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 “NEMARIO”.

Курсовая работа

по дисциплине «Программирование» студента 1 курса группы ПИ-203

Заговенков Руслан Андреевич

направления подготовки 09.03.04 «Программная Инженерия»

|  |  |  |
| --- | --- | --- |
| Научный руководитель  ассистент кафедры компьютерной инженерии и моделирования | (оценка)    (подпись, дата) | Чабанов В.В |

Симферополь, 2021

# РЕФЕРАТ

**Предмет исследования** – инструменты разработки и программирование компьютерной игры в жанре “Приключение”.

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

# Цель работы:

* освоить синтаксис языка программирования C++;
* изучить и реализовать на практике алгоритмы;
* применить на практике знания, полученные за семестр.

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.

# ОГЛАВЛЕНИЕ

[Реферат 2](#_TOC_250007)

[Введение 4](#_TOC_250006)

Глава 1. Анализ инструментальных средств для решения поставленной задачи 5

Глава 2. Программная реализация приложения 8

* 1. [Постановка задачи 8](#_TOC_250005)
  2. [Описание алгоритмов 8](#_TOC_250004)
  3. [Описание структур данных 10](#_TOC_250003)
  4. [Описание основных функций 14](#_TOC_250002)

Глава 3. Тестирование программы 19

Глава 4 Визуальная составляющая игры 20

[Заключение 22](#_TOC_250001)

[Литература 23](#_TOC_250000)

# ВВЕДЕНИЕ

Ни для кого не секрет, что видео игры прочно заняли свою позицию в современной индустрии развлечений. Существуют попытки выделить

компьютерные игры как отдельную область искусства, наряду с театром, кино и т.п. Разработка игр может оказаться не только увлекательным, но и прибыльным делом, примеров этому предостаточно в истории. Первые примитивные

компьютерные и видео игры были разработаны в 1950-х и 1960-х годах. Они работали на таких платформах, как осциллографы, университетские

мейнфреймы и компьютеры EDSAC. Самой первой компьютерной игрой стал

симулятор ракеты, созданный в 1942 году Томасом Голдсмитом Младшим (англ. Thomas T. Goldsmith Jr.) и Истл Рей Менном (англ. Estle Ray Mann). Позже, в 1952 году, появилась программа "OXO", имитирующая игру "крестики-нолики", созданная А.С. Дугласом как часть его докторской диссертации в Кембриджском Университете. Игра работала на большом университетском компьютере,

известном как EDSAC (Electronic Delay Storage Automatic Calculator). В

настоящее время, разработка игры - это многомиллионный процесс, в котором задействована целая команда разработчиков, сложные современные технологии и даже маркетинговые ходы.

# ГЛАВА 1

**АНАЛИЗ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

Для создания проекта использовалась программа Visual Studio 2019. Проект создавался методом проб и ошибок, с постоянными изменениями и улучшениями, как в плане производительности, так и 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 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:

* 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.
* 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.

* Mario – разработана и выпущена еще в 1985 годуи 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 важно2D или 3D, все еще актуальны и очень популярны.

Программа имеет возможность считывать карту из файла заголовка Levels.h, расположенного в папке GameV. 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.

В игре существуют следующие объекты:

Таблица 1. Игровые объекты

|  |  |
| --- | --- |
| Тип объекта | Вид |
| Персонаж |  |
| Враг |  |
| Земля |  |
| Грунт |  |

|  |  |
| --- | --- |
| Платформа |  |
| Фонарь |  |
| Монета |  |

# 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

# 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.

# 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.

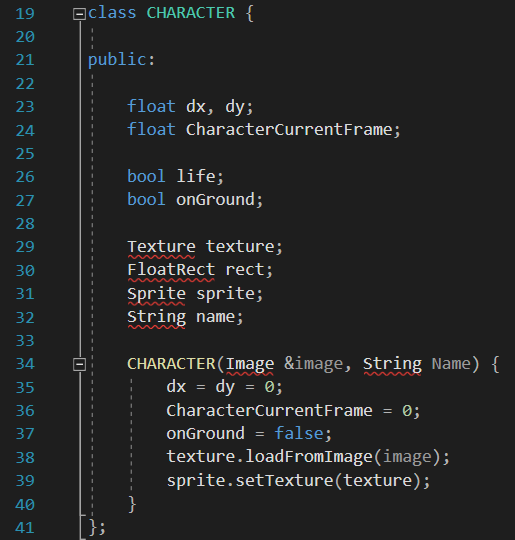


Рисунок 2.2. Класс CHARACTER

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

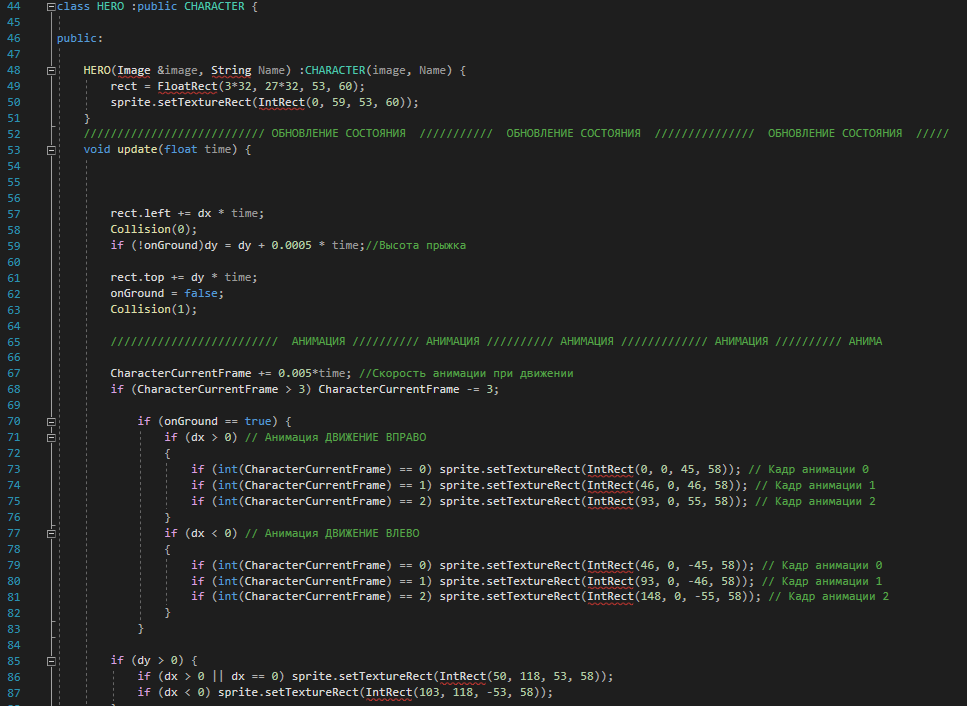


Рисунок 2.3.1. Класс HERO

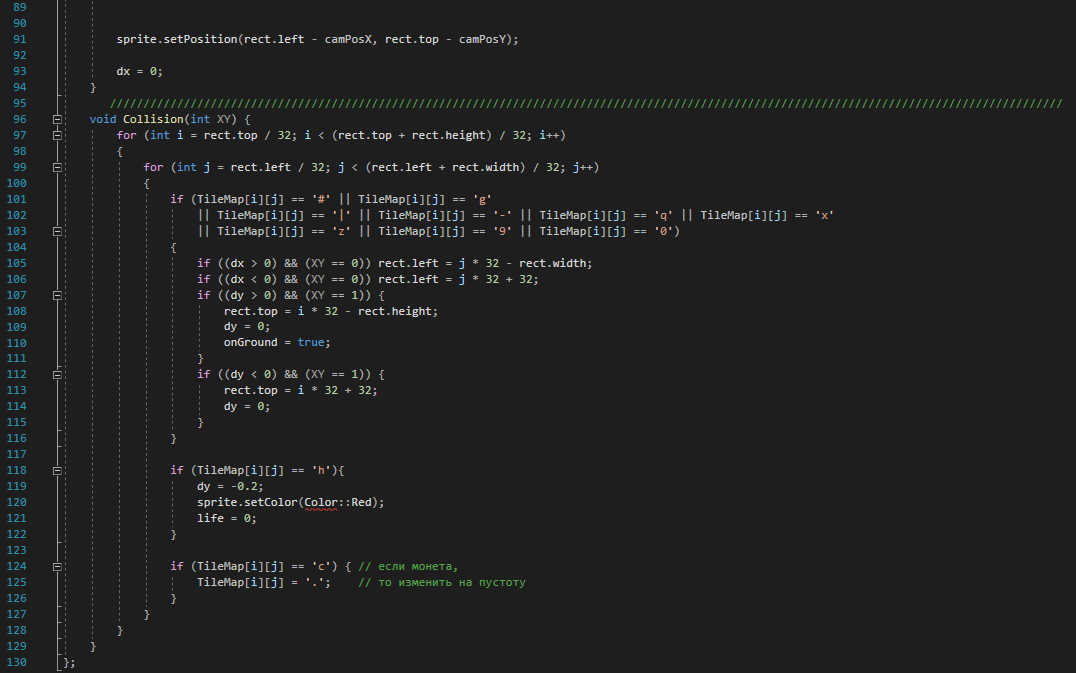
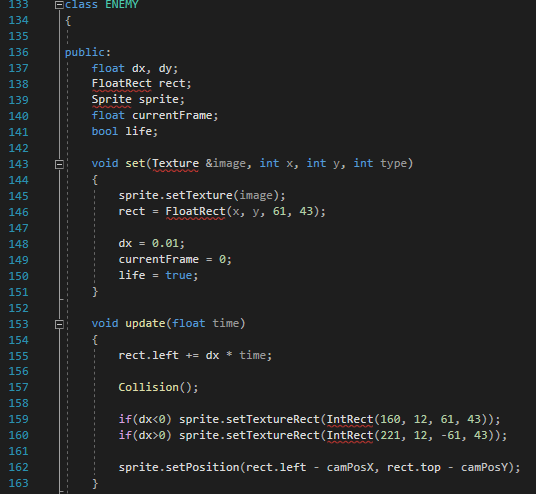


Рисунок 2.3.2. Класс HERO

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



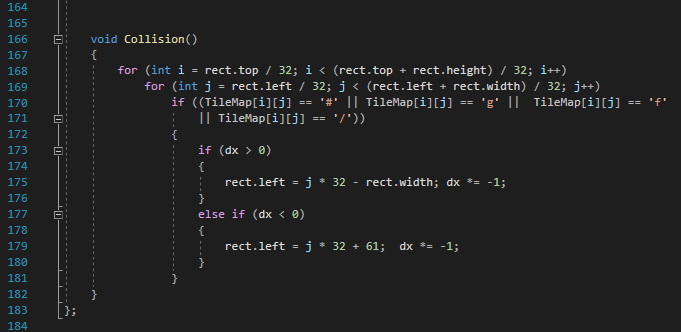


Рисунок 2.4. Класс ENEMY

# Описание основных функций

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

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

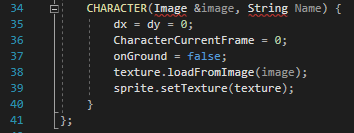


Рисунок 2.5. Конструктор класса CHARACTER

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

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

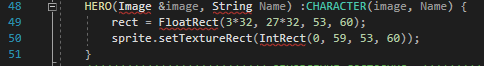


Рисунок 2.6. Конструктор класса HERO

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

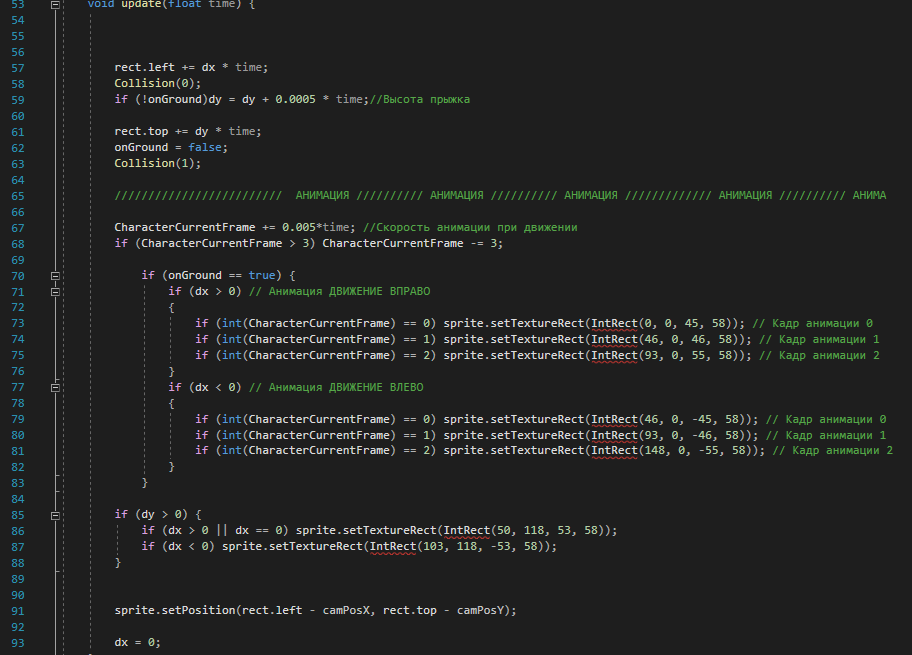


Рисунок 2.7. Метод Update класса HERO

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.

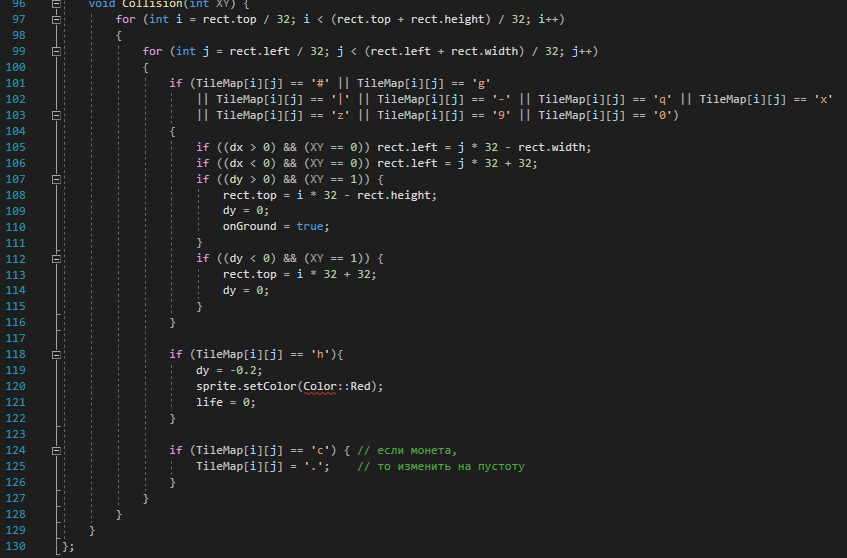


Рисунок 2.8. Метод Collision класса HERO

* + 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.

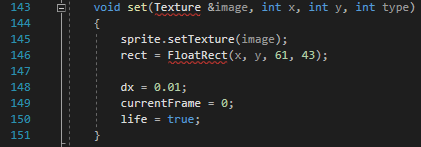


Рисунок 2.9. Метод Set класса ENEMY

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.

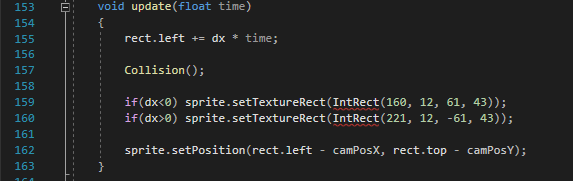


Рисунок 2.10. Метод Update класса ENEMY

Метод Collision класса ENEMY. Данный метод отличается от метода Collision класса HERO, в нем убраны обработки взаимодействий с некоторыми объектами на игровой карта, например монеты (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.

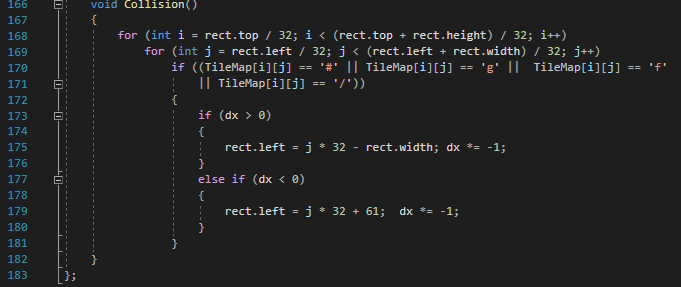


Рисунок 2.11. Метод Collision класса ENEMY

# 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. изображен стартовый экран игры, на который пользователь попадает при открытии программы.



Рисунок 4.1. Стартовое меню

1. На рисунке 4.2. показана позиция на которой появляется персонаж в начале игры.



Рисунок 4.2. Стартовая позиция

1. На рисунке 4.3 изображен враг, для его убийства нужно прыгнуть на него.



Рисунок 4.2. Враг

# 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 платформер, приключения, но до надлежащего вида ей еще далеко и она нуждается в дальнейшей доработке и улучшении.

# ЛИТЕРАТУРА

* 1. Frank B. Brokken “C++ Annotations”
  2. Доусон Майкл “Изучаем C++ через программирование игр”
  3. Richard L. Halterman “Fundamentals of Programming C++”