МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем и цифровых технологий

Работа допущена к защите

_____Руководитель

«21» мая 2021 г.

КУРСОВОЙ ПРОЕКТ

по дисциплине «Проектная деятельность»

на тему: «Разработка игры в жанре "Платформер"»

Студент _____ Бессонов М.П.

Шифр 191009

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 92-ПГ

Руководитель _____ Лукьянов П.В.

Оценка: «отлично» Дата 21.05.2021г.

Орел 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем и цифровых технологий

УТВЕРЖДАЮ:
_____ Зав. кафедрой
«12» марта 2021 г.

ЗАДАНИЕ
на курсовой проект

по дисциплине «Проектная деятельность»

Студент Бессонов М.П. Шифр 191009

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 92-ПГ

1 Тема курсового проекта

«Разработка игры в жанре "Платформер"»

2 Срок сдачи студентом законченной работы «21» мая 2021

3 Исходные данные

Разработать программный проект по теме «Разработка игры в жанре "Платформер"». Проанализировать и описать предметную область, описать аналоги, разработать требования к проекту. Произвести моделирование предметной области и описать спецификации. Произвести проектирование архитектуры и структуры программного проекта, разработать основные алгоритмы, спроектировать интерфейс. Реализовать и протестировать ПО.

4 Содержание курсового проекта

Анализ задач разработки игры в жанре "Платформер"

Разработка спецификаций программного обеспечения

Проектирование программного обеспечения

Реализация программного обеспечения

5 Отчетный материал курсового проекта

Пояснительная записка курсового проекта; презентация

Руководитель _____ Лукьянов П.В.

Задание принял к исполнению: «12» марта 2021

Подпись студента _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный мир предъявляет высокие требования к выпускнику вуза. Одной из важнейших является компетенция техническая. Современный специалист в области IT должен уметь работать в разных программах и с разными ресурсами. Кроме того, он должен уметь разрабатывать собственные программные приложения.

Одним из видов таких приложений является игровое приложение. Разработка таких приложений является актуальной, так как они помогают эффективно провести досуг, снять напряжение и даже расширить свой кругозор. Кроме того, игры могут повышать способность человека к восприятию и тем самым улучшать принимаемые решения. Одним из видов игровых приложений является "Платформер". Приложения такого типа могут помочь в развитии моторики и цветовосприятия, так как потребуют от человека замечать изменения на игровом экране и правильно на них реагировать. Также такие приложения позволяют повысить скорость реакции.

Объектом нашей работы является создание игры. Предметом нашей работы является процесс создания игры типа "Платформер".

Цель нашей работы: приобрести навыки разработки программных приложений на основе создания игры в жанре "Платформер".

Для достижения цели нам необходимо решить соответствующие задачи:

1) анализ предметной области;
2) разработка спецификаций;
3) рабочее проектирование;
4) реализация приложения на языке Actionscript 3.0;
5) тестирование программы.

1 АНАЛИЗ ЗАДАЧ РАЗРАБОТКИ ИГРЫ В ЖАНРЕ "ПЛАТФОРМЕР"

1.1 Обзор предметной области

Платформенные игры (часто упрощенные, как платформер или прыгай и беги) - это жанр видеоигр и поджанр экшн-игр, основной целью которых является перемещение персонажа игрока между точками в визуализированной среде [4].

"Платформер" как игровой жанр появился в 1980-х годах и стал стремительно набирать популярность из-за относительно простого и понятного геймплея. В эти игры обычно играют либо сбоку с использованием 2D движения, либо в 3D с камерой от первого или от третьего лица.

Основной чертой игрового процесса является перемещение игрового персонажа по платформам и сбор предметов (обычно необходимых для завершения уровня). Многие платформеры также имеют бездонные ямы или другие особенности игрового мира, которые немедленно убивают персонажа, если он упадет в них. Помимо этих элементов, компоненты платформенной игры могут сильно различаться, включая различные дополнительные параметры движения, сражения и другие функции [4].

Игры подобного жанра характеризуются нереалистичностью, рисованной мультяшной графикой. Героями таких игр обычно бывают мифические существа (к примеру, драконы, гоблины) или антропоморфные животные. Также героями могут являться нарисованные человечки ("стикмены") или какие-либо абстрактные персонажи, представленные, например, при помощи геометрических фигур.

Уровни в этом жанре, как правило, изобилуют секретами (скрытые проходы в стенах, высокие или труднодоступные места), нахождение которых существенно облегчает прохождение и подогревает интерес игрока.

В какой-то момент платформеры были самым популярным жанром видеоигр. На пике своей популярности, по оценкам, от четверти до одной трети консольных игр были платформерами, но с тех пор их вытеснили шутеры от первого лица. С 2010 года множество запущенных платформеров для мобильных устройств вернули этому жанру популярность [4].

В настоящей работе необходимо реализовать собственную игру типа "Платформер".

Давая краткую характеристику нашего приложения, можно сказать следующее: пользователи, используя игровое приложение, будут осуществлять управление шариком, который будет перемещаться по разным уровням карты, решать различные головоломки и собирать монеты для получения очков.

1.2 Обзор аналогов

Рассмотрим аналоги, на которые будем опираться при разработке нашего приложения (Таблица 1).

Прямым аналогом, который является прототипом нашего приложения, можно назвать серию платформенных игр Red Ball [9]. Как и в нашей игре, в нем можно выбирать уровень для прохождения. Начиная со второй части серии, в игру добавили секреты, которые игрок сможет найти при взаимодействии с окружающим миром, а с четвертой части появилась возможность получать достижения за выполнение определенных действий (например, при прохождении уровня за определенное время, за нахождение уникального секрета или за особую победу над противником). Но, в отличие от нашей игры, с самой первой части Red Ball реализовано плавное взаимодействие с окружающим миром (например, с ящиками или катящимися шарами), плавное торможение при замедлении скорости и скатывание тел с наклонной поверхности (скорость скатывания зависит от угла и типа поверхности). Также в третьей части игры появилась

возможность сражаться с враждебными персонажами, чего не реализовано в нашей игре.

Косвенными аналогами можно назвать любую игру в жанре "Платформер", например, "Vex" [10]. В этой игре главным персонажем является стикмен, который передвигается по уровням при помощи акробатических трюков (прыжки от стен, скольжения, прыжки по канатам и балкам). Соответственно физика в этой игре включает в себя плавные скольжения, плавное торможение на различных поверхностях и физику полета по дуге. При прохождении уровней за определенное время можно получать достижения, но при этом в игре нет возможности использовать магазин или выбирать персонажей и умения. Также в этой игре нет секретов, возможности сохранять свой прогресс и враждебных персонажей.

Таблица 1 – Обзор аналогов

	Игра "Платформер"	Red Ball	Vex
Возможность выбрать уровень	+	+	+
Возможность получать достижения	+	+ (с четвертой части)	+
Реалистичная физика	-	+	+
Возможность использовать магазин для покупки бонусов и персонажей	+	-	-
Возможность находить секреты	+	+ (со второй части)	-
Возможность сражаться с враждебными персонажами	-	+ (с третьей части)	-
Возможность сохранять и загружать прогресс	+	-	-

1.3 Разработка требований к программе

Разработаем требования, в соответствии с которыми мы будем реализовывать нашу программу. В нашей работе мы рассмотрим функциональные требования (определяют функциональность (поведение) программной системы) и нефункциональные требования, которые система должна демонстрировать [5].

К функциональным требованиям нашей программы можно отнести следующие:

- изменение игрового поля при действиях пользователя;
- отображение игрового поля по соответствующему действию пользователя;
- отображение текущей информации об игре и всех объектах в ней в виде графического игрового поля;
- отображение сообщений об окончании уровня или о получении достижения;
- изменение состояния совершенных покупок при соответствующем выборе игрока;
- сохранение текущего прогресса в файл при соответствующем выборе пользователя;
- загрузка выбранного пользователем файла с прогрессом.

К группе нефункциональных требований можно отнести следующие:

1) Требование к надежности: программа должна иметь защиту от некорректных действий пользователей и ошибочных исходных данных. Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

2) Требования к составу и параметрам технических средств:

- Тип системы: 64-разрядная или 32-разрядная операционная

система;

- Процессор: Intel или AMD (от 1,5 ГГц);
- Оперативная память: 2 ГБ и более;
- Операционная система: Windows 10 / 7 / 8 / XP / Vista;
- Видеокарта: Nvidia GeForce или AMD Radeon (от 512 Мб);
- Жесткий диск: 500 Мб и более.

3) Требования к масштабируемости: приложение должно иметь возможность добавления отдельных частей к уже существующим компонентам программы.

2 РАЗРАБОТКА СПЕЦИФИКАЦИЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Функциональная модель

Создадим и опишем функциональную модель нашего приложения.

Пользователь, войдя в приложение, сможет управлять персонажем. Для этого ему нужно выбрать уровень, а также бонусы, которыми он может пользоваться во время прохождения, и персонажа, за которого хочет играть. По ходу игрового процесса игрок будет получать достижения, список которых он может посмотреть из главного меню во время работы с приложением. Также пользователь может тратить полученные им монетки для приобретения товаров в магазине. В нем он может купить новых персонажей или новые бонусы. Кроме того, если пользователь достиг определенного прогресса и не хочет его потерять, он может сохранить его в файл и при новом запуске загрузить свой прошлый прогресс.

Диаграмма вариантов использования в нотации UML, которая определяет перечень аспектов поведения программного обеспечения в процессе взаимодействия с конкретным пользователем, представлена на рисунке 1 [3].

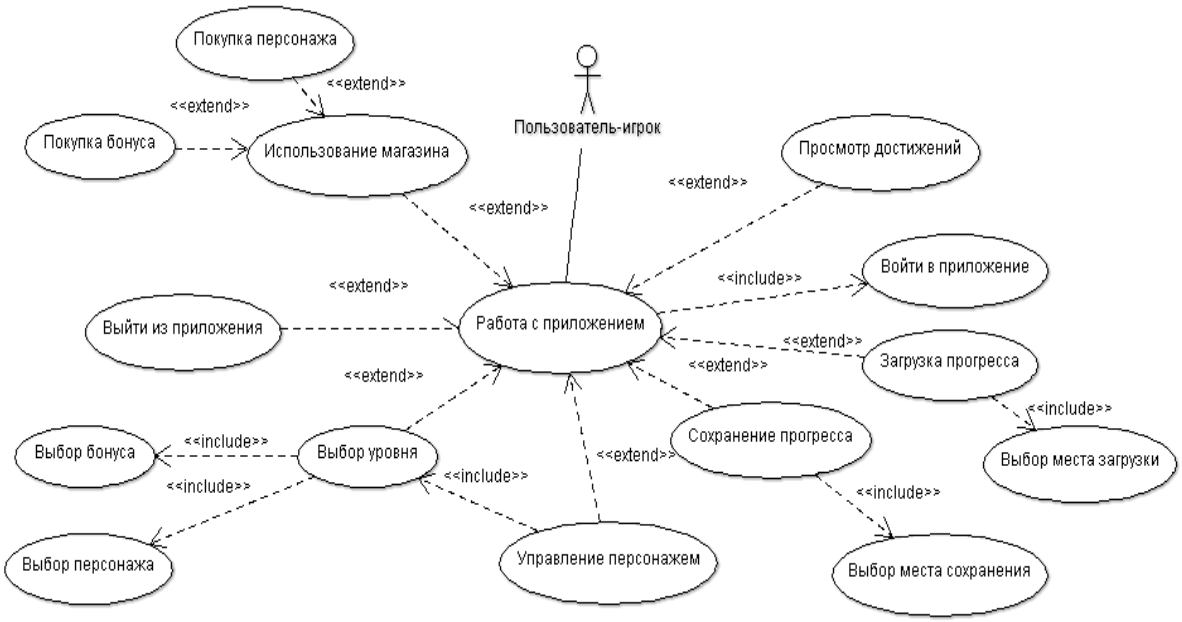

Рисунок 1 – Функциональная модель

2.2 Поведенческая модель

Создадим и опишем поведенческую модель нашего приложения, выделив конкретные состояния.

После осуществления входа приложение будет ожидать действие от пользователя. Если пользователь нажмет на кнопку сохранить приложение, перейдет в состояние сохранения прогресса, где в момент входа в состояние откроется окно с выбором места сохранения. После нажатия кнопки "ОК" или "Отмена" приложение вернется в состояние ожидания. Аналогичная ситуация с состоянием загрузки прогресса. Пользователь может посмотреть свои достижения в приложении, для этого он должен нажать на кнопку "Достижения", тогда приложение перейдет в состояние просмотра достижений и будет показывать на экране весь список до тех пор, пока пользователь не нажмет кнопку "В меню". Аналогичная ситуация с просмотром товаров в магазине. Чтобы начать играть, пользователь должен нажать на кнопку "Новая игра", тогда приложение перейдет в состояние выбора уровня, где будет показывать весь список возможных уровней игры. Для возврата в состояние ожидания пользователь должен нажать кнопку "В меню". Если игрок выбрал уровень, то начнется игровой процесс и приложение перейдет в состояние ожидание действия персонажа, сгенерировав при этом нужную карту. Игрок, нажимая клавиши движения и взаимодействия, может изменять игровой мир. Для выхода из этого состояния пользовать должен нажать на кнопку "Завершить игру", при этом ее параметры сохранятся.

Поведенческая диаграмма в нотации UML нашего приложения представлена на рисунке 2 [3].

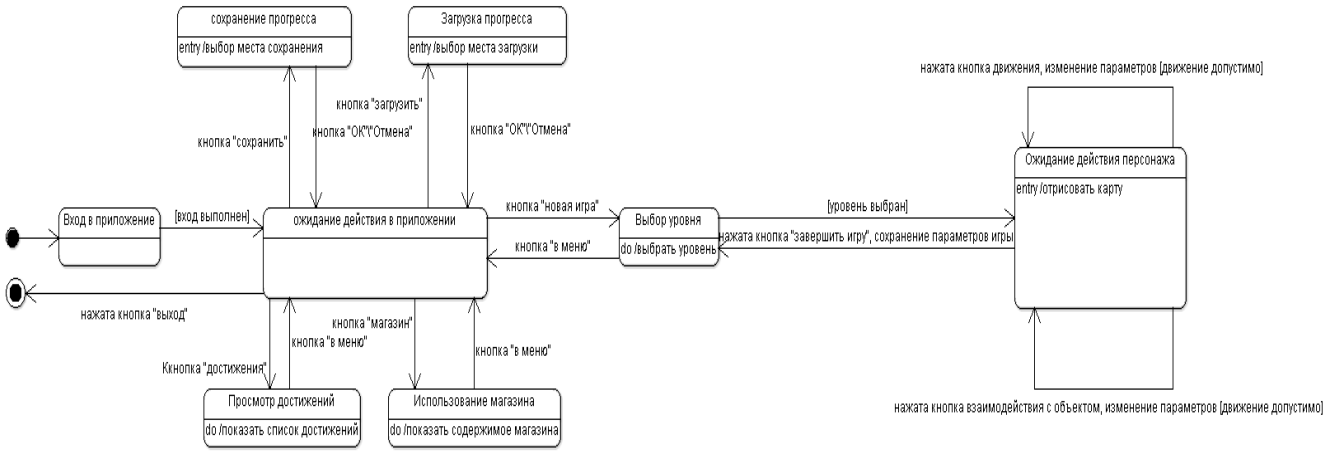

Рисунок 2 – Поведенческая модель

2.3 Концептуальная модель

Выделим сущности, которые в дальнейшем будем реализовывать в нашей игре.

Наша программа содержит в себе несколько сущностей, каждая из которых представляет часть игрового процесса. Среди выделенных нами сущностей есть следующие:

- Сущность "Земля", которая необходима для перемещения игрока по уровню;

- Сущность "Шипы", которые будут являться препятствием для игрока и при столкновении с ними игровой персонаж будет терять жизнь;

- Сущность "Флаг", которая является финальной точкой уровня и при столкновении с которым уровень будет завершен;

- Сущность "Монета", которую будет собирать игрок во время прохождения уровня и которую он сможет обменять в магазине на определенные товары;

- Сущность "ОбъектПрямогоВзаимодействия", которая состоит из объектов, с которыми непосредственно может взаимодействовать (двигать, нажимать, открывать и т.п.) игровой персонаж для прохождения уровней;

- Сущность "Жизнь" отображает количество оставшихся у игрока жизней;

- Сущность "Мяч", которая является нашим игровым персонажем и будет взаимодействовать с игровым миром;

- Сущность "Игровое поле", которая нужна для осуществления взаимодействия между игроком и объектами игрового мира. Она размещает в себе все перечисленные выше сущности.

Диаграмма классов в нотации UML нашей программы представлена на рисунке 3 [3].

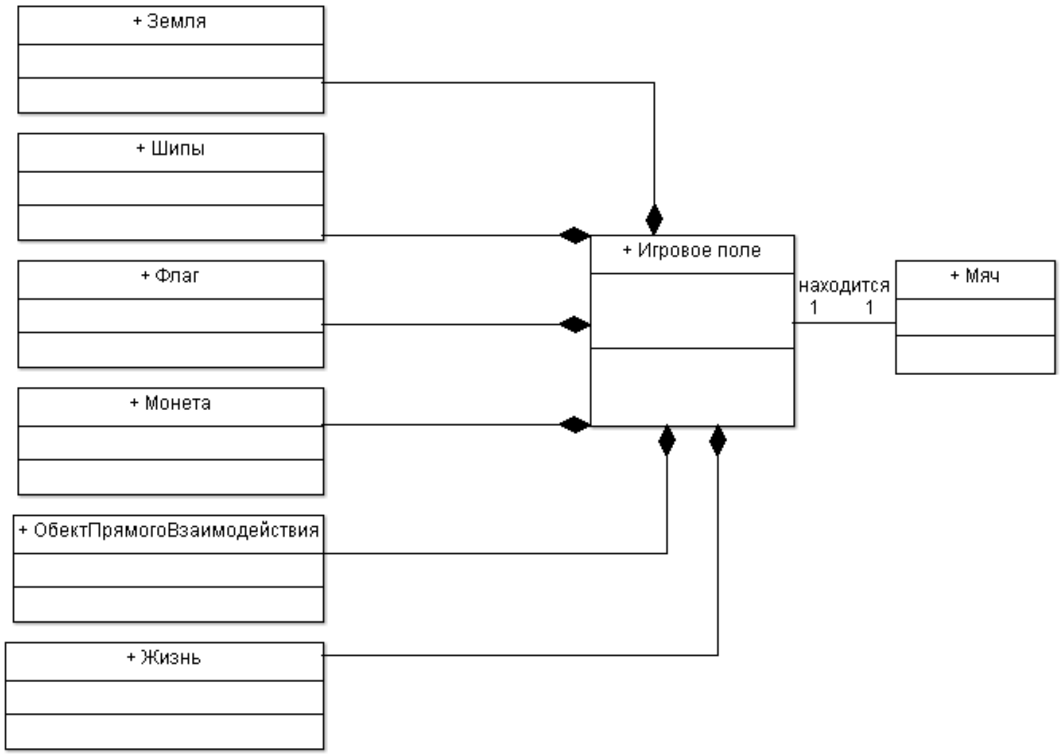

Рисунок 3 – Концептуальная модель

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Архитектура программного обеспечения

Наша программа соответствует модульной архитектуре [1]. Она состоит из трех модулей, которые имеют четко определенную функциональность и набор интерфейсов. На рисунке 4 представлена модульная схема программы.

Рисунок 4 – Схема модульной структуры программы

Модуль 1 представляет собой точку входа в программу. Он также хранит данные об игровом прогрессе, которые он может загрузить из источника данных и передать необходимую информацию при выборе уровня (подгруженных персонажей или бонусы) в Модуль 2. Кроме того, этот модуль описывает пользовательский интерфейс нашего приложения, который отвечает за формирование игрового меню и работу с ним. Этот модуль содержит функции для работы с достижениями, магазином, уровнями, бонусами, загрузкой и сохранением.

Модуль 2 описывает логику игрового процесса нашего приложения и включает в себя функции для работы с самой игрой: персонажем, игровым полем, игровыми объектами. Функциональность этого модуля запускается, когда пользователь в игровом меню нажимает на какой-либо уровень. В этот

момент запускается формирование уровня игры и загружаются выбранные бонусы, которые были сформированы в Модуле 1. Как только уровень игры будет сформирован, результат будет загружен в Модуль 1.

Модуль 3 представляет собой источник данных, который формирует игровой прогресс. Этот прогресс загружается в Модуль 1 и формирует достижения, товары и уровни - все то, чего достиг пользователь во время одного из своих предыдущих прохождений. Кроме того, при прохождении игры можно сохранять новые данные, которые затем можно использовать при следующем включении игры.

3.2 Структура программного обеспечения

Спроектируем наше приложение для его дальнейшей реализации.

Наша программа содержит в себе несколько классов, объекты которых представлены в игре. Все классы, представленные в игре, будут являться наследниками базового класса "Объект". Этот класс содержит в себе свойства и методы, необходимые для взаимодействия объектов и изменения их состояния. Из этого базового класса мы будем использовать методы для работы с игровыми объектами, которые мы будем реализовывать. Для реализации функционала наших игровых объектов будет достаточно тех методов, которые определены в базовом классе. Игровое поле будет при выборе уровня генерировать карту, размещая на ней необходимые объекты созданных классов, а затем при взаимодействии игрока с игровым миром будет изменять карту. В объекте класса "Мяч", которым будет управлять непосредственно игрок, будет использовать комбинации методов базового класса для взаимодействия с игровым миром. Диаграмма классов нашей программы в нотации UML представлена на рисунке 5 [3].

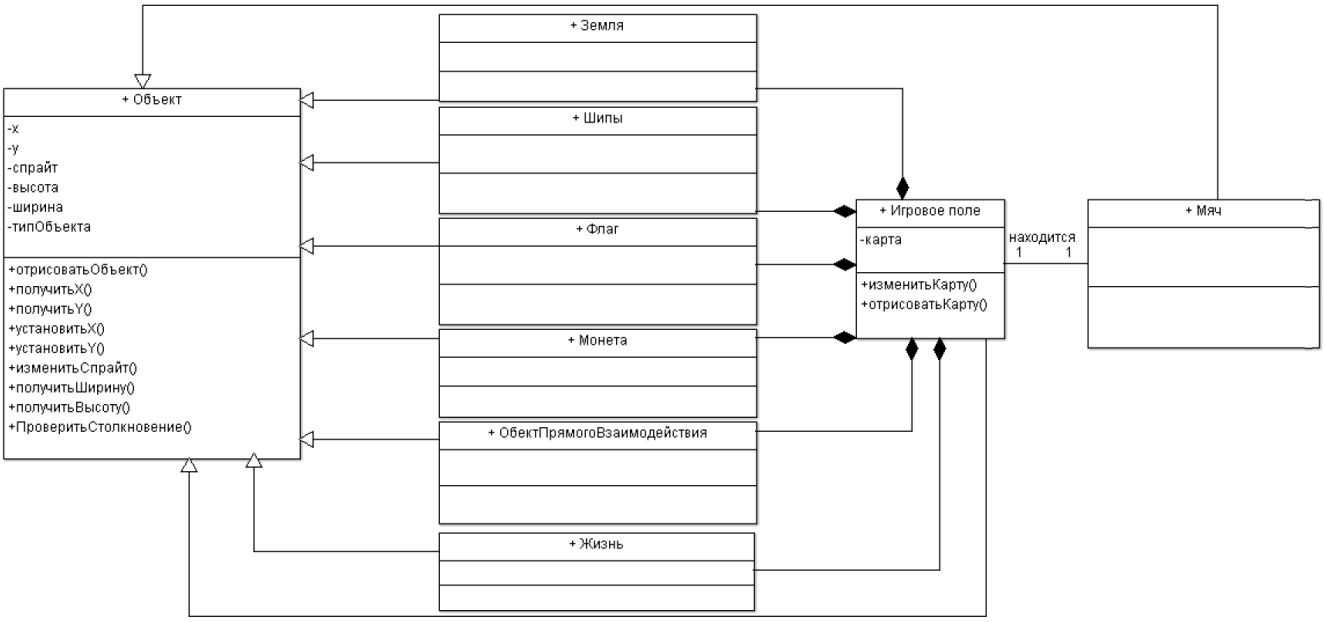

Рисунок 5 – Логическая модель

3.3 Описание алгоритмов

Опишем основной алгоритм, который используется в нашей программе - алгоритм работы уровня программы (Рисунок 6).

Этот алгоритм представляет собой два цикла с условием. Как только программа произведет начальную установку всех параметров, необходимых для количества собранных монет, скорости движения, высоты прыжка и других игровых характеристик, начнется исполнение подпрограммы, отвечающей за движение персонажа. Эта подпрограмма проверяет столкновение игрока с объектами игрового мира. Если игрок прикоснется к монетке, то увеличится счетчик, отвечающий за количество собранных монет. Если игрок умрет, то программа проверит, последняя ли это была жизнь у игрока. Если она последняя, то программа снова установит параметры и запустит подпрограмму движения персонажа. Если же у игрока больше не осталось жизней, программа выведет результат уровня, в котором будет зафиксировано поражение. Алгоритм будет работать, пока игрок не достигнет финального флага или пока у него не закончатся жизни, после чего запустится подпрограмма вывода результатов.

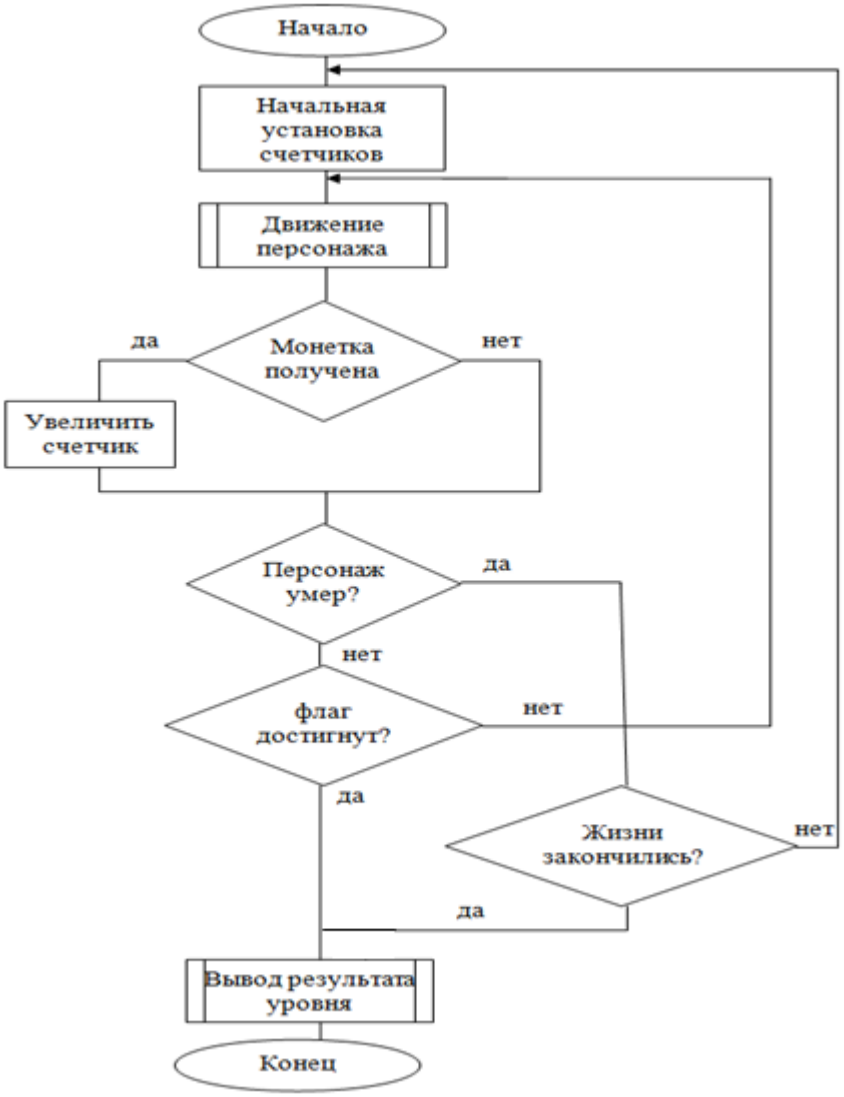

Рисунок 6 – Блок-схема уровня игры

3.4 Проектирование интерфейсов

Учитывая тот факт, что пользователь больше привык работать с графическим интерфейсом, так как это позволяет визуально оценить происходящее в программе и облегчает понимание и освоение программы неподготовленным пользователем, будем разрабатывать приложение с использованием графического интерфейса.

При запуске приложения пользователь должен увидеть перед собой главное меню, которое будет содержать несколько визуальных кнопок, при нажатии на которые будут отображаться части нашего приложения: магазин,

таблица достижений, карта уровней, а также экраны загрузки и сохранения прогресса.

В окне достижений пользователь сможет нажать на конкретное достижение, чтобы увидеть, что необходимо сделать для его получения.

В окне магазина пользователь сможет при нажатии на кнопку купить определенный товар или персонажа. После этого при нажатие определенных кнопок будет происходить установка/сброс выбранного бонуса. Если бонус установлен, он будет подгружаться вместе с персонажем в игру.

При нажатии на кнопку уровня на карте уровней будет запущен непосредственно процесс игры. При этом на экране будет отображаться сама игровая карта, персонаж, с которым возможно осуществлять взаимодействие при помощи клавиш управления клавиатуры, и все игровые объекты, необходимые для этого уровня. Для прохождения уровня игрок должен достичь красного флага, который в момент касания игрока с ним запустит анимацию. После чего на экран выведутся результат уровня, в котором будет указано, сколько монеток удалось собрать пользователю за этот уровень. Если же пользователь не смог пройти уровень и его персонаж погиб, то будет показано окно с текстовым сообщением о поражении.

Если пользователь при завершении уровня выполнил действие, которое разблокирует какое-либо достижение, то на экран выведется форма с сообщением, в котором будет указано название полученного достижения.

Карта диалоговых окон, которая отображает взаимосвязь между всеми окнами в приложении, представлена на рисунке 7.

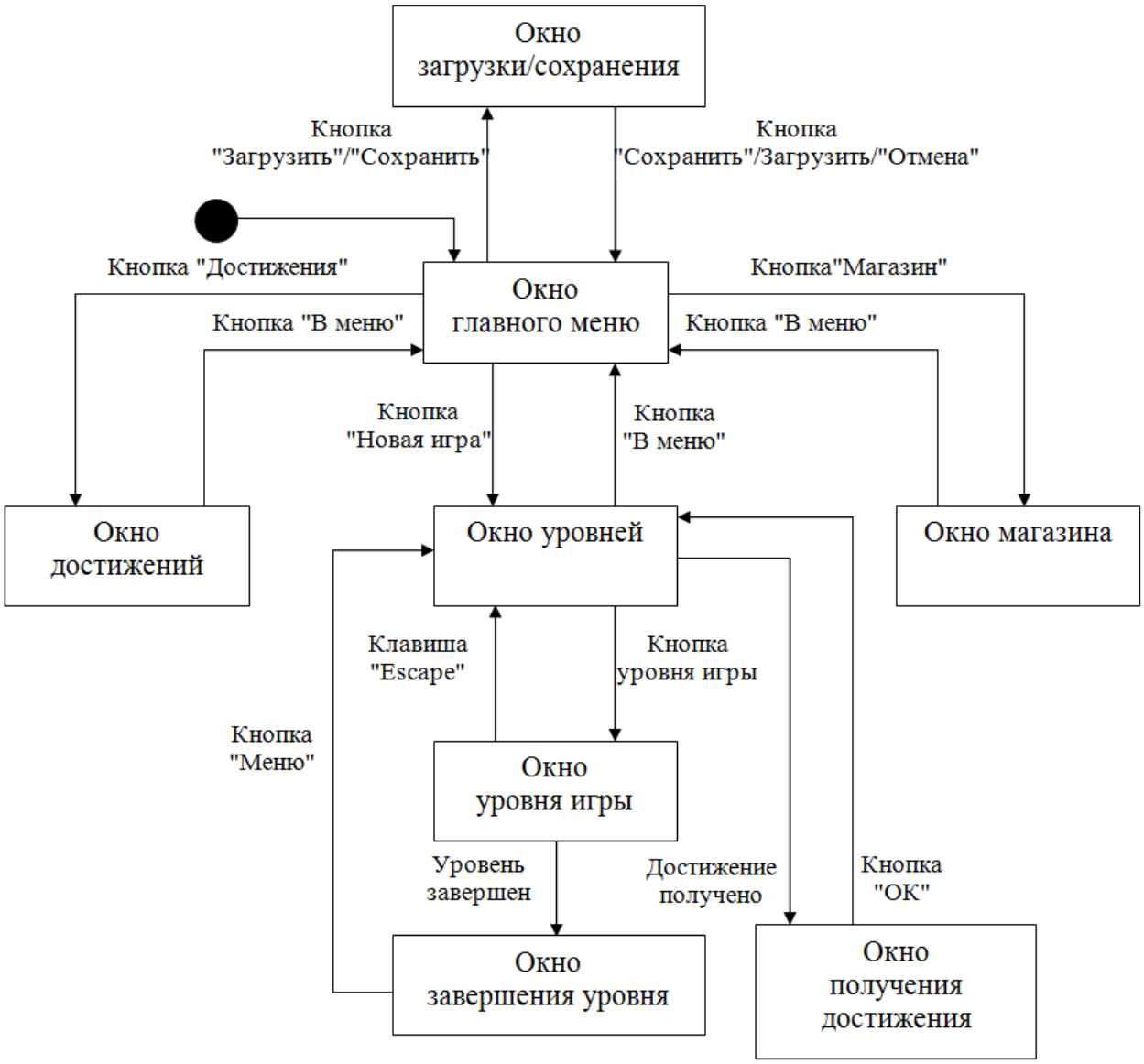

Рисунок 7 – Карта диалоговых окон

4 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

4.1 Выбор инструментария

Для реализации нашей игры будем использовать объектно-ориентированный язык программирования ActionScript 3.0 на платформе Adobe Flash [2]. Эта платформа является основой для разработки многофункциональных интерактивных веб-приложений, презентаций и пользовательских интерфейсов. Приложения Flash позволяют разработчикам добиться более полного контроля над объектами и их параметрами, анимировать объекты программными способами (то есть без использования монтажного стола), выводить данные из Flash и получать их из внешних источников. Кроме того, стоит отметить, что игры на Flash писать значительно проще и удобнее, чем, например, на платформе HTML5.

Создавать наше приложение будем на чистом языке ActionScript без использования дополнительных библиотек. Для этого воспользуемся интегрированной средой разработки Adobe Animate CC 2020, которая позволяет разрабатывать интерактивную графику и анимировать объекты для создания игр, мультфильмов, сайтов и т.п. [6]. Эта среда разработки обладает более стабильной производительностью, что влияет на улучшение интеграции рабочих процессов. Также в эту IDE были добавлены новые возможности для работы с графикой, что позволяет использовать новые параметры для работы с анимацией.

4.2 Особенности реализации модуля логики игрового процесса

Язык ActionScript 3.0 позволяет размещать все объекты, которые будут участвовать во взаимодействии на так называемых кадрах (frame). Для кадрового взаимодействия в этом языке есть базовый класс MovieClip, который обеспечивает изменение кадров при помощи методов gotoAndPlay() и gotoAndStop() [8]. Этот класс в свою очередь использует функции своего родительского абстрактного класса DisplayObject, который является базовым

классом для всех объектов, которые можно разместить на экране [7]. Класс DisplayObject поддерживает базовые функции, такие как положение объекта по осям x и y, а также имеет функции для проверки столкновения с определенным объектом (hitTestObject) и с определенной точкой (hitTestPoint). Все свойства этого абстрактного класса являются открытыми, что позволяет изменять их напрямую.

Взаимодействие с игрой начинается после вызова функции createGame(), которая генерирует карту в зависимости от того, какой уровень был выбран. После этого запускается процедура OnEnterFrameHandler(), которая работает в бесконечном цикле и отвечает за взаимодействие между всеми объектами игрового мира. Опишем все классы, которые участвуют в нашей игре:

- Класс ground_h, объекты которого реализует физику взаимодействия с игроком путем проверки столкновения объектов;

- Класс thorns, объекты которого являются ловушкой для игрока и при столкновении с ними герой может погибнуть;

- Класс flag, объекты которого являются концом уровня и проверяют столкновение с главным героем, запускают анимацию завершения уровня;

- Класс money, объекты которого являются монетками, которые может собрать игрок при столкновении с ними. При этом увеличится счетчик собранных монет;

- Класс InteractionObject, объекты которого представляют собой объекты, с которыми непосредственно может взаимодействовать игрок для достижения своих целей в игре. Этот класс включает в себя объекты ящика, секретов, лифтов, платформ-ловушек;

- Класс life, объекты которого наглядно показывают, сколько жизней на данный момент осталось у игрока. При смерти игрока объект этого класса изменяет свой внешний вид в соответствии с глобальным счетчиком оставшихся жизней;

- Класс ground, который содержит в себе объекты всех классов. Генерация карты происходит при выборе уровня из меню уровней при помощи функции createGame(). После этого в зависимости от конкретных действий игрока объект может изменять свой внешний вид при помощи смены кадров;

- Класс ball_frag, объект которого является главным героем уровня. Он осуществляет взаимодействие со всеми объектами игрового мира, используя для этого проверку столкновений. При столкновении объект этого класса запускает анимацию смерти, последовательно изменяя кадры.

Все классы, размещенные на игровом поле (ground), реализуют метод, который устанавливает и фиксирует начальный кадр каждого объекта при генерации уровня.

Объекты классов ground и ball_frag в свою очередь располагается на сцене (stage). Этот класс определен самим языком программирования и отвечает за все отображение содержимого Flash. В этом классе мы определили методы, которые обеспечивают взаимодействие между пользователем и игровым процессом и обеспечивают обновление состояния сцены. Методы KeyUpHandler() и KeyDownHandler() фиксируют нажатие клавиш клавиатуры пользователем. Метод RestartGame() переводит все объекты на сцене на момент начального состояния уровня. Метод timerCount() необходим для отсчета времени до определенных событий, происходящих на сцене.

Физическая диаграмма классов со всеми функциями, реализованными при помощи языка программирования, представлена на рисунке 8.

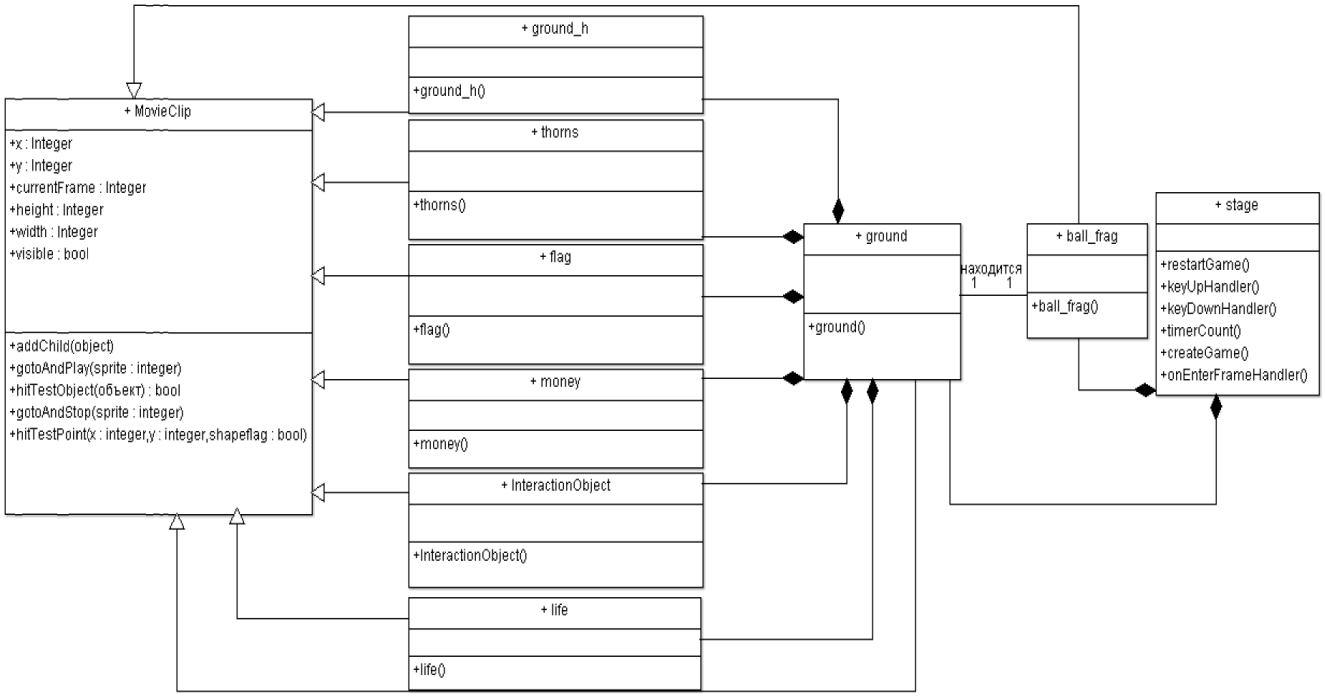

Рисунок 8 – Физическая модель

4.3 Экранные формы

Для реализации графического интерфейса будем использовать приложение Adobe Animate CC 2020 [6]. Опишем основной интерфейс нашего приложения.

Рассматриваемая в данной работе игра "Платформер" выглядит так:

1) В главном окне программы, кроме кнопок взаимодействия с окном, находятся кнопки "Новая Игра", "Достижения", "Магазин", а также кнопки для взаимодействия с файлом: "Загрузить", "Сохранить" (Рисунок 9);

Рисунок 9 – Главное меню игры

2) При нажатии на кнопку "Новая игра" появится окно уровней с тремя кнопками уровней и кнопкой "В меню", возвращающей пользователя в главное меню (Рисунок 10);

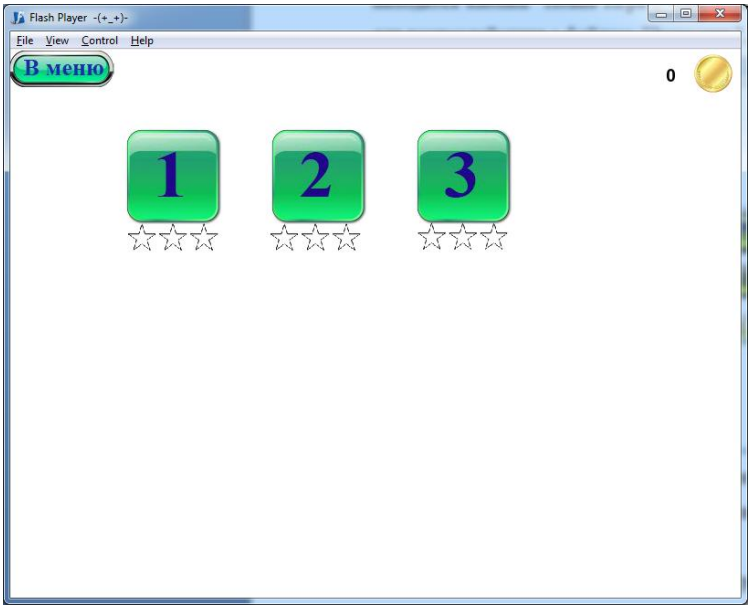

Рисунок 10 – Окно уровней

3) При выборе определенного уровня появится сформированный уровень, в котором пользователь, управляя персонажем, может взаимодействовать с различными объектами окружающего мира (Рисунок 11);

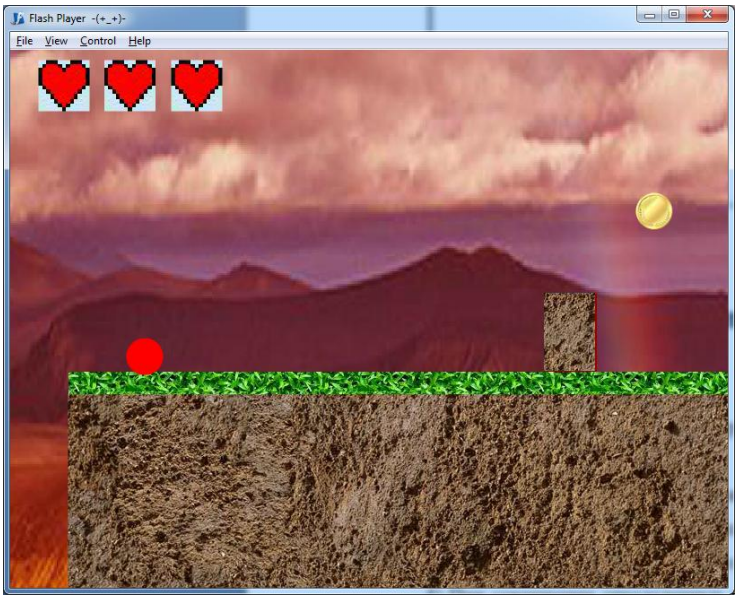

Рисунок 11 – Уровень игры

4) После прохождения уровня появится окно с информацией об общем успехе прохождения определенного уровня (Рисунок 12);

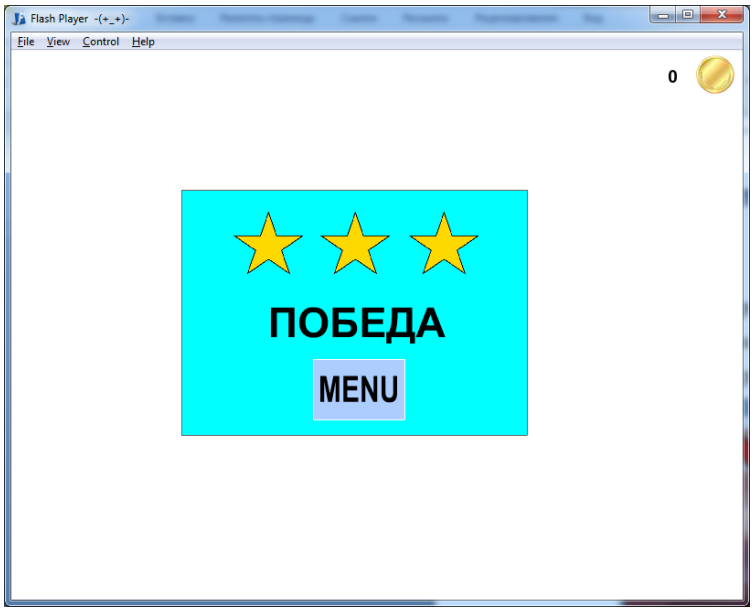

Рисунок 12 – Окно завершения уровня

5) При совершении определенных действий после прохождения уровня может появиться окно, информирующее пользователя о получении им определенного достижения в игре (Рисунок 13);

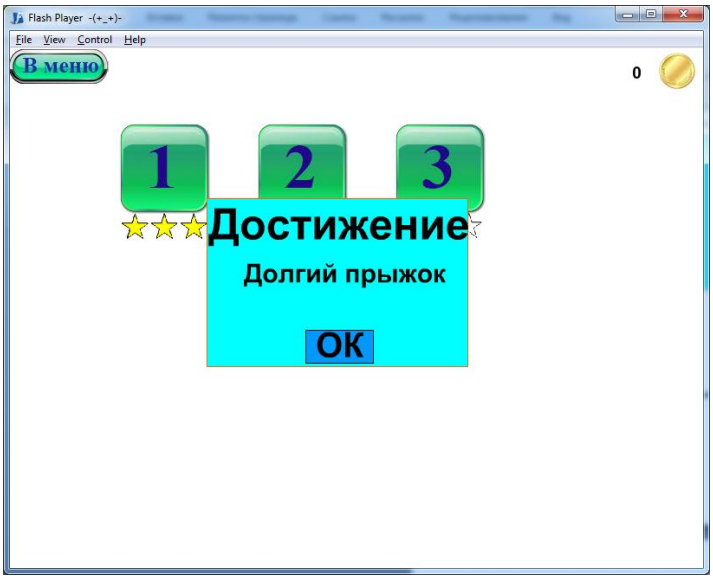

Рисунок 13 – Окно достижений

6) При нажатии в главном меню кнопки "Достижения" на экране покажется список всех достижений. Если достижение не получено, кликом мыши на него игрок сможет узнать, что требуется для его получения (Рисунок 14);

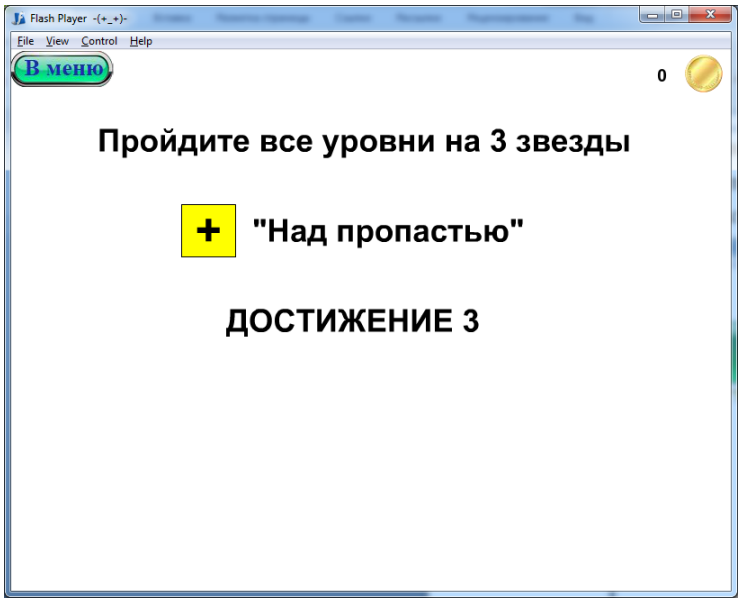

Рисунок 14 – Список достижений

7) При нажатии в главном меню кнопки "Магазин" на экране появится список всех возможных товаров. Если у игрока достаточно денег, он сможет приобрести определенный товар. При нажатии на кнопку "Отмена" товар пропадет из списка активных у пользователя, а при нажатии на кнопку

"Выбрать" умение или персонаж снова станут активными и их можно использовать на уровнях (Рисунок 15);

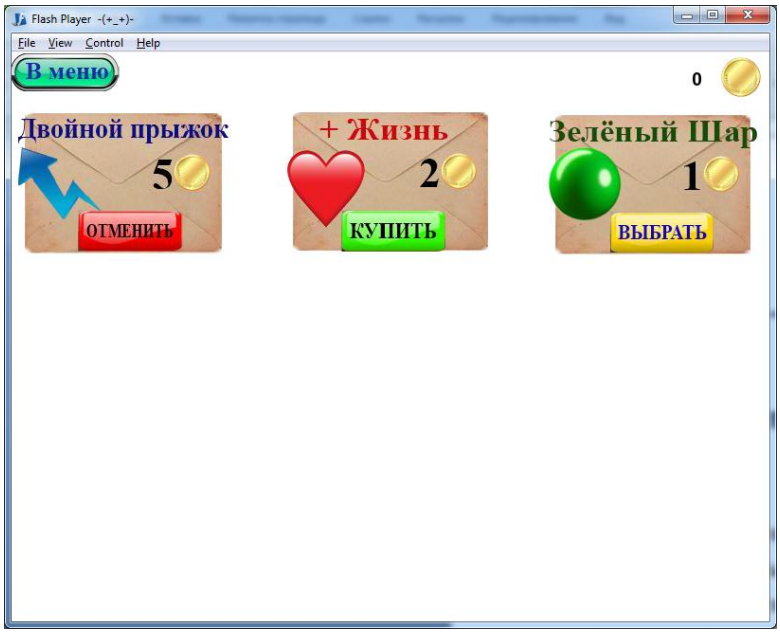

Рисунок 15 – Окно магазина

8) При нажатии в главном меню кнопки "Сохранить" или "Загрузить" появится окно с выбором места сохранения/загрузки текущего прогресса игры (Рисунок 16).

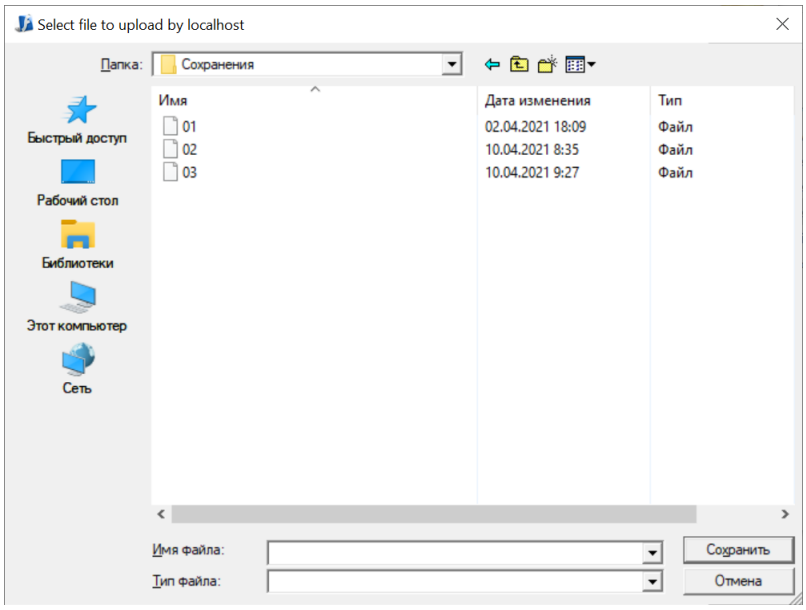

Рисунок 16 – Окно загрузки/сохранения

ЗАКЛЮЧЕНИЕ

В настоящей работе было разработано и описано программное приложение - игра "Платформер" с возможностью сохранения и загрузки игрового прогресса. В будущем это игровое приложение можно расширять, добавляя в него новые уровни, достижения, товары, а также увеличивая его функционал добавлением новых механик, ловушек, головоломок и сражений с враждебными персонажами.

Мы разработали основные требования к программе, спроектировали и рассмотрели особенности ее реализации. Таким образом, можно сказать, что все поставленные перед нами задачи были выполнены и цель курсового проекта была достигнута.

СПИСОК ЛИТЕРАТУРЫ

1. Модульная архитектура и модульный дизайн [Электронный ресурс]. - Режим доступа: https://www.archidizain.ru/2019/04/blog-post_11.html (дата обращения: 2.04.2021).

2. Описание языка ActionScript 3.0 [Электронный ресурс]. - Режим доступа: https://helpiks.org/7-8903.html (дата обращения: 2.04.2021).

3. ОСНОВЫ УНИФИЦИРОВАННОГО ЯЗЫКА МОДЕЛИРОВАНИЯ [Электронный ресурс]. - Режим доступа: https://www.sites.google.com/site/anisimovkhv/learning/pris/lecture/tema11 (дата обращения: 2.04.2021).

4. Платформенная игра - Platform game [Электронный ресурс]. - Режим доступа: https://ru.qaz.wiki/wiki/Platform_game (дата обращения: 2.04.2021).

5. Функциональные и нефункциональные требования (Functional and Non-functional Requirements) [Электронный ресурс]. - Режим доступа: https://studfile.net/preview/2152457/page:4/ (дата обращения: 2.04.2021).

6. Adobe Animate CC 2020 [Электронный ресурс]. - Режим доступа: http://www.efxi.ru/news/news_4092.html (дата обращения: 2.04.2021).

7. DisplayObject - AS3 AS3 [Электронный ресурс]. - Режим доступа: https://help.adobe.com/ru_RU/FlashPlatform/reference/actionscript/3/flash/display/DisplayObject.html (дата обращения: 2.04.2021).

8. MovieClip - AS3 [Электронный ресурс]. - Режим доступа: https://help.adobe.com/ru_RU/FlashPlatform/reference/actionscript/3/flash/display/MovieClip.html (дата обращения: 2.04.2021).

9. Red Ball описание игры [Электронный ресурс]. - Режим доступа: https://greatgamer.ru/short_reviews/red_ball.html (дата обращения: 2.04.2021).

10. VEX [Электронный ресурс]. - Режим доступа: https://ag.ru/games/vex (дата обращения: 2.04.2021).

ПРИЛОЖЕНИЕ А

(обязательное)

РЕАЛИЗАЦИЯ ПРОГРАММЫ

```
function restartGame() { //Функция перезапуска игры
                player.gotoAndStop(1);
                if(shop_jump[2]==1)player.gotoAndStop(16);
                player.visible=true;
                player.rotation=0;
                player.x = 150;
                player.y = 365;
                ground.x = -save_x+300;
                ground.y = 400;
                ground.col_life.x=-ground.x+60;
                ground.col_life.y=-360;
                if(levels[1]==1){
                        ground.gr_h.trap_1.t1.x=45;
                        ground.gr_h.trap_1.t1.y=-35;
                        ground.gr_h.box.x=2022;
                        ground.gr_h.box.y-153;
                }
                if(levels[2]==1){
                        timer.reset();
                        ground.gr_h.tr_b.y=-152;
                        ground.gr_h.tr_b.visible=true;
                        ground.btn_vis.gotoAndStop(1);
                        ground.gr_h.trap_t_2.y=-68;
                        timeLeft=0;
                        timer=new Timer(1000,timeLeft);
            timer.addEventListener(TimerEvent.TIMER,timerCount);
                        timer.reset();
                }
                death=0;
            }
//Функция нажатия кнопок движения
            function keyDownHandler (e:KeyboardEvent) {
                switch(e.keyCode) {
                        case 65:
                                left = true;
                                break;
                        case 37:
                                left = true;
                                break;
```

```
                    case 87:
                        up = true;
                        break;
                    case 38:
                        up = true;
                        break;
                    case 39:
                        right = true;
                        break;
                    case 68:
                        right = true;
                        break;
                    case 82:
                        restart_button = true;
                        break;
                    case 27:
                        escape_button = true;
                        break;
                }
//Функция изменения состояния достижения
function explanation_1 (e:MouseEvent) {
            if(flag_dost[0]!=1){
                if(selector[0]==0){
                    ach_1.gotoAndStop(2);
                    selector[0]=1;
                }
                else if(selector[0]==1){
                    ach_1.gotoAndStop(1);
                    selector[0]=0;
                }
            }
        }
//условие прокрутки экрана при движении
if (player_sideScrollingMode) {
                    ground.x -= player_xRight;
                    ground.col_life.x += player_xRight;
                    ground.x += player_xLeft;
                    ground.col_life.x -= player_xLeft;
                    ground.y -= player_y;
                    ground.col_life.y += player_y;
                } else {
                    player.x += player_xRight;
                    player.x -= player_xLeft;
                    player.y += player_y;}
```