikipedia.org/wiki/.NET_Compact_Framework) и [Silverlight](https://ru.wikipedia.org/wiki/Silverlight).

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)).

Visual Studio имеет множество функции:

* Модульная установка
* Создание облачных приложений для Azure
* Создание приложений для интернета
* Создание кроссплатформенных приложений и игр
* Подключение к базам данных
* Откладка, тестирование и совершенствование кода
* Развертывание готового приложения
* Управление исходным кодом и совместная работа

**2.4 Основные пространства имен WPF**

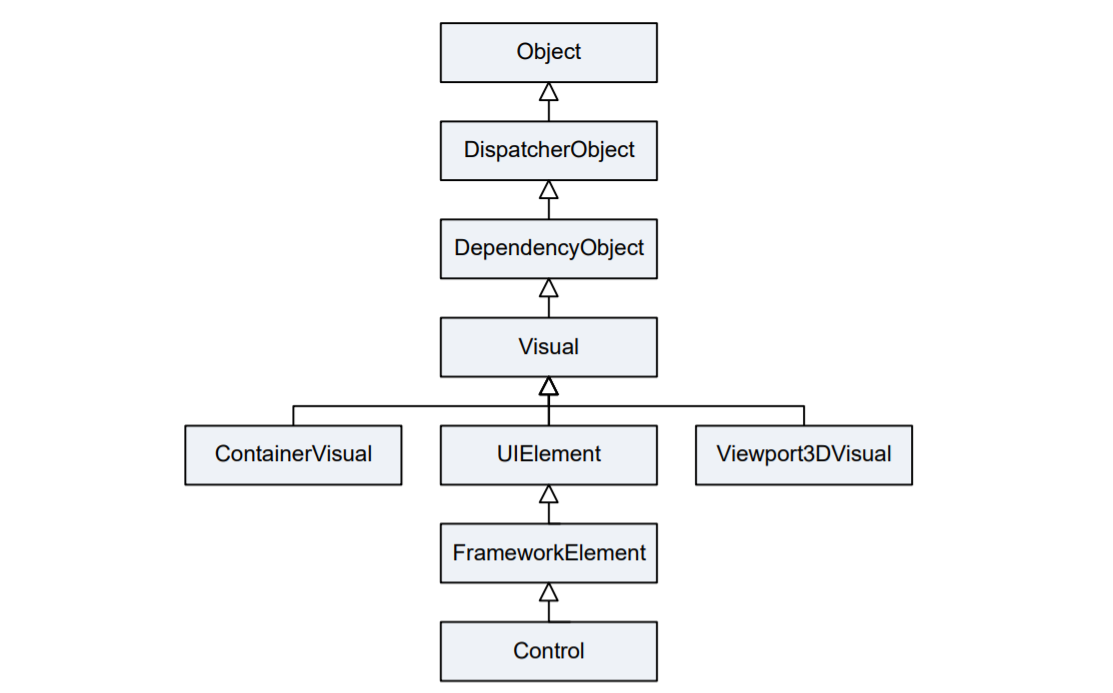


Рисунок 1 (классы WPF)

Как видно из рис. 1, классы WPF порождаются из класса *DispatcherObject*. Класс *DispatcherObject* предназначен для обеспечения доступа к объекту *Dispatcher*, который предоставляет службы для управления очередью рабочих элементов для потока. Дело в том, что приложения WPF используют однопоточную модель (Single-Thread Affinity, STA), а это означает, что весь пользовательский интерфейс принадлежит только одному потоку. Взаимодействовать с элементами пользовательского интерфейса из других потоков небезопасно. Чтобы содействовать работе этой модели, каждое приложение WPF управляется диспетчером, координирующим сообщения от клавиатуры, перемещений курсора мыши и др. Поэтому при наследовании от класса *DispatcherObject*, каждый элемент пользовательского интерфейса WPF может обратиться к диспетчеру, чтобы направить код в поток пользовательского интерфейса. [3]

Класс *DependencyObject* включает службы системы свойств Windows Presentation Foundation (WPF) для множества своих производных классов. Основной функцией системы свойств является вычисление значений свойств и системное уведомление о значениях, которые изменились. Наследуясь от *DependencyObject*, классы WPF получают поддержку системы свойств WPF. [3]

Класс *Visual* как единственный объект рисования инкапсулирует в себе инструкции рисования, дополнительные подробности рисования (наподобие отсечения, прозрачности и настроек трансформации) и базовую функциональность (такую как проверка попадания). Класс *Visual* также обеспечивает связь между управляемыми библиотеками WPF и библиотекой визуализации milcore.dll. Любой класс, унаследованный от *Visual,* обладает способностью отображаться в окне. [3]

Класс *UIElement* добавляет поддержку компоновки, ввода, фокуса и событий. В этом классе обработка нажатий кнопок мыши и клавиатуры трансформируется в более удобные события, такие как *MouseEnter*. Класс *UIElement* также добавляет поддержку привязки данных, анимации и стилей. [3]

Класс *Fr*ameworkElement — конечный пункт в центральном дереве наследования WPF. Он реализует некоторые члены, которые просто определены в классе *UIElement*. [3]

Класс *Control* добавляет дополнительные свойства для установки шрифта, а также цветов переднего плана и фона. Но наиболее интересная деталь, которую он предоставляет, — это поддержка шаблонов, которая позволяет заменять стандартный внешний вид элемента управления вашим собственным рисованием. [3]

# **ГЛАВА 3**

# **ТЕСТИРОВАНИЕ ИГРЫ**

## **3.1 Тестирование исходного кода**

**Тест 1. Движение**

Проверял работу движения, стрелкой вниз и вверх, двигается вниз и вверх, соответственно, чтобы обходил препятствия. (см. рис. 2) Тест успешно пройден.

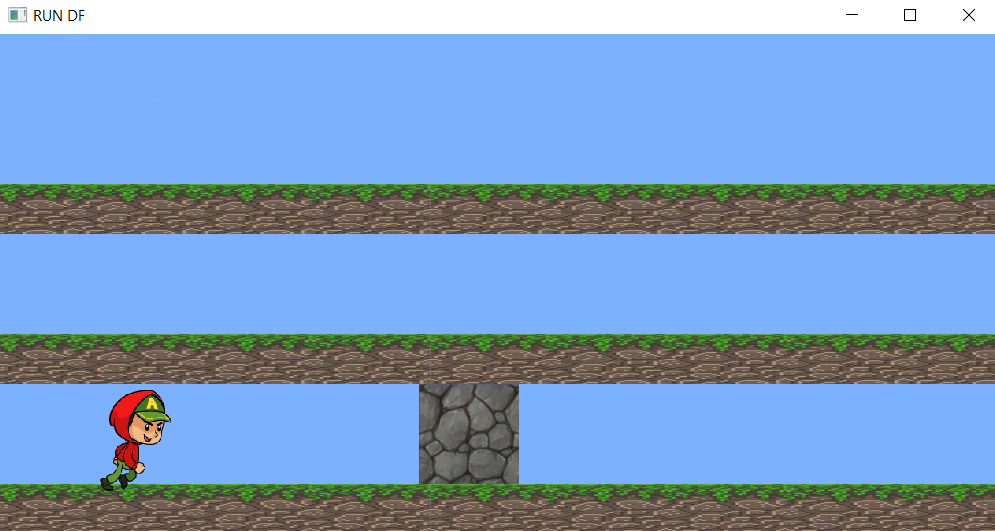


Рисунок 2 (тест движения)

**Тест 2. Счет очков при движении**

Далее добавил счет очков при движении. Чем больше пробегаете расстояния, тем больше получите очков. Данная функция работает исправно. (см. рис. 3) Тест успешно пройден.



Рисунок 3 (тест на счет очков при движении)

**Тест 3. Столкновение с препятствиями**

При столкновении с препятствием персонаж будет подпрыгивать и морщится, и будет написано Game over. (см. рис. 4) Таким образом игра заканчивается. Тест успешно пройден.



Рисунок 4 (тест столкновение с препятствиями)

**Тест 4**

Файл Source.cpp не выдал ошибок, RUN DF заработал (Рис. 2)

Локальный отладчик Windows выдал: D:\KFU\Programming\CourseWork\Code\main\Debug\main.exe (процесс 1772) завершил работу с кодом 0.

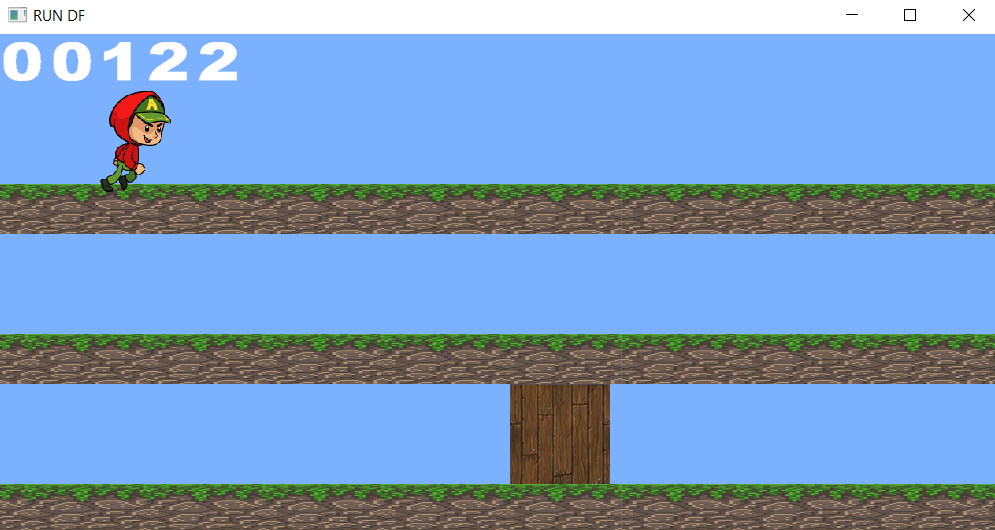


Рисунок 5 (игра RUN DF работает)

**Вывод**

Игра выполняет поставленные задачи. Основное внимание необходимо уделить структурному оформлению кода, стилю наименования переменных и массивов для повышения читаемости кода.

## **3.2 Тестирование интерфейса игры**

При запуске игры, требуется нажать курсором мыши на любое место в игре. Игра запускается и стрелочками вниз и вверх на клавиатуре управляем персонажем DF. Если не сможете обойти препятствия, то будет написано “GAME OVER” и все начинается сначала.

Все кнопки работают правильно и без ошибок. Подсчет очков производится корректно.

# **ГЛАВА 4**

# **ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОЕКТА**

## **4.1 Перспективы технического развития**

Перспективы развития игры:

* Возможность разместить игру на Android и IOS устройства;
* Добавить анимационное меню
* Добавить новых персонажей, карты, новые препятствия и оружия, чтобы убирать препятствия;
* Добавить встроенные платные покупки.

Добавление предложенного функционала сделает игру более удобным при использовании мобильного телефона. Расширенный функционал даст более положительные оценки за игру и положительные эмоции от игры.

## **4.2 Перспективы монетизации**

Для перспективы монетизации бесплатного приложения необходимо встроить платные дополнения. В качестве платного дополнения может выступать смена скина персонажа DF, смена локации и т.п. Платное дополнение предлагает подход для каждого пользователя: кто-то хочет получить побыстрее классный скин, а кто- то хочет сменить обстановку в игре, к примеру, бегать по морю.

# **ЗАКЛЮЧЕНИЕ**

В результате создана игра RUN DF. Игра выполняет свои функции. Для создания игры был использован один из основных языков программирования C++ и библиотеки SFML. C++ - компилируемый язык программирования общего назначения, сочетает свойства как высокоуровневых, так и низкоуровневых языков программирования.

Выполнены следующие **задачи:**

* Изучена тема о аналогах подобной игры
* Изучены теоретические аспекты одного из основных языков программирования C++
* Изучено создание библиотеки SFML
* Создать игру RUN DF с библиотекой SFML на языке программирование C++

Игра может быть использована, как и детям, так и взрослым, чтобы к примеру, скоротать время.

# **ЛИТЕРАТУРА**

1. Герберт Шилдт. C + + для начинающих. Пер. с англ. М: ЭКОМ Паблишерз 2007.
2. Visual C++ на примерах. Анатолий Хомоненко, Галина Довбуш, 2007. – 507 с.
3. Самоучитель Microsoft Visual Studio. Алексей Голощапов, 2010. – 543 с.
4. SFML Blueprints Cpp. Maxime Barbier, 2015. – 298 с.
5. История компьютерных игр [Электронный ресурс]. – группа в VK.: Твои любимые игры, 2019.
6. Бобровский, С. П. **Самоучитель программирования на языке C++/** С. П. Бобровский. - КУДИЦ-Пресс, 2013 – 238 с.
7. Герберт Шилдт **Полный справочник по C++** /Издание 4. 2015 год.
8. Страуструп, Б. Язык программирования C++: Специальное издание / Б. Страуструп; Пер. с англ. Н.Н. Мартынов. — М.: БИНОМ, 2017. — 1136 c.

# **ПРИЛОЖЕНИЕ 1**

# **КОД ОСНОВНЫХ МОДУЛЕЙ ПРОЕКТА**

Код модуля Source.cpp:

#include <SFML/Graphics.hpp>

#include <time.h>

using namespace sf;

int main()

{

srand(time(0));

RenderWindow window(VideoMode(1000, 500), "RUN DF");

Texture f;

f.loadFromFile("D:/KFU/Programming/CourseWork/Pic/fon.png");

Sprite fon(f);

Texture e;

e.loadFromFile("D:/KFU/Programming/CourseWork/Pic/earth.png");

Sprite earth[3];

for (int i = 0; i < 3; i++) {

earth[i].setTexture(e);

earth[i].setScale(1, 0.25);

earth[i].setPosition(0, 150 + 150 \* i);

}

Texture m;

m.loadFromFile("D:/KFU/Programming/CourseWork/Pic/man.png");

Sprite man(m);

man.setScale(0.8, 0.8);

man.setTextureRect(IntRect(0, 0, 80, 130));

float y = 200, frame = 0, py = 0, dy = 0, speed = 0.5;

bool up = false, down = false, anim = true;

Texture t;

t.loadFromFile("D:/KFU/Programming/CourseWork/Pic/text.png");

Sprite text(t);

text.setPosition(300, 50);

float fx = 0;

int game = 0;

Texture b;

b.loadFromFile("D:/KFU/Programming/CourseWork/Pic/block.png");

Sprite block[2];

int bx[2], by[2], bty[2];

for (int i = 0; i < 2; i++) {

block[i].setTexture(b);

bx[i] = rand() % 3 + 1;

by[i] = rand() % 2 + 1;

bty[i] = rand() % 3 + 1;

block[i].setTextureRect(IntRect(100 \* (bx[i] - 1), 100 \* (by[i] - 1), 100, 100));

block[i].setPosition(1000 + 500 \* i, 50 + 150 \* (bty[i] - 1));

}

Texture go;

go.loadFromFile("D:/KFU/Programming/CourseWork/Pic/gameover.png");

Sprite gameover(go);

gameover.setPosition(400, 50);

Texture sc;

sc.loadFromFile("D:/KFU/Programming/CourseWork/Pic/score.png");

Sprite score[5];

float fs[5];

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

score[i].setTexture(sc);

score[i].setTextureRect(IntRect(0, 0, 44, 55));

score[i].setPosition(50 \* i, 5);

fs[i] = 0;

}

while (window.isOpen())

{

Event event;

while (window.pollEvent(event))

{

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

window.close();

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

if (event.key.code == Mouse::Left)

if (game == 0)

game = 1;

if (game == 1)

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

if (event.key.code == Keyboard::Up) {

dy = -150;

up = true;

}

else if (event.key.code == Keyboard::Down) {

dy = 150;

down = true;

}

}

if (game == 0) {

frame += 0.001;

if (frame > 4)

frame -= 4;

man.setTextureRect(IntRect(80 \* int(frame), 0, 80, 130));

}

else if (game == 1) {

if (py != dy) {

if (dy > 0) {

py = speed;

dy = dy - speed;

}

if (dy < 0) {

py = -speed;

dy = dy + speed;

}

y += py;

anim = false;

}

else {

up = false;

down = false;

anim = true;

}

if (y < 55)

y = 55;

else if (y > 355)

y = 355;

if (anim) {

frame += 0.01;

if (frame > 6)

frame -= 6;

switch (int(frame))

{

case 0:

man.setTextureRect(IntRect(0, 135, 80, 130));

break;

case 1:

man.setTextureRect(IntRect(80, 135, 90, 130));

break;

case 2:

man.setTextureRect(IntRect(172, 135, 100, 130));

break;

case 3:

man.setTextureRect(IntRect(0, 270, 88, 130));

break;

case 4:

man.setTextureRect(IntRect(88, 270, 88, 130));

break;

case 5:

man.setTextureRect(IntRect(172, 270, 90, 130));

break;

}

}

else {

if (up)

man.setTextureRect(IntRect(0, 400, 80, 150));

if (down)

man.setTextureRect(IntRect(82, 400, 100, 150));

}

Vector2f eh[3];

for (int i = 0; i < 3; i++) {

earth[i].move(-0.4, 0);

eh[i] = earth[i].getPosition();

if (eh[i].x < -800)

earth[i].setPosition(0, 150 + 150 \* i);

}

Vector2f bk[2];

for (int i = 0; i < 2; i++) {

block[i].move(-0.4, 0);

bk[i] = block[i].getPosition();

if (bk[i].x < -100) {

bx[i] = rand() % 3 + 1;

by[i] = rand() % 2 + 1;

bty[i] = rand() % 3 + 1;

block[i].setTextureRect(IntRect(100 \* (bx[i] - 1), 100 \* (by[i] - 1), 100, 100));

block[i].setPosition(1000 + 500 \* i, 50 + 150 \* (bty[i] - 1));

}

if (man.getGlobalBounds().intersects(block[i].getGlobalBounds()))

game = 2;

}

fs[4] += 0.01;

if (fs[4] > 10) {

fs[4] -= 10;

fs[3]++;

if (fs[3] > 9) {

fs[3] -= 10;

fs[2]++;

if (fs[2] > 9) {

fs[2] -= 10;

fs[1]++;

if (fs[1] > 9) {

fs[1] -= 10;

fs[0]++;

if (fs[0] > 9)

fs[0] -= 10;

score[0].setTextureRect(IntRect(44 \* int(fs[0]), 0, 44, 55));

}

score[1].setTextureRect(IntRect(44 \* int(fs[1]), 0, 44, 55));

}

score[2].setTextureRect(IntRect(44 \* int(fs[2]), 0, 44, 55));

}

score[3].setTextureRect(IntRect(44 \* int(fs[3]), 0, 44, 55));

}

score[4].setTextureRect(IntRect(44 \* int(fs[4]), 0, 44, 55));

}

else if (game == 2)

man.setTextureRect(IntRect(190, 400, 100, 150));

window.clear(Color::White);

window.draw(fon);

for (int i = 0; i < 3; i++)

window.draw(earth[i]);

man.setPosition(100, y);

window.draw(man);

if (game == 0) {

fx += 0.005;

if (fx > 5)

fx -= 5;

if (fx < 3)

window.draw(text);

}

for (int i = 0; i < 2; i++)

window.draw(block[i]);

if (game == 2)

window.draw(gameover);

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

window.draw(score[i]);

window.display();

}

return 0;

}