Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Операционные системы и системное программирование (ОСИСП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

«Игровое программное средство “Jumper”»

БГУИР КП 1-40 01 01 408 ПЗ

Выполнил:

Студент гр. 851004

Дрозд А.С.

Проверил:

Шульга Е.С.

Минск, 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

––––––––––––––––––––––––

(подпись)

––––––––––––––––– 2020 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту    Дрозд Алексей Сергеевич, 851004

1. Тема работы  Игровое программное средство «Jumper»

2. Срок сдачи студентом законченной работы––02.09.2020 г.

3. Исходные данные к работе   Операционная система – Windows 10; среда разработки – Visual Studio 19; язык программирования – Cи, интерфейс программирования приложений Windows API. Проанализировать существующие аналоги. Спроектировать и реализовать алгоритмы игровой логики. Реализовать отображение игрового процесса в оконном приложении. Реализовать возможность приостановки и возобновления игрового процесса.

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение.

1. Теоретические сведения;

2. Определение и анализ требований;

 3. Разработка программного средства;

4.Тестирование программного средства;

5.Руководство пользователя.;

Заключение;

Список источников;

Приложение А;

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. «Игровое программное средство Jumper», «Работа игрового программного средства Jumper. Схема алгоритма».

6. Консультант по курсовой работе Шульга Е.С.

7. Дата выдачи задания 10.09.2020 г.–––––––––––––––––––––––   –

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1 к 28.09.2020 – 15 % готовности работы;

разделы 2 к 21.10.2020 – 30 % готовности работы;

разделы 3 к 18.11.2020 – 60 % готовности работы;

раздел 4, 5 к 01.12.2020 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 10.12.2020 – 100 % готовности работы.

Защита курсовой работы с 01.12.2020 по 18.12.2020 г.––––––––––––––––––––

РУКОВОДИТЕЛЬ–––––– E.С.Шульга

(подпись)

Задание принял к исполнению А.С.Дрозд

(дата и подпись студента)

Содержание

[Введение ………………………………………………………………...2](#_Toc59222610)

[1. Анализ предметной области и аналогов, формирование требований к программному средству 4](#_Toc59222611)

[1.1 Общие понятия 4](#_Toc59222612)

[1.2 Обзор аналогов 5](#_Toc59222613)

[1.3 Разработка функциональных требований 6](#_Toc59222614)

[2. Моделирование предметной области и разработка функциональных требований 8](#_Toc59222615)

[2.1 Описание функциональных требований к программному средству 8](#_Toc59222616)

[2.2 Спецификация функциональных требований 8](#_Toc59222617)

[2.3 Обзор средств и технологий разработки программного средства 8](#_Toc59222618)

[3. Проектирование и разработка программного средства 11](#_Toc59222619)

[3.1 Общая архитектура 11](#_Toc59222620)

[3.2 Проектирование модулей программного средства 12](#_Toc59222621)

[3.3 Разработка модулей программного средства 12](#_Toc59222622)

[4. Тестирование программного средства 16](#_Toc59222623)

[4.1 Тестирование игровой функциональности 16](#_Toc59222624)

[5. Руководство пользователя 19](#_Toc59222625)

[Заключение 20](#_Toc59222626)

[Список использованных источников 21](#_Toc59222627)

[Приложение A 22](#_Toc59222628)

**ВВедение**

Данный курсовой проект посвящён созданию игрового приложения «Jumper» на языке С++ средствами WinAPI.

Технический прогресс в последние несколько десятков лет сильно изменил жизнь обычного человека. Цифровые технологии ускорили развитие различных областей науки, позволили повысить качество жизни. Немалое влияние развитие информационных технологий оказало и на индустрию развлечений: видеоблогинг, стриминг, кино и, конечно, видеоигры[1].

История видеоигр начинается в 1950-х годах, когда создавались первые простые игры и симуляции. Изобретателем компьютерных игр обычно считают одного из трёх людей: Ральфа Бауэра (инженер, выдвинувший идею интерактивного телевидения в 1951 году), Александра Дугласа (разработчик «ОХО» – компьютерной реализации крестиков-ноликов) или Уильяма Хигинботама (создатель игры «Tennis for Two», 1958)[2].

Врачи, психологи и исследователи со всей планеты на протяжении долгих лет докладывают о пользе тетриса. В 2009 году было доказано, что эта игра повышает работоспособность головного мозга. Позднее выяснилось, что с её помощью можно преодолеть посттравматический синдром. А не так давно тетрис назвали отличным средством от чувства беспокойства и состояния тревоги. Причём версия, в которой скорость фигурок увеличивается по мере уничтожения линий, оказалась эффективнее обычной – она как бы безмолвно хвалит играющего, подкидывая ему всё более сложные испытания и позволяя на время отвлечься от проблем реальной жизни[3].

Простая игра с динозавром в главной роли — одно из ключевых «пасхальных яиц» браузера. Доступ к ней открывается при попытке открыть сайт без подключения к сети. Недавно в блоге Goolge появилось интервью с создателями проекта[4].

Игра — бесконечная «бегалка», в которой динозавру нужно помогать уворачиваться от возникающих на его пути препятствий. Пользователь может нажимать на «прыжок» или «приседать», а бег начинается автоматически. Задача — набрать как можно больше очков (они насчитываются за время в игре). Или дождаться, пока интернет снова не начнёт работать.

По словам дизайнера Chrome **Себастьена Габриэля**, динозавтра в качестве героя выбрали для отсылки к «доисторическим временам», когда подключиться к сети можно было только в конкретных местах: дома, в школе, на работе или в интернет-кафе. А самому тиранозавру дали имя «Проект Болан» — в честь **Марка Болана**, вокалиста глэм-рок группы T-Rex из 70-х.

Игра впервые вышла в сентябре 2014 года, но добиваться стабильности на разных Android-устройствах пришлось ещё несколько месяцев — до декабря 2014-го.

Google утверждает, что каждый месяц пользователи Android и Chrome запускают игру 270 млн раз, а наибольшую популярность она получила в регионах со нестабильным интернетом: Индии, Бразилии, Индонезии и Мексике.

В десктоп-версии браузера её можно запустить, введя chrome://dino в адресную строку. В этом случае пользователь получит доступ к полноэкранной версии разработки. А в честь 4-летнего юбилея в Chrome для ПК добавили праздничне мотивы: торты, надувные шары и праздничные головные уборы.

Создатели напоминают, что не стоит стремиться к выигрышу: игра автоматически заканчивается через 17 млн лет игрового времени — примерно столько тиранозавры просуществовали на Земле[6].

1. **Анализ Предметной области и аналогов, формирование требований к программному средству**
   1. **Общие понятия**

Компьютерная игра — [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), служащая 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 выступающая 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

Компьютерные игры классифицируют по нескольким основным признакам:

* жанр;
* количество игроков;
* визуальное представление;
* платформа.

Жанр игры определяется целью и основной механикой игры:

* action;
* аркада;
* симуляторы/Менеджеры;
* стратегии;
* приключения;
* музыкальные игры;
* ролевые игры;
* головоломки, логические, пазлы;
* платформеры;
* шутер.

По количеству игроков игры разделяются на два вида:

* однопользовательские;
* многопользовательские.

По визуальному представлению компьютерные игры можно разделить наследующие виды:

* текстовые – минимальное графическое представление, общение с игроком проходит с помощью текста;
* 2D – все элементы отрисованы в виде двумерной графики;
* 3D – все элементы отрисованы в виде трехмерной.

По типу платформы:

* персональные компьютеры;
* игровые приставки/консоли;
* мобильные телефоны.

Представленная классификация не является полной и может быть дополнена.

* 1. **Обзор аналогов**

Первой зарегистрированной игрой, которую можно отнести к видеоигре, является «Ракетный симулятор», интерфейсом которого выступало аналоговое устройство на базе электронно-лучевой трубки. Однако, из-за того, что  «Ракетный симулятор» не имел растрового типа изображения, которое характерно для мониторов и телевизоров, то данное устройство чаще принимают за [электронную игру](https://en.wikipedia.org/wiki/Electric_game), но относят к видеоиграм. Но вместе с тем, с технической точки зрения, «Ракетный симулятор» является видеоигрой.

Спустя несколько десятилетий, в 1972 году, Нолан Башнелл выпустил всеми известную игру под названием «Pong», сутью которой было удержание мячика на игровом поле с помощью специальных тарелок. Это событие стало одним из самых важных в истории развития компьютерных игр. Далее стали появляться первые игры в жанре бродилок, приключений, гонок и других. Еще через несколько лет была выпущена первая игровая приставка с одной единственной игрой, а затем и ее аналоги с несколькими играми. С тех пор персональные компьютеры и игровые приставки усовершенствовались, а игры для них становились функциональней, интересней и ярче. Появился целый рынок компьютерной индустрии, который по своим оборотам и вложениям обгонял даже рынок киноиндустрии.

Простой динозаврик в chrome является лучшей и неповторимой игрой, которая затягивает и спасла google от миллионов недовольных отзывов при отсутствии интернета. Это просто и гениальное решение как убить время в

процессе ожидания подключения к сети. И зачастую многие заходят в нее поиграть просто так (см. рисунок 1.1).

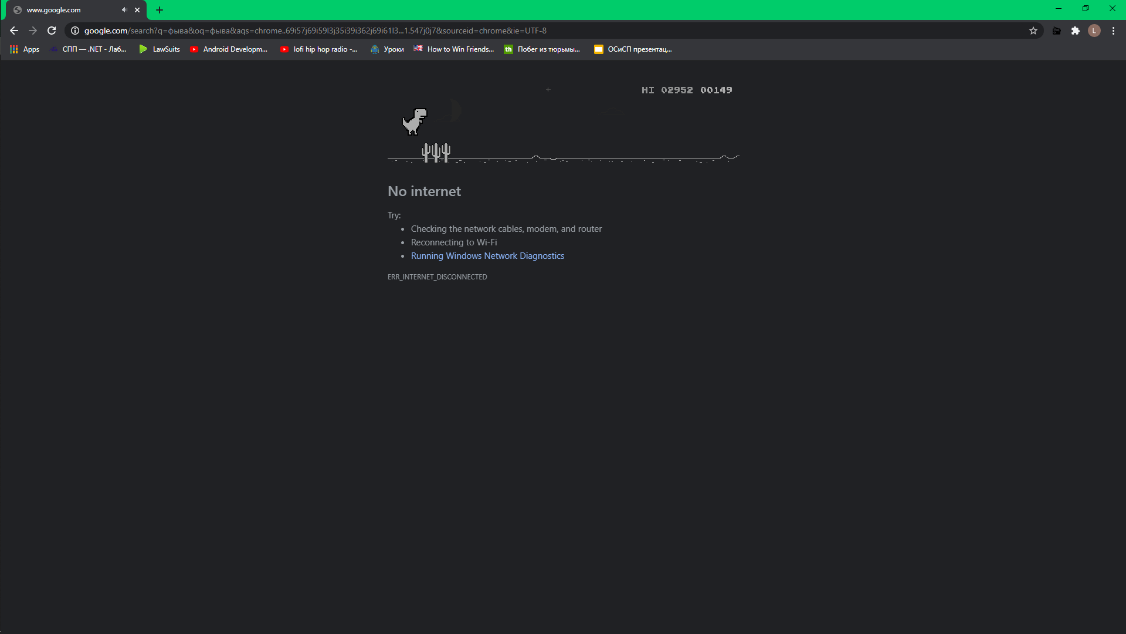


Рисунок 1.1 – Динозаврик

В Microsoft Edge появился продвинутый аналог игры про динозавра из Google Chrome. Разработчики из компании Microsoft подготовили для пользователей браузера Edge настоящий подарок — начиная со сборки 83.0.478.37 каждый сможет отдохнуть в видеоигре на тему сёрфинга, которая выглядит достаточно привлекательно, особенно если учитывать монохромного динозавра-конкурента из браузера Google Chrome. Для доступа к новой игре пользователю нужно перейти в автономный режим (отключение интернета) или просто ввести в адресной строке edge://surf. Стоит отметить, что изначально данная игра была доступна инсайдерам ещё с конца февраля, но теперь поиграть могут абсолютно все желающие на стабильной версии браузера, которую можно скачать с официального сайта (см. рисунок 1.2)[7].



Рисунок 1.2 – Surf

* 1. **Разработка функциональных требований**

В рамках данного курсового проекта планируется разработать игровое приложение «Jumper» с помощью средств WinAPI.

По результатам анализа аналогичных программных средств было принято решение отдать предпочтение обеспечению минималистичного дизайна и интуитивно понятного геймплея.

Планируется реализовать следующие функциональные возможности:

* игра в оконном режиме;
* возможность умереть;
* возможность остановить игру и возобновить ее позже;
* возможность менять цвет и проходит сквозь препятствия.

Для разработки коммуникационного модуля будет использоваться язык программирования С++, набор базовых функций интерфейса программирования приложений операционных систем Microsoft Windows WinAPI, среда разработки Microsoft Visual Studio.

1. **моделирование предметной области И разрабОтка функциональных требований**

**2.1 Описание функциональных требований к программному средству**

Проектируемое игровое приложение должно обеспечить пользователя необходимым функционалом для взаимодействия с игровым приложением, предоставить свободу выбора действий, контролировать ошибочные операции, реализованные пользователем, иметь логическое начало и завершение, проверять соблюдение исконно заверенных правил игры.

Ориентируясь на указанные задачи, я разработала игровое приложение, которое выполняет следующие функции:

* Предоставляет свободу выбора действий;
* Определяет логическое начало игры;
* Проверяет соблюдение исконно заверенных правил игры;
* Обеспечивает сохранность целостности и общности законов игры;
* Определяет логическое завершение игры.

**2.2 Спецификация функциональных требований**

Основные функции игрового приложения – анимация движение, автоматическая генерация препятствий, правила прохода препятствий. Более подробно функциональные требования могут быть описаны следующим образом:

* создание описательных структур для отрисовки препятствий;
* создание описательных структур для отрисовки игрока;
* корректный выбор значений для имитации притяжения;
* организация случайного выбора препятствия, которое появится на игровом поле следующей;
* предоставление игроку возможности остановить игру и возобновить ее позже;
* подсчет очков, заработанных игроком.

**2.3 Обзор средств и технологий разработки программного средства**

В данном разделе будут рассмотрены особенности средств и технологий, используемых для реализации проектируемого программного средства.

Для реализации проектируемого программного средства использовался язык программирования С++. С++ — [компилируемый](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D0%B8%D1%80%D1%83%D0%B5%D0%BC%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), [статически типизированный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [язык 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) общего назначения.

Поддерживает такие [парадигмы программирования](https://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](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%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), [объектно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%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%D0%BE%D0%B5_%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), [обобщённое программирование](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%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%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как [высокоуровневых](https://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](https://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). В сравнении с его предшественником — языком [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%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)) — наибольшее внимание уделено поддержке [объектно-ориентированного](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%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%D0%BE%D0%B5_%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 [обобщённого программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%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%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), разнообразных прикладных программ, [драйверов](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80) устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр.

Синтаксис C++ унаследован от языка [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%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)). Одним из принципов разработки было сохранение совместимости с C. Тем не менее C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как [компиляторами](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Таким образом, язык С++ является оптимальным для реализации игрового приложения «Тетрис».

Так же для реализации проектируемого программного средства была использована среда разработки 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 средств. 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 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)).

Таким образом, Micrisoft Visual Studio является наиболее подходящей средой разработки для создания игрового приложения на языке С++, так как предоставляет большое разнообразие инструментов разработки.

Набор базовых функций интерфейсов программирования приложений операционных систем Microsoft Windows WinAPI был полезен для реализации проектируемого программного средства. WinAPI предоставляет прямой способ взаимодействия приложений пользователя с операционной системой Windows. Для создания программ, использующих Windows API, корпорация «[Майкрософт](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B9%D0%BA%D1%80%D0%BE%D1%81%D0%BE%D1%84%D1%82)» выпускает [комплект разработчика программного обеспечения](https://ru.wikipedia.org/wiki/SDK), который называется [Platform SDK](https://ru.wikipedia.org/wiki/Platform_SDK) и содержит документацию, набор [библиотек](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)), утилит и других инструментальных средств для разработки.

Windows API спроектирован для использования в языке [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%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)) для написания [прикладных программ](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), предназначенных для работы под управлением операционной системы MS Windows. Работа через Windows API — это наиболее близкий к операционной системе способ взаимодействия с ней из прикладных программ. Более низкий [уровень доступа](https://ru.wikipedia.org/w/index.php?title=%D0%A3%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0&action=edit&redlink=1), необходимый только для [драйверов](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80) устройств, в текущих версиях Windows предоставляется через [Windows Driver Model](https://ru.wikipedia.org/wiki/Windows_Driver_Model).

Windows API представляет собой множество функций, структур данных и числовых констант, следующих соглашениям языка Си. В то же время [конвенция вызова](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B3%D0%BB%D0%B0%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BE_%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2%D0%B5) функций отличается от cdecl, принятой для языка C: Windows API использует stdcall (winapi). Все языки программирования, способные вызывать такие функции и оперировать такими типами данных в программах, исполняемых в среде Windows, могут пользоваться этим API. В частности, это языки [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [Pascal](https://ru.wikipedia.org/wiki/Pascal), [Visual Basic](https://ru.wikipedia.org/wiki/Visual_Basic) и многие другие.

Для облегчения программирования под Windows в компании Microsoft и сторонними разработчиками было предпринято множество попыток создать библиотеки и среды программирования, частично или полностью скрывающие от программиста особенности Windows API и предоставляющие ту или иную часть его возможностей в более удобном виде. В частности, сама Microsoft в разное время предлагала библиотеки [Active Template Library](https://ru.wikipedia.org/wiki/Active_Template_Library) (ATL)/[Windows Template Library](https://ru.wikipedia.org/wiki/Windows_Template_Library) (WTL), [Microsoft Foundation Classes](https://ru.wikipedia.org/wiki/Microsoft_Foundation_Classes) (MFC), [.Net](https://ru.wikipedia.org/wiki/.Net)/[WinForms](https://ru.wikipedia.org/wiki/WinForms)/[WPF](https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation), TXLib. Компания Borland (ныне Embarcadero, её преемник в части средств разработки) предлагала [OWL](https://ru.wikipedia.org/wiki/Object_Windows_Library) и [VCL](https://ru.wikipedia.org/wiki/Visual_Component_Library). Есть кросс-платформенные библиотеки, такие как [Qt](https://ru.wikipedia.org/wiki/Qt), [Tk](https://ru.wikipedia.org/wiki/Tk) и многие другие. Весомая часть этих библиотек сконцентрирована на облегчении программирования [графического интерфейса пользователя](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).

Таким образом, WinAPI является подходящим инструментом для разработки игрового приложения для операционной системы Windows.

1. **ПРОЕКТИРОВАНИЕ и разработка ПРОГРАММНОГО СРЕДСТВА**

**3.1 Общая архитектура**

Проектирование архитектуры программного средства – это проектирование структуры всех компонент программного средства, функционально связанных с решаемой задачей, связей между этими компонентами.

Архитектура программного средства – это базовая организация системы, воплощенная в ее компонентах, их отношениями между собой и окружением, а также принципы, определяющие проектирование и развитие системы.

Программным средством, разрабатываемым в рамках данного курсового проекта является игровое приложение «Jumper», которое будет спроектировано в рамках модульной архитектуры.

Модульное программирование — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам. Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

Модуль — функционально законченный фрагмент [программы](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). Во многих языках (но далеко не обязательно) оформляется в виде отдельного [файла](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB) с [исходным кодом](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) или поименованной непрерывной её части. Некоторые языки предусматривают объединение модулей в [пакеты](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_(%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)).

Роль модулей могут играть [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), [библиотеки функций](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)), [классы](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)), [сервисы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B8%D1%81%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%D0%B0%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0) и др. программные единицы, реализующие некоторую функциональность и предоставляющие [интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%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%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) к ней.

Программный код часто разбивается на несколько файлов, каждый из которых [компилируется](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%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)) отдельно от остальных. Такая модульность программного кода позволяет значительно уменьшить время перекомпиляции при изменениях, вносимых лишь в небольшое количество исходных файлов, и упрощает групповую [разработку](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%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). Также это возможность замены отдельных компонентов (таких как [jar-файлы](https://ru.wikipedia.org/wiki/Jar), so или dll библиотеки) конечного программного продукта, без необходимости пересборки всего проекта (например, разработка [плагинов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD) к уже готовой программе).

С точки зрения разработки, существует множество различных способов реализации игрового приложения «Jumper». Их можно разделить по функциональности:

* Математическое описание логики и графическое отображение препятствий;
* Математическое описание логики и графическое отображение игрового юнита.

**3.2 Проектирование модулей программного средства**

Проектируемое программное средство будет являться игровым приложением, состоящей из двух компонентов – модуля основного кода игры и модуля, хранящего в себе классы игровых объектов для дальнейшего взаимодействия с ними. Для реализации проектируемого программного средства будут использованы такие модули, как:

* SimpleJumper.cpp;
* SimpleJumper.h;
* Color.h;
* Hind.h;
* Player.h.

SimpleJumper.cpp – служит для запуска игры. Содержит в себе основную логику работы игрового приложения, остановки и возобновления игры, подсчета очков, набранных игроком.

Hind.h, Player.h – содержит структурные описания фигур для дальнейшего взаимодействия с ними.

**3.3 Разработка модулей программного средства**

Метод WinMain вызывается системой как начальная точка входа, для базирующейся на Win32, прикладной программы.

Синтаксис: int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow)

Параметры:

1. hinstance - Идентифицирует текущий образец прикладной программы.

2. hPrevInstance - Идентифицирует предыдущий образец прикладной программы. Для базирующейся на Win32 прикладной программы, этот параметр всегда имеет значение ПУСТО (NULL).

3. lpCmdLine - Указывает на строку с нулевым символом в конце, определяющую командную строку для прикладной программы.

4. nCmdShow - Определяет, как окно должно быть показано.

WinMain инициализирует прикладную программу, отображает на экране её основное окно и, затем, входит в цикл сообщений «поиск и отправка по назначению (диспетчеризация)», который является управляющей структурой верхнего уровня для остальных элементов выполнения прикладной программы. Цикл сообщений завершается, когда получено сообщение WM\_QUIT. В этой точке WinMain завершает работу прикладной программы, возвращая значение, переданное в параметр wParam сообщения WM\_QUIT. Если WM\_QUIT было получено в результате вызова PostQuitMessage, значение wParam – значение параметра nExitCode функции PostQuitMessage.

В методе WinMain происходит инициализация параметров окна, задается функция, которая будет обрабатывать приходящие сообщения (wndclass.lpfnWndProc = WndProc;), происходит отображение окна, начальная информация, организуется цикл обработки сообщений.

Метод WndProc производит обработку сообщений для окна. Он связывается с окном при заполнении структуры класса окна WNDCLASS в поле lpfnWndProc: (wndclass.lpfnWndProc = WndProc;). В методе WndProc организуется конструкция switch-case, в которой будут реализованы обработчики сообщений, приходящих окну.

Реализованы обработчики сообщений:

1. WM\_CREATE – сообщение приходит при создании окна. Когда это сообщение пришло, создается 2 таймера, один вызывает постоянную перерисовку, второй – ускоряет первый таймер для видимости усложнения игры со временем.

2. WM\_SIZE – сообщение приходит при изменении размеров окна. В случае изменений размеров окна, приложение будет подстраиваться под эти размеры.

3. WM\_KEYDOWN – сообщение приходит при нажатии пользователем какой-либо клавиши на клавиатуре или компьютерной мыши. При получении такого сообщения нажатая кнопка будет обрабатываться в соответствии со своей задачей.

4. WM\_DESTROY - посылается тогда, когда окно разрушается. Оно посылается оконной процедуре разрушаемого окна после того, как окно удалено с экрана. Это сообщение посылается сначала разрушаемому окну, а затем дочерним окнам (если таковые имеются) когда они разрушаются. В ходе обработки сообщения, оно может быть принято, так как все дочерние окна все еще существуют. Когда будет получено такое сообщение – будет вызвана функция KillTimer(), уничтожающая таймер игры, а также функция PostQuitMessage(), которая поместит сообщение WM\_QUIT в очередь сообщений. В последствии его отловит PeekMessage() и программа прекратит работу.

Метод TimerProc осуществляет всего одно действие – вызов функции InvalidateRect(), которая будет вызывать событие WM\_PAINT, в котором будет реализована логика игры.

Логика работы с WinMain описана на рисунке 3.1.

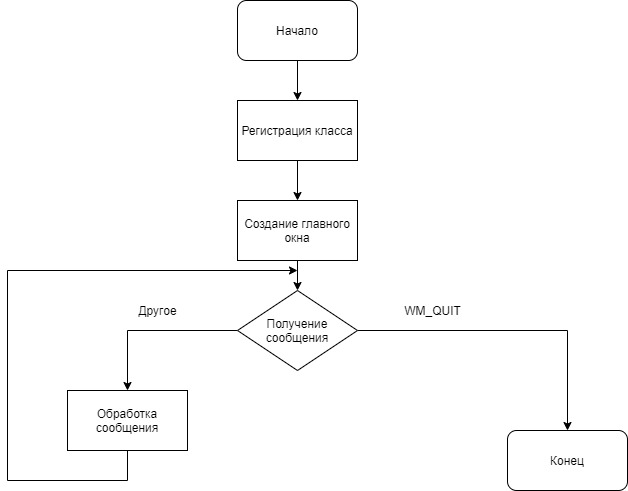


Рисунок 3.1 – winMain

Для понимания логики работы игры была написана схема (рисунок 3.2).

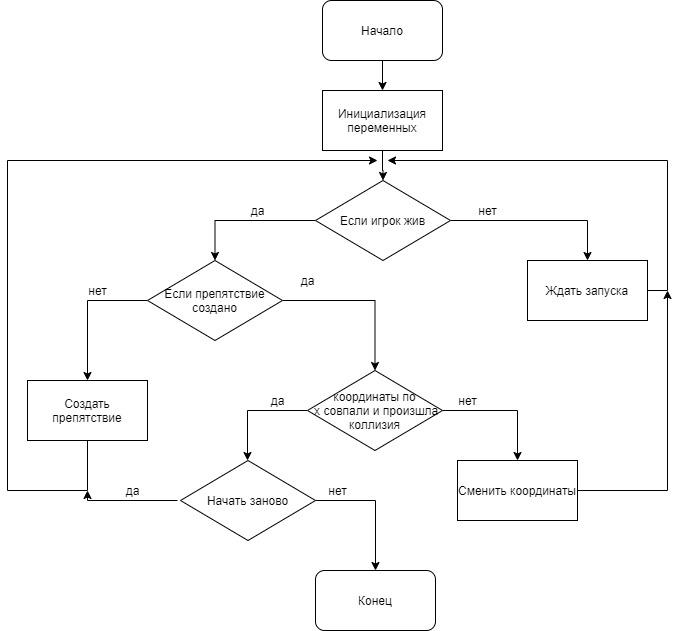


Рисунок 3.3.2 – Логика работы игры

**4. Тестирование Программного средства**

**4.1 Тестирование игровой функциональности**

Таблица 1 – Тестирование прыжка

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка реализации прыжка |
| Команда | Нажатие на клавишу space |
| Фактический результат: | Игрок подпрыгивает и плавно опускается на землю (см. Рисунок 4.1, Рисунок 4.2). |

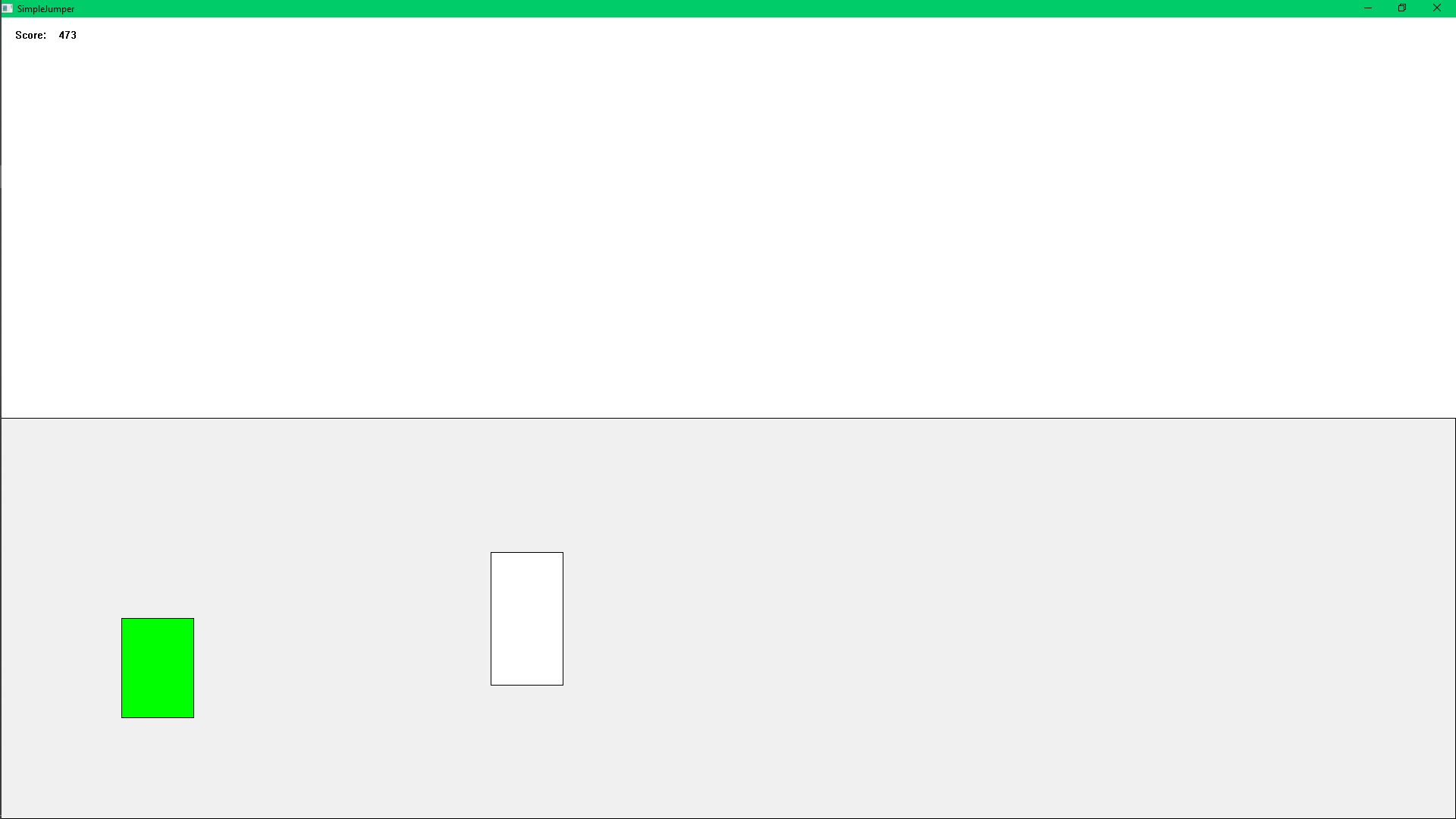


Рисунок 4.1 – Тестирование прыжка

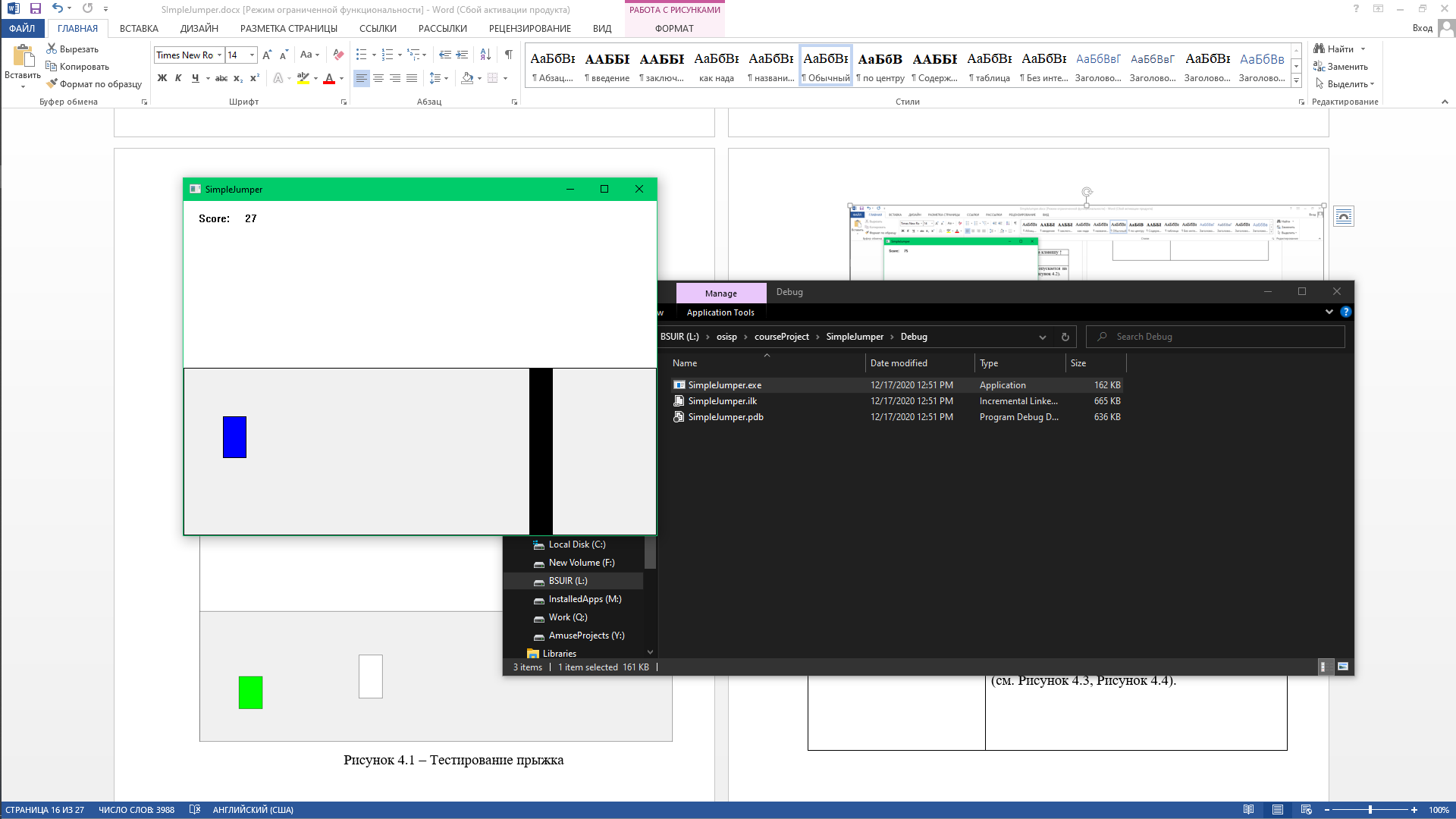


Рисунок 4.2 – Тестирование прыжка

В Таблице 2 приведена последовательность действий для тестирования реализации нажатия на клавиши 1,2,3,4,5.

Таблица 2 - Тестирование реализации смены цвета

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка реализации смены цвета |
| Команда | Нажатие на клавиши 1,2,3,4,5 |
| Фактический результат: | Смена цвета  (см. Рисунок 4.3, Рисунок 4.4). |

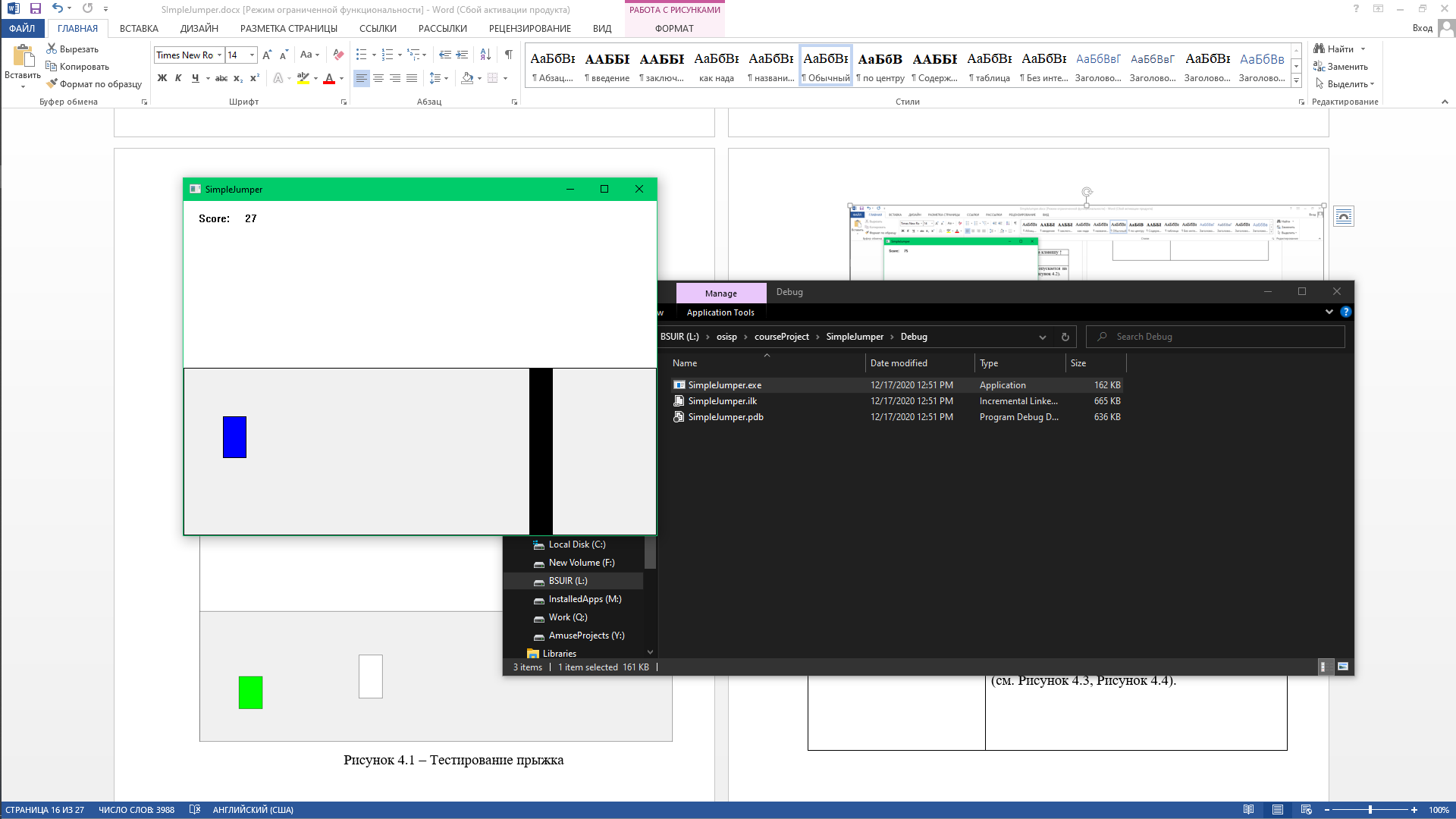


Рисунок 4.3 – Тестирование реализации смены цвета

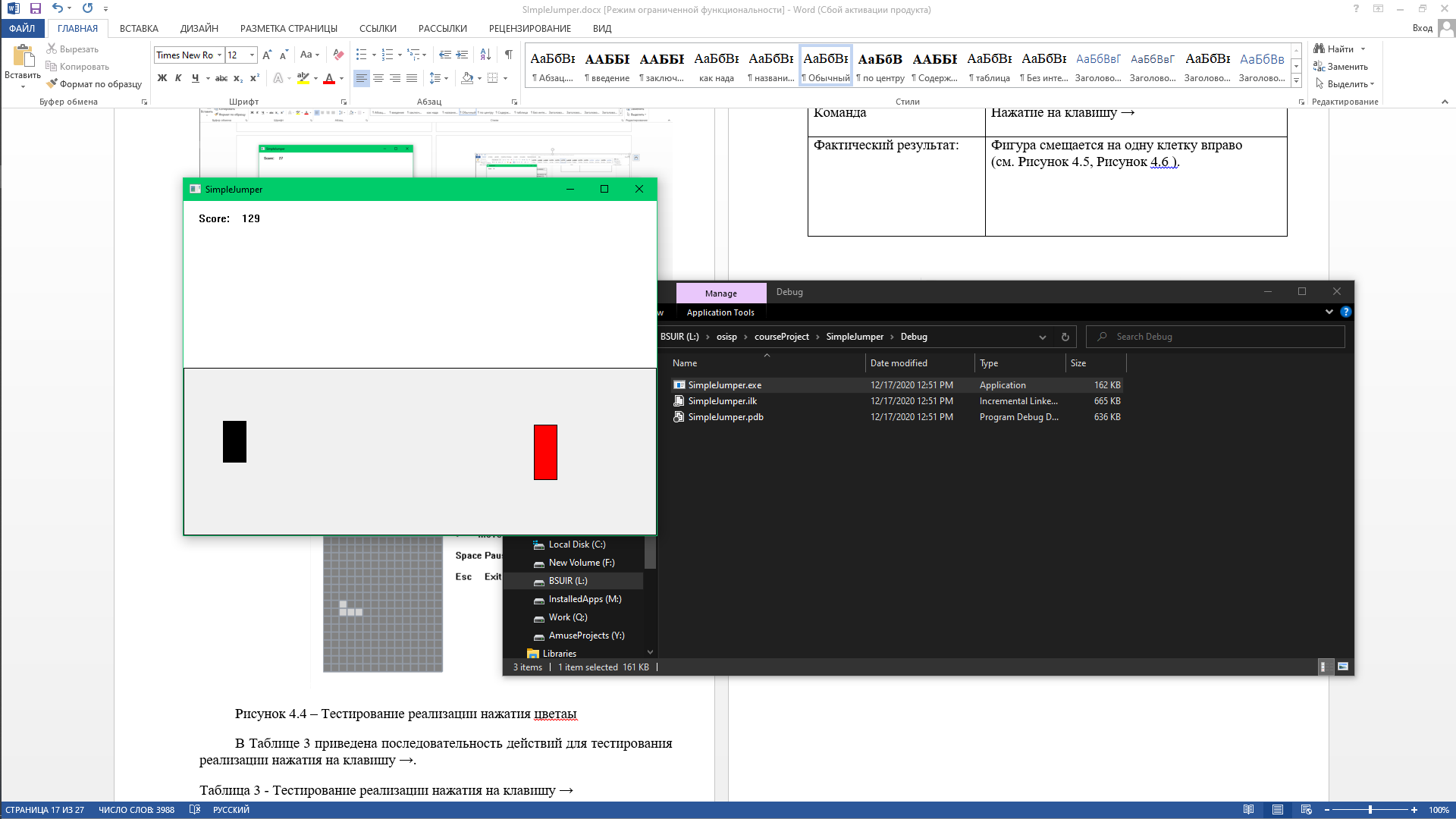


Рисунок 4.4 – Тестирование реализации нажатия цвета

**5. Руководство поЛЬЗОВАТЕЛЯ**

Для успешной работы игрового приложения необходима операционная система Microsoft Windows 10.

Игрок может управлять посредством нажатия клавиш space,1,2,3,4,5. При нажатии на клавиши цветов, фигура поменяет цвет и сможет проходить сквозь препятствия того же цвета. У игры нет конца, счет идет до бесконечности (пока игрок не проиграет). Игрок может ставить игру на паузу нажатием клавиши F2, а также может возобновить игру повторным нажатием на F2.

**Заключение**

В рамках данной курсовой работы была решена поставленная задача. Задача представляет собой windows-приложение, эмулирующее игру «Jumper». Приложение имеет интуитивно-понятный интерфейс. В ходе тестирования приложение показало себя стабильным и устойчивым к некорректным действиям пользователя.

Для разработки приложения использовался язык высокого уровня C++. Средой разработки была выбрана среда Microsoft Visual Studio 2008.

**Список использованных источников**

[1] Ричард Вайнер, Льюис Пинсон, "С++ изнутри" Киев, "ДиаСофт", 1993 г.

[2] Элджер Дж. С++: библиотека программиста. - СПб.: ЗАО <Издательство Питер, 2009 г.

[3] Стивен Романс, Win32 API programming with Visual Basic - "O'REILLY", 1999 г.

[4] История компьютерных игр (internet версия).

[5] Bjarne Stroustrup, The C++ Programming Language/ Addison-Wesley Professional; 4 edition (May 19, 2013)

[6] https://dev.by/news/chrome-dino-game-explained

[7] https://blogs.windows.com/msedgedev/2020/05/26/surf-game-edge-stable/

Приложение А

(обязательное)

Исходный код программы

#include "SimpleJumper.h"

#include "Hind.h"

#include <Windows.h>

#include <math.h>

#include <stdlib.h>

#include "Color.h"

#include "Player.h"

#define TIMER\_NEXT\_FRAME 1

#define TIMER\_ACCELERATION 2

#define TIME\_ACCELERATION 10000

#define TIME\_INTERVAL 40

HWND mainWindow;

RECT mainWindowSize,gameRegion;

HDC context;

Hind hind;

COLORREF backColorDay;

COLORREF backColorNight;

Color colorstruct;

Player player;

int score = 0;

int curSpeed = 0;

int acceleration = 2;

bool pause = true;

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM , LPARAM);

VOID CALLBACK TimerProc(HWND, UINT, UINT, DWORD);

VOID CALLBACK TimerAcceleration(HWND, UINT, UINT, DWORD);

PAINTSTRUCT FillPaintStruct(RECT);

void GameLogic(HDC);

void DrawGameField(HDC);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

{

WNDCLASSEX wcex; MSG msg;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_DBLCLKS;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = L"SimpleJumperClass";

wcex.hIconSm = wcex.hIcon;

RegisterClassEx(&wcex);

mainWindow = CreateWindow(L"SimpleJumperClass", L"SimpleJumper",

WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT,

640, 480, NULL, NULL, hInstance, NULL);

//SetWindowLongPtr(mainWindow, GWL\_STYLE, WS\_VISIBLE | WS\_POPUP);

GetClientRect(mainWindow, &mainWindowSize);

GetClientRect(mainWindow, &gameRegion);

gameRegion.top += gameRegion.bottom/2 ;

backColorDay = RGB(240, 240, 240);

backColorNight= RGB(100, 100, 100);

ShowWindow(mainWindow, nCmdShow);

UpdateWindow(mainWindow);

while (1)

{

if (PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE)) {

if (msg.message == WM\_QUIT) break;

TranslateMessage(&msg);

DispatchMessage(&msg);

}

else {

}

}

return (int)msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_CREATE:

{

SetTimer(hWnd, TIMER\_NEXT\_FRAME , TIME\_INTERVAL, (TIMERPROC)TimerProc);

}break;

case WM\_ACTIVATE:

{

// TODO: set function Pause()

}

break;

case WM\_COMMAND:

{

}

break;

case WM\_KEYDOWN:

{

switch (wParam) {

case 49:

player.ChangeColor(Color::red);

break;

case 50:

player.ChangeColor(Color::blue);

break;

case 51:

player.ChangeColor(Color::green);

break;

case 52:

player.ChangeColor(Color::black);

break;

case 53:

player.ChangeColor(Color::white);

break;

case VK\_SPACE:

player.Jump();

break;

case VK\_F1:

if (!player.exists) {

score = 0;

curSpeed = TIME\_INTERVAL;

SetTimer(hWnd, TIMER\_ACCELERATION, TIME\_ACCELERATION, (TIMERPROC)TimerAcceleration);

player.fillPlayer(gameRegion, colorstruct.getRandColor());

}

break;

case VK\_F2:

if (pause) {

KillTimer(hWnd, TIMER\_ACCELERATION);

KillTimer(hWnd, TIMER\_NEXT\_FRAME);

pause = !pause;

}

else {

SetTimer(hWnd, TIMER\_ACCELERATION, TIME\_ACCELERATION, (TIMERPROC)TimerAcceleration);

SetTimer(hWnd, TIMER\_NEXT\_FRAME, curSpeed, (TIMERPROC)TimerProc);

pause = !pause;

}

break;

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

ps = FillPaintStruct(mainWindowSize);

HDC hdc = BeginPaint(hWnd, &ps);

// HDC hdcComp = CreateCompatibleDC(hdc);

// HBITMAP bitmap = CreateCompatibleBitmap(hdcComp, mainWindowSize.right-mainWindowSize.left, mainWindowSize.bottom-mainWindowSize.top);

// SelectObject(hdcComp, bitmap);

DrawGameField(hdc);

GameLogic(hdc);

//BitBlt(hdc, 0, 0, gameRegion.right, gameRegion.bottom, hdcComp, 0, 0, SRCCOPY);

//DeleteObject(bitmap);

//DeleteDC(hdcComp);

DeleteDC(hdc);

EndPaint(hWnd, &ps);

}

break;

case WM\_TIMER:

{

}

break;

case WM\_SIZE:

{

GetClientRect(mainWindow, &mainWindowSize);

GetClientRect(mainWindow, &gameRegion);

gameRegion.top += gameRegion.bottom / 2;

if (hind.exists)

hind.Resize(gameRegion);

if (player.exists)

player.Resize(gameRegion);

InvalidateRect(mainWindow, NULL, TRUE);

}

break;

case WM\_DESTROY:

KillTimer(mainWindow,TIMER\_NEXT\_FRAME);

KillTimer(mainWindow, TIMER\_ACCELERATION);

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

PAINTSTRUCT FillPaintStruct(RECT winSize) {

PAINTSTRUCT ps;

ps.fErase = true;

ps.rcPaint = gameRegion;

return ps;

}

VOID CALLBACK TimerProc(HWND hWnd, UINT uMessage, UINT uEventId, DWORD dwTime) {

InvalidateRect(hWnd, NULL, TRUE);

}

VOID CALLBACK TimerAcceleration(HWND hWnd, UINT uMessage, UINT uEventId, DWORD dwTime) {

KillTimer(hWnd, TIMER\_NEXT\_FRAME);

curSpeed -= acceleration;

SetTimer(hWnd, TIMER\_NEXT\_FRAME, curSpeed, (TIMERPROC)TimerProc);

}

void GameLogic(HDC hdc) {

TCHAR score\_str[50];

// Plotting the current score

wsprintf(score\_str, TEXT("Score: %6d "), score);

TextOut(hdc, 20, 15, score\_str, lstrlen(score\_str));

if (player.exists) {

player.Decending();

player.DrawPlayer(hdc);

if (hind.exists) {

hind.MoveHind();

hind.DrawHind(hdc);

if (hind.x < 0) {

hind.emptyHind();

}

if (abs(player.x+player.width - hind.x) <= hind.step) {

if (((player.y + player.height >= hind.y) && (hind.type == HindType::onGroundShort)) ||

((player.y <= hind.y + hind.height) && (hind.type == HindType::FlyingShort))||

((player.color!=hind.color)&&(HindType::onGroundLong==hind.type))) {

hind.exists = false;

player.Die();

KillTimer(mainWindow, TIMER\_ACCELERATION);

KillTimer(mainWindow, TIMER\_NEXT\_FRAME);

}

}

else {

score++;

}

}

else {

HindType hindType = HindType::onGroundShort;

COLORREF color = colorstruct.getRandColor();

switch (rand() % 3) {

case 0:

hindType = HindType::onGroundShort;

break;

case 1:

hindType = HindType::onGroundLong;

break;

case 2:

hindType = HindType::FlyingShort;

break;

}

hind.fillHind(hindType, gameRegion, color);

}

}

else {

TextOut(hdc,20, 50, L"Press F1 to start", 18);

TextOut(hdc, 20, 70, L"Press F2 to pause", 18);

}

}

void DrawGameField(HDC hdc) {

HBRUSH brush = CreateSolidBrush(backColorDay);

SelectObject(hdc, brush);

Rectangle(hdc, gameRegion.left, gameRegion.top, gameRegion.right, gameRegion.bottom);

DeleteObject(brush);

}

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 408 ПЗ | | | | Пояснительная записка | | | | 26 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 851004  408 ПД | | | | Работа игрового программного средства Jumper. Схема алгоритма | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 408 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Игровое программное средство «Jumper»  Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Дрозд А.С |  |  | Т |  | |  | 26 | 26 |
| Пров. | | Шульга Е.С |  |  | Кафедра ПОИТ  гр. 851004 | | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |