МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО» ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

**ИГРА “OIRAM”**

Курсовая работа

по дисциплине «Программирование»

студента 1 курса группы ПИ-б-о-202(1)

Пислякова Олега Игоревича

направления подготовки 09.03.04 «Программная Инженерия»

|  |  |  |
| --- | --- | --- |
| Научный руководитель  Старший руководитель кафедры компьютерной инженерии и моделирования | (оценка)    (подпись, дата) | Тимофеева С.В. |

Симферополь, 2021

# РЕФЕРАТ

Писляков О.И. Разработка игры «OIRAM» в жанре «Приключение»

Курсовая работа по специальности 09.03.04 Программная инженерия / Кафедра компьютерной инженерии и моделирования Физико-технического института Крымского федерального университета им. В. И. Вернадского. – Симферополь, 2021. Количество страниц – 22.

**Объект исследования** – компьютерная игра в жанре “Приключение”.

**Предмет исследования** – особенности разработки/методика разработки компьютерной игры в жанре “Приключение”.

**Цель работы** – разработать игру «OIRAM» с использованием языка программирования C++ и графической библиотеки SFML.

Аннотация работы – в работе проведён анализ инструментальных средств для разработки игры в жанре «Приключение», была определена концепция игрового процесса. Описан процесс тестирования и разработки. В конечном счёте был получен рабочий программный продукт. Были закреплены навыки проектирования программного продукта.

# ОГЛАВЛЕНИЕ

[РЕФЕРАТ 2](#_Toc86043594)

[ОГЛАВЛЕНИЕ 3](#_Toc86043595)

[ВВЕДЕНИЕ 4](#_Toc86043596)

[ГЛАВА 1 7](#_Toc86043597)

[ГЛАВА 2 8](#_Toc86043598)

[Постановка задачи 8](#_Toc86043599)

[2.1 Описание алгоритмов 9](#_Toc86043600)

[2.2 Описание структур данных 11](#_Toc86043601)

[2.3 Описание основных классов 14](#_Toc86043602)

[ГЛАВА 3 18](#_Toc86043603)

[ГЛАВА 4 19](#_Toc86043604)

[ЗАКЛЮЧЕНИЕ 21](#_Toc86043605)

[ЛИТЕРАТУРА 22](#_Toc86043606)

# ВВЕДЕНИЕ

Для создания проекта использовалась программа Visual Studio 2017. Проект создавался методом проб и ошибок, с постоянными изменениями и улучшениями в плане удобства для пользователя. Исходя из того, что нет предела совершенству, можно сказать, что программа не окончена и требует дополнений.

Платформеры появились в начале 1980-х, когда игровые консоли не были достаточно мощными, чтобы отображать трёхмерную графику или видео. Они были ограничены технологиями своего времени и были представлены статическими игровыми мирами, которые помещались на один экран, а игровой герой был виден в профиль. Персонаж лазал вверх и вниз по лестницам или прыгал с платформы на платформу, часто сражаясь с противниками и собирая предметы. Первыми играми этого типа были *Space Panic* и *Apple Panic*. За ними последовала игра *Donkey Kong*, аркадная игра, созданная фирмой «*Nintendo*» и выпущенная в 1981 году.

Даже сегодня, когда трехмерная графика и открытый мир – обычная игра, не вызывающая особого восторга, выходят достаточно интересные проекты в жанре платформер. Примерами таких игр послужат:

* Super Meat Boy - [хардкорный](https://cubiq.ru/hardkornye-igry/) платформер в ретро-стиле, рассказывающий историю Мясного парня, который спасает свою подругу. Super Meat Boy содержит более трех сотен уровней, каждый из которых бросит серьезный вызов рефлексам и нервам. Пройти основные главы игры непросто, еще сложнее преодолеть дополнительные «темные» уровни и открыть всех дополнительных персонажей. Но если вы пройдете Super Meat Boy, то больше ничего не будете бояться (кроме, разве что, следующего платформера от создателя этой игры – The End Is Nigh, а также грядущего сиквела SMB).
* Limbo - это монохромный двухмерный платформер про мальчика, который отправляется в таинственный мир на поиски своей пропавшей сестры. В Limdo приятный дизайн и захватывающий сюжет, который затягивает многих, несмотря на двухмерность.
* Inside - Игра от создателей Limbo, не менее мрачная и загадочная. Геймерам предстоит взять на себя управление маленьким мальчиком, который стремится вырваться из некоего тоталитарного мира. Герою предстоит преодолеть немало препятствий, столкнуться с различными опасностями, не раз рисковать жизнью, чтобы в итоге прийти к неожиданному и даже шокирующему финалу. Секретная концовка проекта еще больше запутывает игрока, что, в сочетании с обволакивающей атмосферой, стильной картинкой, затягивающим геймплеем и необычным сеттингом возводит его в топ лучших представителей жанра платформер.
* Cuphead – это в прямом смысле слова "классический" платформер. Классический, потому что все в нем выдержано в духе 1930-х: от графики, кажется, вышедшей из-под пера самого Уолта Диснея, до акварельных фонов и джазового музыкального сопровождения.
* Unravel - это милый и трогательный "тряпичный" платформер, в котором игроки управляют куклой из ниток и проволоки, перемещая ее по уровням, используя собственное "распущенное" тело.
* Little Nightmares - Еще один жуткий платформер с ребенком в главной роли – на этот раз маленькой девочкой по имени Шестая, которая пытается сбежать с мрачного подводного [корабля](https://cubiq.ru/morskoj-boj-luchshie-igry-pro-korabli-na-pk/), наполненного уродливыми голодными существами. Все, что есть у героини – желтый дождевик, зажигалка и неистребимая воля к жизни, которая заставляет ее идти вперед, преодолевать препятствия, бросать вызов чудовищам.
* Mario – разработана и выпущена еще в 1985 году и это вероятно самый популярный платформер в мире, например, было продано около 300 миллионов картриджей игры *Super Mario Bros* по всему миру.

В итоге, можно сделать вывод, что платформеры, не важно 2D или 3D, все еще актуальны и очень популярны.

Программа имеет возможность считывать карту из файла заголовка Levels.h, расположенного в папке OIRAM. Ознакомившись с условными обозначениями объектов на карте (их полный список указан в файле ниже), пользователь может самостоятельно дополнять и полностью перерабатывать ее по своему желанию. Конец игры наступает, как только главный герой добирается до конца карты, после чего он может либо перезапустить игру, нажав на клавишу Tab, либо выйти из приложения, нажав на клавишу Escape.

**Цель работы:**

* разработать игру «OIRAM»;
* получить опыт разработки конечно программного продукта;
* закрепить навыки проектирования и тестирования проекта.

Для реализации поставленной цели необходимо решить следующие задачи:

* + подключить графическую библиотеку SFML;
  + написать и внедрить логику игры;
  + найти / нарисовать текстуры для объектов;
  + реализовать функцию выхода и перезапуска.

# ГЛАВА 1

**АНАЛИЗ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

При выборе графической библиотеки C++ я остановился на SFML.

SFML (Simple and Fast Multimedia Library) является самой удобной и быстрой графической библиотекой для C++, по моему мнению. Её самое главное преимущество — минимальные требования к уровню знаний языка и лёгкость освоения. Второе преимущество — кроссплатформенность: SFML работает под Windows, Linux (только X11, но не Wayland) и Mac OS.

SFML библиотека используется по большей части небольшими стартапами и программистами, для которых создание игр — хобби. SFML популярен среди небольших команд благодаря тому, что разработка графической части программы не требует написания больших объёмов кода.

Также, во время анализа была рассмотрена библиотека Qt, которая в свою очередь, обогнала SFML по кроссплатформенности, так как доступна на мобильных платформах, но уступает по скорости – чего нам ни в коем случае допускать нельзя.

# ГЛАВА 2

**ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ**

## Постановка задачи

Необходимо написать программу, в которой пользователь имеет возможность управлять игровым персонажем, с целью пройти уровень до конца и набрать наибольшее количество монет.

Программа включает в себя:

* создание и отрисовку персонажа, врагов, монет, земли, заднего фона
* возможность управления персонажем по заданному уровню (идти вправо, идти влево, прыжок)
* уничтожение врагов
* создание панели меню для перезапуска уровня, либо выхода из игры

В игре существуют следующие объекты:

Таблица 1. Игровые объекты.

|  |  |
| --- | --- |
| Тип объекта | Вид |
| Персонаж - призрак |  |
| Враг - сантипеда |  |
| Земля |  |
| Грунт |  |
| Платформа |  |
| Фонарь |  |
| Алмаз |  |
| Шипы |  |

## 2.1 Описание алгоритмов

Для решения поставленной задачи воспользуемся алгоритмом, представленным на рисунке 2.1.

Он представляет собой основной цикл программы. В нем считываются нажатия клавиш для управления персонажем, на экран выводится карта и другие объекты, проверяются столкновения с врагами, вызываются функции обновления состояния врагов, персонажа.



Рисунок 2.1. Блок-схема основного цикла.

Управление происходит с помощью клавиатуры, на выбор можно использовать ‘W A D’, либо ‘← → ↑’.

А или ← − при нажатии персонаж идет влево. D или → − при нажатии персонаж идет вправо. W или ↑ - при нажатии персонаж подпрыгивает.

Находясь в воздухе, главный герой может смещаться вправо и влево.

## 2.2 Описание структур данных

1. Класс CHARACTER является родителем класса HERO, изображен на рисунке 2.2. Отвечает за текстуры, некоторые общие переменные, которые, в случае дальнейших обновлений игры, можно будет применять не только к главному герою, но и к другим сущностям.

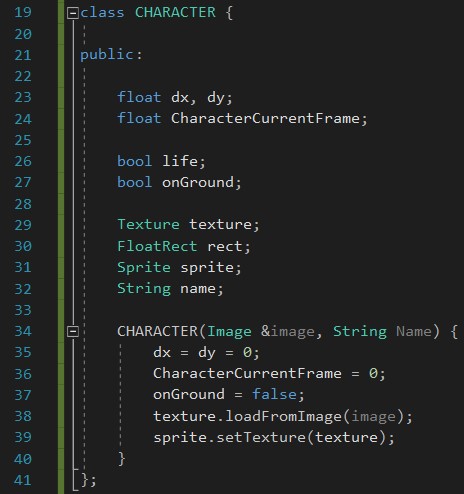


Рисунок 2.2. Класс CHARACTER

Класс HERO изображен на рисунках 2.3.1., 2.3.2, 2.3.3 Содержит в себе метод Update, который отвечает за изменение положения героя в пространстве, анимацию и проверку столкновений.

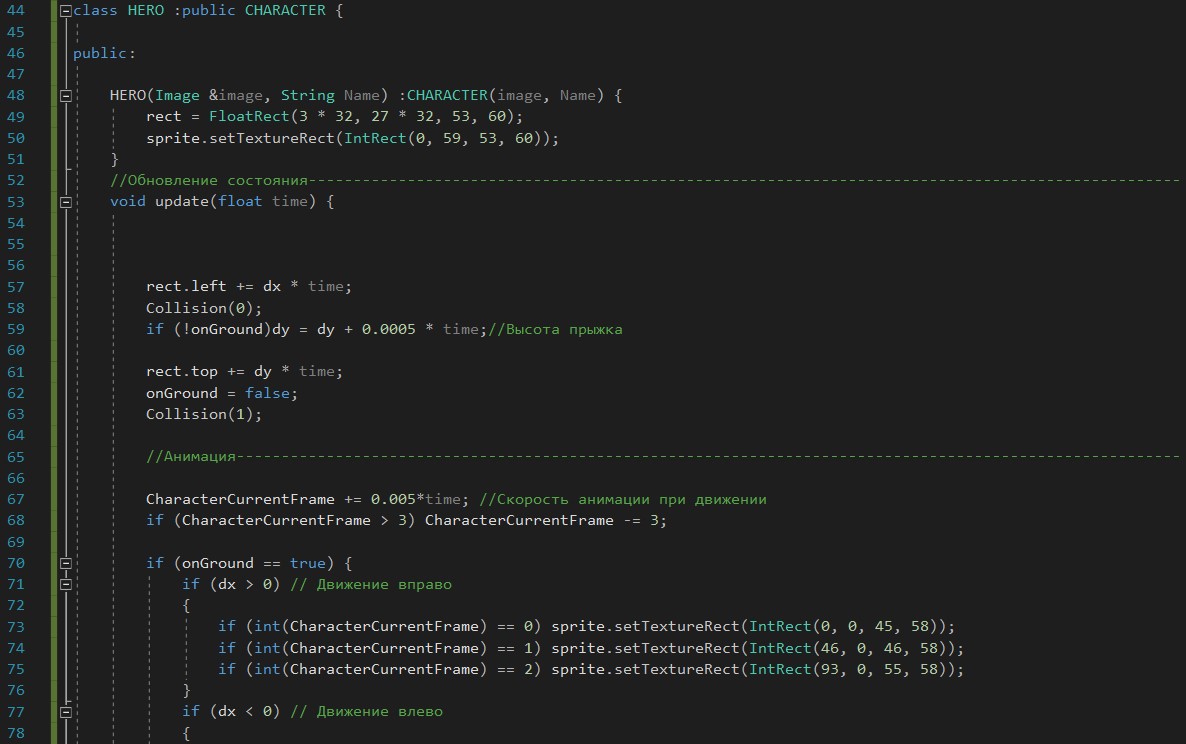


Рисунок 2.3.1. Класс HERO

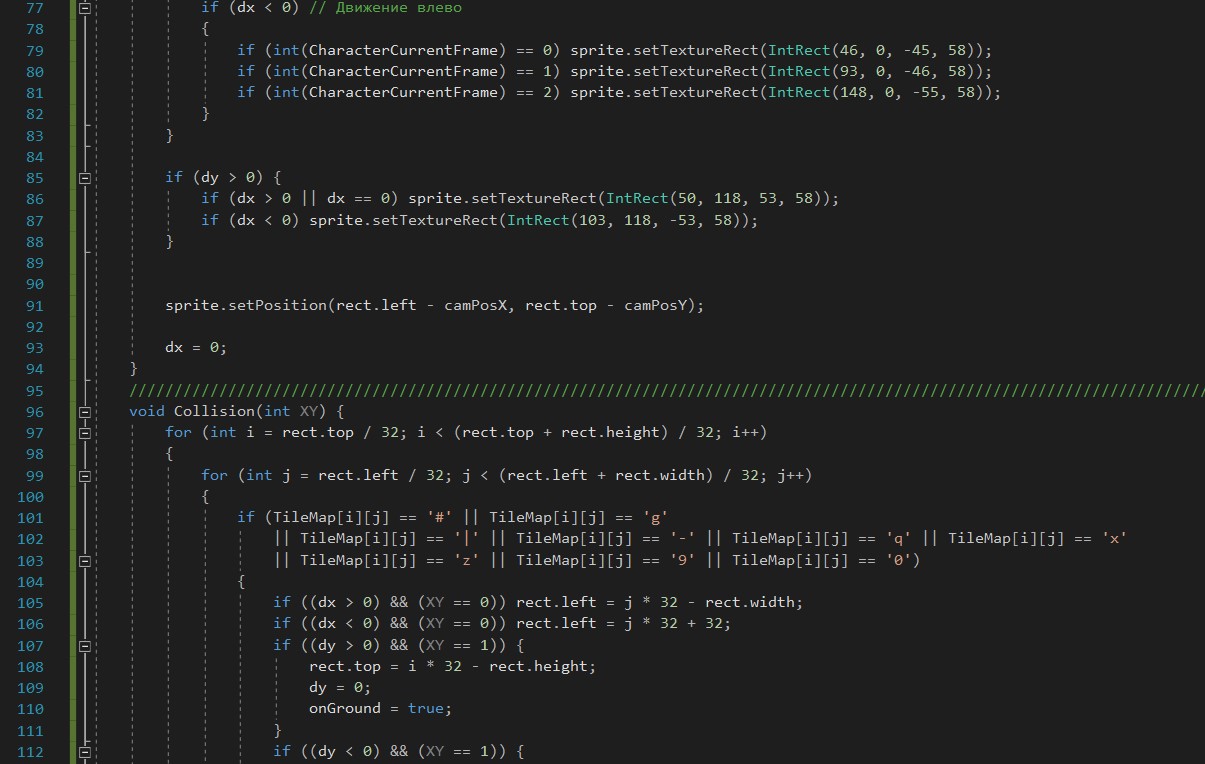


Рисунок 2.3.2. Класс HERO

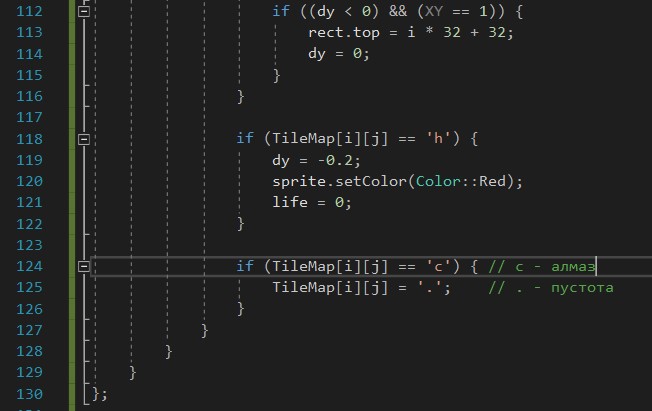


Рисунок 2.3.3. Класс HERO

1. Класс ENEMY. Данный класс отвечает за врагов и представлен на рисунке 2.4.1, 2.4.2 Имеет методы Set, Update и Collision. Метод Collision в классе HERO отличается от данного, враги не могут взаимодействовать с некоторыми объектами, например, алмазы, а также отключена проверка столкновений по оси Y, так как враги ходят по плоскости.

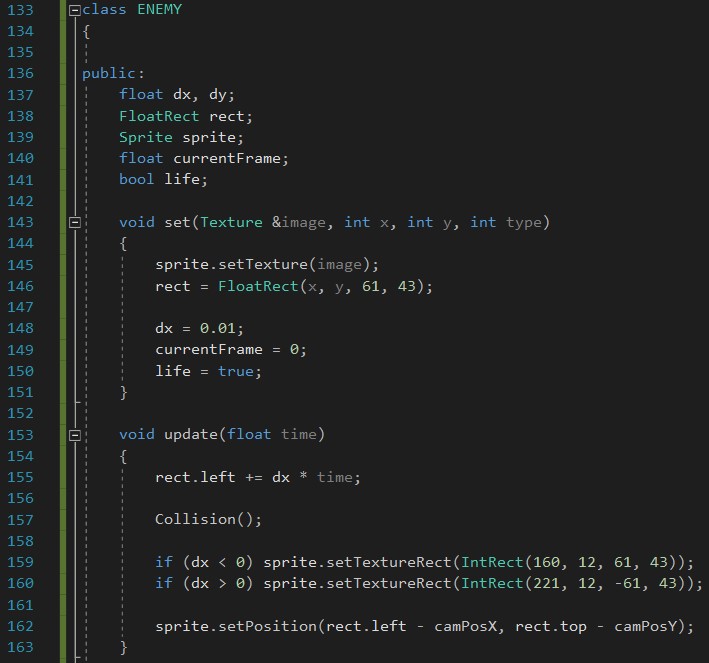


Рисунок 2.4.1. Класс ENEMY

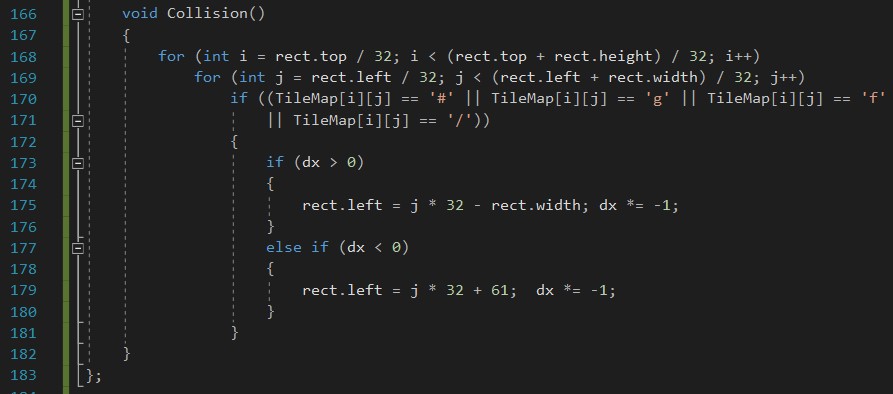


Рисунок 2.4.2. Класс ENEMY

## 2.3 Описание основных классов

* + 1. **Описание методов класса CHARACTER**

При инициализации объекта класса CHARACTER в конструкторе его свойства задаются начальными значениями, как показано на рисунке 2.5.

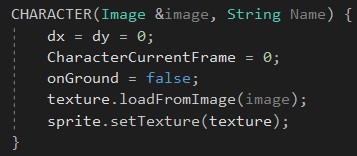


Рисунок 2.5. Конструктор класса CHARACTER

* + 1. **Описание методов класса HERO**

При инициализации объекта класса HERO в конструкторе его свойства задаются начальными значениями, как показано на рисунке 2.6.

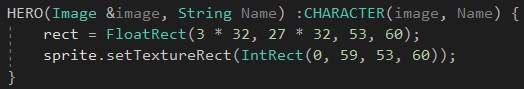


Рисунок 2.6. Конструктор класса HERO

Метод Update класса HERO показан на рисунке 2.7. Он вызывается перед отрисовкой каждого кадра (60 раз в секунду) на экран, принимает time и отвечает за обновления состояния главного героя:

* + - изменяется положение по осям X и Y
    - меняются текстуры главного героя (анимация)
    - изменяется положение камеры

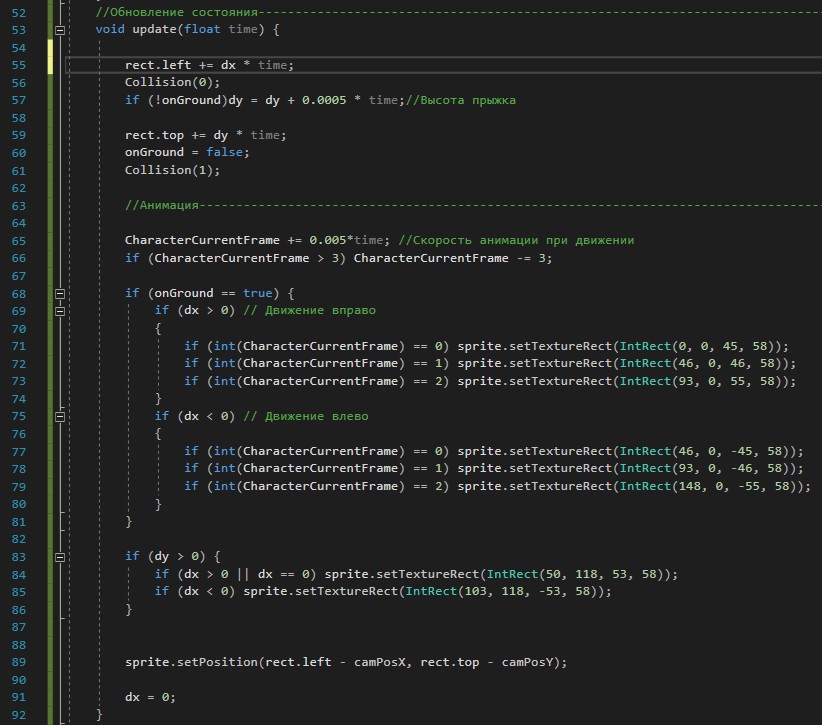


Рисунок 2.7. Метод Update класса HERO

Метод Collision класса HERO. Показан на рисунке 2.8. В нем реализуется проверка столкновений главного героя с окружающим миром, т.е. всеми объектами, находящимися ка игровой карте.

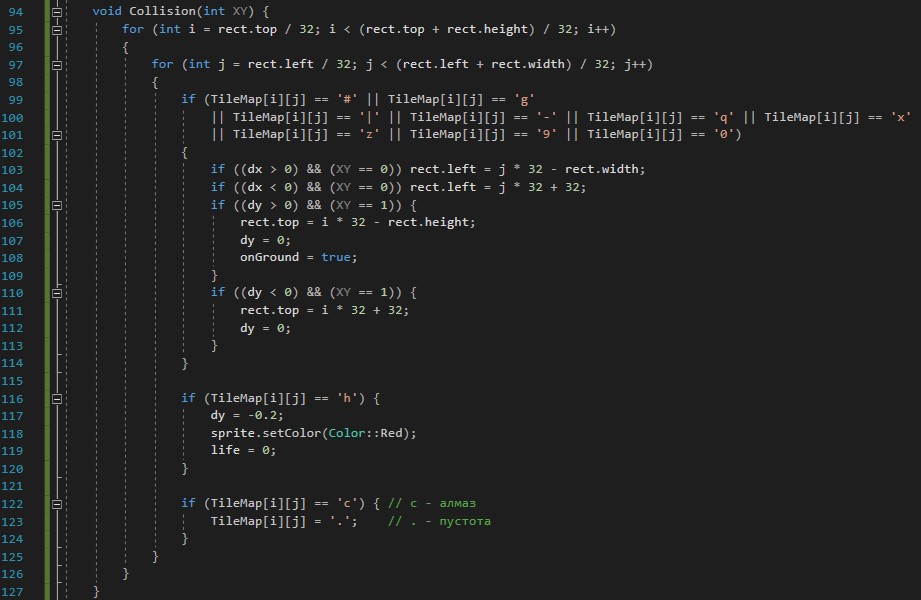


Рисунок 2.8. Метод Collision класса HERO

* + 1. **Описание методов класса ENEMY**

В методе Set класса ENEMY объекту задаются первоначальные значения, которые были переданы в него после создания. Реализация этого метода показана на рисунке 2.9.

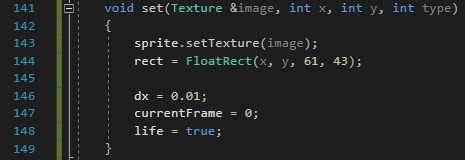


Рисунок 2.9. Метод Set класса ENEMY

Метод Update класса ENEMY. Вызывается, так же как и метод Update класса HERO, каждый раз перед отрисовкой объектов. Он отвечает за:

* + - * смещение спрайта по оси X
      * вызов метода Collision класса ENEMY
      * изменение спрайта, в зависимости от направления движения
      * •положение спрайта относительно камеры его реализация показана на рисунке 2.10.

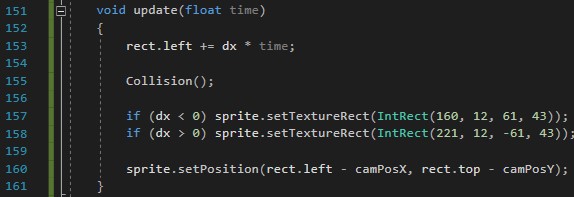


Рисунок 2.10. Метод Update класса ENEMY

Метод Collision класса ENEMY. Данный метод отличается от метода Collision класса HERO, в нем убраны обработки взаимодействий с некоторыми объектами на игровой карте, например, алмазы (враги моги бы их собирать). Так же, в угоду оптимизации, отсутствует обработка столкновений по оси Y. Реализация метода Collision класса ENEMY приведена ниже на рисунке 2.11.

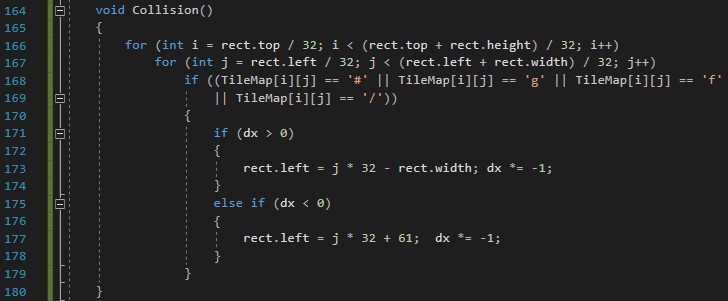


Рисунок 2.11. Метод Collision класса ENEMY

# ГЛАВА 3

**ТЕСТИРОВАНИЕ ПРОГРАММЫ**

Тестирование приложения, в большей части происходило прямо по ходу написания кода, добавления новых возможностей и функций. Было произведено огромное количество проверок и перезапусков игры, в ходе чего, было исправлено большое количество багов и неисправностей.

При заключительном тестировании были проверены следующие пункты:

1. Управление персонажем
   1. Движение вправо
   2. Движение влево
   3. Прыжок
2. Место появления персонажа
3. Коллизия
   1. Столкновения с блоками
   2. Столкновения с врагами
   3. Столкновения с монетами

Итог:

* + - На данный момент, явных багов не обнаружено
    - Утечки памяти, усечение из double в float
    - Присутствуют недочеты в плане геймплея

# ГЛАВА 4

**ВИЗУАЛЬНАЯ СОСТАВЛЯЮЩАЯ ИГРЫ**

1. На рисунке 4.1 показана позиция на которой появляется персонаж в начале игры.



Рисунок 4.1. Стартовая позиция

1. На рисунке 4.2 изображен враг, для его убийства нужно прыгнуть на него.



Рисунок 4.2. Враг

1. На рисунке 4.3 изображён алмаз, который нужно собрать.



Рисунок 4.3. Алмаз

1. На рисунке 4.4 изображены шипы, если на них наступить игра начнётся заново.



Рисунок 4.4. Шипы

# ЗАКЛЮЧЕНИЕ

В результате работы была создана игра на языке программирования С++ с использованием графической библиотеки SFML. Получены хорошие практические знания в построении алгоритмов, работой с массивами, функциями и циклами. В итоге создана игра в жанре «Приключение», но до надлежащего вида ей еще далеко, и она нуждается в дальнейшей доработке и улучшениях.

# ЛИТЕРАТУРА

1. Герберт Шилдт. С++ для начинающих. Шаг за шагом.
2. Роберт Лафоре. Объектно-ориентированное программирование в С++.
3. Страуструп. Программирование. Принципы и практика с использованием C++.
4. Раймондас Пупиус. SFML Game Development By Example.
5. Артур Морейра, Хенрик Вогелиус Ханссон, и Ян Халлер. SFML Game Development.
6. [Milcho G. Milchev](https://www.google.com/search?sa=X&rlz=1C1SQJL_ru__914__914&sxsrf=AOaemvKDp3Hgu_KU-ELVtjeFNvI7yIl1gw:1635140093966&q=sfml+essentials+milcho+g.+milchev&stick=H4sIAAAAAAAAAOPgE-LVT9c3NEwqTysrqzDPVoJxU4xMcg1ytGSyk630k_Lzs_XLizJLSlLz4svzi7KtEktLMvKLFrEqFqfl5iikFhen5pVkJuYUK-Rm5iRn5Cuk60FYqWU7WBkByNJwY2UAAAA&ved=2ahUKEwjxqdXD6-TzAhVSAxAIHfF7AxIQmxMoAXoECDAQAw). SFML Essentials.
7. SFML Blueprints.