**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ**

**Кафедра веб-технологий и компьютерного моделирования**

Суражев

Кирилл Леонидович

**Создание игрового приложения под Android с использованием gameEngine**

Дипломная работа

Научный руководитель: доцент, кандидат физ.-мат. наук Барвенов Сергей Александрович

Допущен к защите

«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Зав. кафедрой веб-технологий и компьютерного моделирования

Доктор физико-математических наук, доцент Волков В.М.

Минск,2022

ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc105185470)

[РЕФЕРАТ 4](#_Toc105185471)

[ВВЕДЕНИЕ 7](#_Toc105185472)

[ГЛАВА 1. ФУНКЦИИ, ЖАНРЫ, СОЗДАНИЕ ИГР 8](#_Toc105185473)

[1.1 Анализ игровых движков 8](#_Toc105185474)

[1.2 Анализ жанров игр 10](#_Toc105185475)

[1.3 Анализ игр в жанре аркада 12](#_Toc105185476)

[1.3.1 Общая информация и примеры наиболее известных игр 12](#_Toc105185477)

[1.3.2 Анализ целевой аудитории 14](#_Toc105185478)

[1.4 Анализ игр в жанре платформер 16](#_Toc105185479)

[1.4.1 Общая информация и примеры наиболее известных игр 16](#_Toc105185480)

[1.4.2 Анализ целевой аудитории 19](#_Toc105185481)

[1.5 Анализ возможных видов монетизации 21](#_Toc105185482)

[ГЛАВА 2. РАЗРАБОТКА ИГРЫ FLAPPY BIRD SIMULATOR 22](#_Toc105185483)

[2.1 Конкретное описание игры 22](#_Toc105185484)

[2.2 Технологическая часть 22](#_Toc105185485)

[2.3 Базовая настройка 24](#_Toc105185486)

[2.4 Управление птицей, её физика, анимация 25](#_Toc105185487)

[2.5 Генерация уровня, сложность 25](#_Toc105185488)

[2.6 Интерфейс в игре 26](#_Toc105185489)

[2.7 Экран загрузки 27](#_Toc105185490)

[2.8 Главное меню 27](#_Toc105185491)

[2.9 Звук 28](#_Toc105185492)

[2.10 Ожидание игрока 28](#_Toc105185493)

[2.11 Диаграмма классов 28](#_Toc105185494)

[2.12 Итоговый результат 29](#_Toc105185495)

[2.13 Тестирование игры 30](#_Toc105185496)

[2.13 Дальнейшие возможности развития игры 31](#_Toc105185497)

[ГЛАВА 3. РАЗРАБОТКА ИГРЫ ABOVETALE 32](#_Toc105185498)

[3.1 Конкретное описание игры 32](#_Toc105185499)

[3.2 Технологическая часть 33](#_Toc105185500)

[3.3 Введение 34](#_Toc105185501)

[3.4 Игровой уровень и движение персонажа 34](#_Toc105185502)

[3.5 Анимации персонажа (бездействие, ходьба, прыжок) 36](#_Toc105185503)

[3.6 Движение камеры и параллакс эффект 37](#_Toc105185504)

[3.7 Инвентарь, подбор и сброс предметов 38](#_Toc105185505)

[3.8 Диалоги 39](#_Toc105185506)

[3.9 Управление под Android 41](#_Toc105185507)

[3.10 Квесты 41](#_Toc105185508)

[3.11 Смена сцен, локаций 42](#_Toc105185509)

[3.12 Работа со звуком 43](#_Toc105185510)

[3.13 Диаграмма классов 44](#_Toc105185511)

[3.14 Иерархия сцен 44](#_Toc105185512)

[3.15 Тестирование игры 45](#_Toc105185513)

[3.16 Дальнейшие возможности развития игры 46](#_Toc105185514)

[ЗАКЛЮЧЕНИЕ 47](#_Toc105185515)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 48](#_Toc105185516)

[ПРИЛОЖЕНИЕ А 49](#_Toc105185517)

[А1 Скрипт GameHandler.cs 49](#_Toc105185518)

[А2 Скрипт GameAssets.cs 49](#_Toc105185519)

[ПРИЛОЖЕНИЕ B 51](#_Toc105185520)

[B1 Скрипт Quests.cs 51](#_Toc105185521)

[B1 Скрипт DialogueManager.cs 52](#_Toc105185522)

РЕФЕРАТ

Дипломная работа 53 страниц, 50 рисунков, 13 использованных источников.

Цель дипломной работы: изучение технологий разработки мобильных игр, создание демонстрационной версии игрового приложения, позволяющего хорошо провести досуг.

В ходе выполнения работы были созданы две демострационные версии разных игр, и для каждой из них были реализованы следующие основные задачи:

* выбор движка для разработки.
* выбор игрового жанра
* анализ наиболее популярных представителей выбранного жанра
* анализ технологий выбранного движка для разработки игры
* создание игрового приложения на основе изученных технологий

Дальнейшее развитие проекта возможно за счёт увеличения контента, а также существенных инвестиций. Работа имеет практическую ценность, так как её, в дальнейшем, можно монетизировать. Дипломная работа выполнена автором самостоятельно.

**РЭФЕРАТ**

Дыпломная праца 53 старонак, 50 малюнкаў, 13 выкарыстаных крыніц.

Цель дыпломнай працы: вывучэнне тэхналогій распрацоўкі мабільных гульняў, стварэнне дэманстрацыйнай версіі гульнявога дадатка, якое дазваляе добра правесці досуг.

У ходзе выканання работы былі створаны дзве дэмастрацыйныя версіі розных гульняў, і для кожнай з іх былі рэалізаваны наступныя асноўныя задачы:

• выбар руху для распрацоўкі.

• выбар гульнявога жанру

• аналіз найбольш папулярных прадстаўнікоў выбранага жанру

• аналіз тэхналогій абранага руху для распрацоўкі гульняў

• стварэнне гульнявых прыкладанняў на аснове вывучаных тэхналогій

Дальнейшае развіццё праекта магчыма для зняцця павелічэння кантэнту, а таксама істотных інвестыцый. Работа мае каштоўную каштоўнасць, так як яе, у далейшым, можна манетызаваць. Дыпломная работа выканана аўтарам самастойна.

**ABSTRACT**

The diploma work 53 pages, 50 drawings, 13 sources used.

The purpose of the diploma work: to study the technology of mobile game development, to create a demo version of the game application that allows you to have a good time.

In the course of the work, two demo versions of different games were created, and the following main tasks were implemented for each of them:

• selection of the engine for development.

• choice of game genre

• analysis of the most popular representatives of the chosen genre

• analysis of the technologies of the selected engine for game development

• creation of a game application based on the studied technologies

Further development of the project is possible due to the increase in content, as well as significant investments. The work has practical value, as it can be monetized in the future. The diploma work was done by the author himself.

ВВЕДЕНИЕ

Целью данной дипломной работы разработать игру под Android с использованием gameEngine.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* Проанализировать и выбрать gameEngine для разработки игры
* Проанализировать и выбрать жанр разрабатываемой игры
* Проанализировать особенности выбранного жанра и классических представителей этого жанра.
* Провести исследования и дать характеристику целевой аудитории игр в выбранном жанре
* Разработать игру в выбранном жанре
* Провести тестирование игры

Актуальность работы заключается в том, что, разрабатывая новую игру приходится использовать знания из многих разделов и программирования, и дизайна. Также игры можно монетизировать. Например, можно внедрить внутриигровые покупки, рекламу, разместить игру в магазине и т.д. Разработчики игр могут создавать свои миры, людей, флору и фауну, различных существ и связь всего этого друг с другом. И эти творения способны развлекать, развивать и обладают силой затрагивать человеческие сердца.

# ГЛАВА 1. ФУНКЦИИ, ЖАНРЫ, СОЗДАНИЕ ИГР

## 1.1 Анализ игровых движков

Игровой движок (game engine) — программное обеспечение, предназначенное для разработки игр на различных платформах

В данной работе будет рассмотрено 3 наиболее распространенных игровых движка для разработки мобильных игр: Unity 2018, Unreal Engine 4, CryEngine V.

**Unity 2020** — кроссплатформенный движок для разработки игр, разработанный компанией Unity Technologies [2]. Имеет огромное количество поддерживаемых платформ (ПК, приставки, мобильные ОС, VR платформы и др.). Можно отметить довольно низкие системные требования данного движка.

Имеет широкий спектр различных инструментов, как для художников-разработчиков (например, Timeline для разработки анимационных сцен, Cinemachine — набор «умных» и динамических камер, Progressive Lightmapper для работы с освещением, и Autodesk Maya - для работы с 3D-анимацией и моделированием), так и для программистов-разработчиков.

Движок позволяет вести разработку как 2D, так и 3D проектов.

Unity 2020 поддерживает такие языки программирования как C# и UnityScript (основанный на JavaScript).

Имеется 3 версии этого игрового движка — Personal (бесплатно), Plus (35$ в месяц), Pro (125$ в месяц). Данные версии отличаются лишь предлагаемым функционалом и максимально допустимым годовым доходом (100 тыс. долларов, 200 тыс. долларов и без ограничений соответственно).

Согласно статистике [3], в мире установлено уже более 28 млрд игр на платформе Unity, кроме того, на Unity сделано 50% мобильных игр и 60% контента для альтернативной и виртуальной реальности.

Кроме того, данный движок является самым распространенным в мире и его используют около 45% разработчиков.

**CryEngine V** — самая последняя версия игрового движка от компании Crytek [1].

Этот игровой движок делает большой упор на визуальную составляющую проекта, поэтому имеет большое количество функций и инструментов для достижения превосходного качества изображения. Кроме того, в движке реализовано немало инструментов для работы с физикой объектов.

CryEngine поддерживает языки программирования С++ и Lua, в нем доступна система визуальных скриптов Flowgraph, которая позволяет создавать и контролировать игровую логику и события без необходимости писать скрипты вручную.

Имеется большое количество настроек анимации персонажей, в том числе параметрическая скелетная анимация. Кроме того, CryEngine включает в себя встроенную система искусственного интеллекта, позволяющая настроить реалистичное поведение неигровых персонажей.

Распространяется со свободной лицензией, при условии, что продукт, созданный на движке, приносит прибыль менее 5 тысяч долларов в год, в противном случае 5% прибыли отчисляется Crytek.

Данный движок является, прежде всего, отличным инструментом для разработки 3D проектов, где делается упор на визуальную составляющую и физику объектов.

**Unreal Engine 4** — игровой движок, разработанный компанией Epic Games [4]. Обладает широким спектром поддерживаемых платформ, хотя и не настолько широким, как у Unity.

Движок позволяет добиться фотореалистичной графики с помощью большого количества настроек рендера, динамических теней, отражений каналов освещения.

Одной из отличительных особенностей является полный доступ к исходному коду на языке C++. Также движок содержит систему визуальных скриптов Blueprint, которая позволяет управлять игровой логикой без написания скриптов вручную, хоть такая возможность и имеется (на том же языке C++).

Unreal Engine также имеет широкую поддержку разработки мультиплеерных игр. Данный движок поставляется с масштабируемой и проверенной архитектурой клиент/сервер.

Данный движок подходит прежде всего для разработки трехмерных игр с упором на мультиплеер.

В заключение можно сказать, что все 3 рассмотренных движка являются неплохими инструментами для создания игр под Android инди-разработчиками, при этом каждый имеет свои отличительные черты, которые смогут помочь каждому разработчику выбрать именно тот, что лучше всего подойдет для реализации его проекта. В данной работе был выбран движок Unity, т.к. он проще остальных в освоении, а с точки зрения необходимых для дальнейшей работы встроенных в движок функций подходит идеально.

## 1.2 Анализ жанров игр

1.Action

Этот жанр означает использование виртуального пространства, в котором есть мишень и стрелок. Может быть человек, например, и какое—либо транспортное средство (танк, космический корабль, самолет). Пространство игры или двухмерное, или трехмерное. Основная цель «экшена» – поразить как можно большее количество мишеней. В качестве примера: Left 4 Dead, третий Doom, четвертый Devil May Cry.

2.MMORPG

Речь идет о многопользовательском игровом жанре, в котором предусмотрен отдельный виртуальный мир. В этом пространстве приходится уживаться большому количеству различных персонажей. Чаще всего, речь идет о мире фэнтези. Здесь, как известно, можно встретить кого угодно – фантастические персонажи. У каждого из них есть определенные способности, собственная цель, а также роль в игре. Игрок имеет возможность стать тем, кто ему больше по душе.

Сравнивая со стандартной RPG, геймплей MMORPG не исчезает никуда (не имеет значения, находится пользователь в игре или нет). В качестве примера: World Of Tanks, World of Warcraft, второй LineAge.

3.RPG

Категория ролевых игр может иметь как развлекательные, так и учебные разработки. Игра начинается в придуманном мире. В основе может использоваться сюжет художественного вида или от начала до конца придуманный разработчиками игры. В данной вымышленной реальности геймеру, как правило (но не всегда), приходится решать какую—нибудь определенную задачу. Во время игры необходимо достичь определенных навыков. В виртуальной реальности можно увидеть адекватную реакцию на различные действия, а также поступки. Для примера: второй Mass Effect, King’s Bounty, четвертый Fallout.

4.Аркады

Достаточно примитивный жанр игры, несмотря на это, очень популярной в наши дни. «Перекочевал» на компьютеры с различных игровых автоматов, а также консолей. Аркадными бывают гонки, файтинги и еще большое количество разновидностей игр. Суть обычной аркады основывается на том, что геймеру необходимо пройти определенный уровень за минимальный период времени. Во время игры приходится преодолевать различные препятствия, а также собирать бонусы. В качестве примера: Super Mario, Super Meat Boy, Pac—Man.

5.Головоломки

О смысле этой игры можно легко догадаться из названия. Геймеру нужно разобраться с какой—нибудь непростой задачей. К примеру, речь может идти о составлении предмета или выбора из большого количества вариантов одного правильного решения. Также возможны и другие вариации.

Известно, головоломки являются самым старым видом игр на Земле. В древнюю китайскую игру «Танграм» коротали время свыше 3000 лет назад. В качестве примера: Crayon Physics Deluxe, World of Goo.

6.Гонки

Смысл этого жанра понятен каждому геймеру. Игрок должен участвовать в различных гонках. Из возможных вариантов могут быть соревнования «Формулы 1», турниры стритрейсеров, а также демонстрация мастер—класса на более сложных транспортных средствах – космические корабли или самолеты. Гонки могут быть как трехмерными, так и двухмерными.

7.Приключения

Данный вид игр отличается наличием специального литературного сюжета. Именно он используется для создания геймплея. Герой, которым придется стать игроку, будет жить в данном мире. Во время игры он будет решать большое количество различных интеллектуальных задач, в том числе и головоломок.

Сравнивая с RPG, в данном случае геймеру не придется случайно взаимодействовать с другими персонажами игры – есть четкий сюжет, за рамки которого нельзя выходить. В качестве примера: Syberia, Fahrenheit, The Longest Journey.

8.Симуляторы

Речь идет о полноценной и очень точной имитации управления определенным объектом (может быть машина, танк или самолет). Сначала симуляторы создавались для военных солдат, чтобы обучать их управлять техническими устройствами. Однако прогресс стал причиной того, что обучающее программное обеспечение заняло свое отдельное место в индустрии игровых разработок.

Сравнивая с гонками, в симуляторах реализовано более точное управление и нет необходимости соревноваться. В качестве примера: Trainz, пятый Silent Hunter.

9.Спортивные

Данная категория игр представляет собой соревнования различных команд. В основном, речь идет о футбольном, баскетбольном, хоккейном или теннисном матчах. Геймерам придется взять шефство над командой: формировать ее состав и привести к победе. В качества примера: серия PES и серия Fifa.

10.Стратегии

Наверное, мы сейчас говорим о наиболее интеллектуальном виде игровых разработок. С самого начала геймер получает возможность создать собственный мир или, в крайнем случае, государство, где придется не только жить, а и управлять. Это не так легко и приятно, как можно подумать сначала. Необходимо детально продумать инфраструктуру, тонко организовать все планы, создать для жителей подходящие условия проживания, дать требуемое количество рабочих мест. В этом деле будут мешать жадные правители других государств – им так и хочется заполучить вашу территорию, поэтому ее нужно будет защищать.

В качестве примера: третий Warcraft, Age of Empires III.

11.Файтинги

Данный жанр игр подходит тем, кто желает ощутить себя в роли «настоящего мужика». Геймера ждет нескончаемое число рукопашных боев с самыми различными врагами. Каждый бой проходит на арене. Покинуть ее удастся, если закончить игру. То есть, игровой мир в этом случае сильно ограничен. В качестве примера: игры серии Mortal Kombat или Tekken [8].

В результате анализа жанров и личного опыта игры в них, было решено реализовать две игры: первая – игра в жанре аркада Flappy Bird Simulator, вторая – игра в жанре платформер с элементами адвенчуры и квеста AboveTale. Далее будет описан анализ игр в выбранных жанрах.

## 1.3 Анализ игр в жанре аркада

### 1.3.1 Общая информация и примеры наиболее известных игр

Если обобщить, все аркады имеют несколько общих черт, по которым их можно легко узнать. Это, в частности:

* упрощенный, но интенсивный процесс;
* упрощенная графика;
* отсутствие явно прослеживаемого сюжета;
* бесконечность (в игре может быть огромное число уровней);
* возможность набрать очки за выполнение различных действий;
* небольшой промежуток времени, необходимый для их прохождения;
* наличие нескольких жизней, чтобы была возможность пройти один и тот же уровень несколько раз.

Для лучшего понимания жанра аркада были рассмотрены классические представители данного жанра, которые получили наибольшую популярность и одобрение среди игроков.

**Jetpack Joyride**

Игра в жанре аркада от разработчика Halfbrick Studios (см. Рисунок 1.1) [5].

Игроку необходимо перемещать персонажа в строго горизонтальном направлении, собирать монеты и избегать препятствий. К плюсам можно отнести инновацию в плане игрового процесса – игра стала первооткрывателем конкретной модели игры. К минусам можно отнести довольно примитивное музыкальное сопровождение и уже устаревшую графику.



Рисунок 1.1 Скриншот из игры Jetpack Joyride

**Ski Safari**

Игра в жанре аркада от разработчика Defiant Development (см. Рисунок 1.2) [6]. Игроку необходимо выполнять спуск с горы, при этом исполняя различные эквилибристические приемы. К плюсам можно отнести отличную динамику игрового процесса и хорошую графическую составляющую. Минусом игры является вторичность игрового процесса на рынке. Игра приобрела свою популярность благодаря динамичному игровому процессу и ярким сценам.



Рисунок 1.2 – Скриншот из игры Ski Safari

**Subway Surfers**

Игра в жанре аркада от разработчика Kiloo Aps (см. Рисунок 1.3) [7].

Игрок бесконечно бежит по железнодорожным путям, избегая препятствия и собирая на своем пути монеты и игровые бонусы. Плюсами можно назвать разнообразную кастомизацию персонажа, динамичный игровой процесс и красочный интерфейс. К минусам можно отнести излишнюю монетизацию игры, а также невозможность синхронизации результата с другими устройствами. Игра является рекордсменом по количеству загрузок в своем жанре. Пользователи установили игровое приложение из мобильного магазина Google Play более 1 млрд. раз.



Рисунок 1.3 – Скриншот из игры Subway Surfers.

### 1.3.2 Анализ целевой аудитории

При разработке компьютерной игры важно учитывать целевую аудиторию. Незнание целевой аудитории может повлечь за собой такие неприятные последствия, как безызвестность продукта среди тех, кому он действительно нужен.

Чтобы понять, кто целевая аудитория игр в жанре аркада, необходимо ответить на вопрос: почему аркады популярны? В большинстве своем, аркады не заставляют сильно думать. Там большая часть геймплея основана на рефлексах человека. Тот же World of Tanks отличается большой аркадной составляющей, так как управлять реальным танком сложнее. Но тем не менее, можно получить массу удовольствия, так как в один и тот же момент происходит и отдых, и развитие отдельных частей мозга. Так, например, было показано экспериментально, что люди, играющие в подобные аркады, отличаются максимально быстрой реакцией, так как реагируют на десятки незапланированных, не поддающихся анализу, явлений.

Таким образом, делаем вывод: наша игра рассчитана на людей, обладающих быстрой реакцией и способных следить за несколькими явлениями одновременно (одним словом – на хардкорную аудиторию).

Также для анализа целевой аудитории игроков был проведен опрос, в котором респонденты указывали свой возраст, пол, а также играют ли они в игры в жанре аркада или нет.

При проведении опроса было опрошено 1000 человек, среди них 500 человек женского пола и 500 человек мужского пола.

Из 500 человек женского пола, на вопрос «Играете ли Вы в игры в жанре аркада?» положительно ответили 100 человека, что составляет 20 %.

Из 500 человек мужского пола, на вопрос «Играете ли Вы в игры в жанре аркада?» положительно ответили 290 человек, что составляет 58 %.

Из этого можно сделать вывод, что основная аудитория, играющая в игры подобного жанра, состоит из людей мужского пола.

Таким образом, результаты анализа (см. Рисунок 1.4)

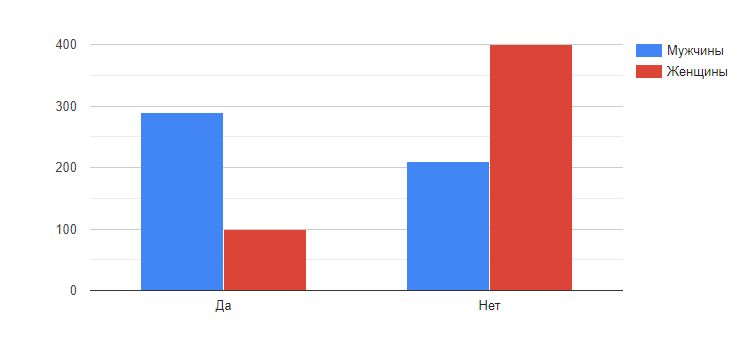
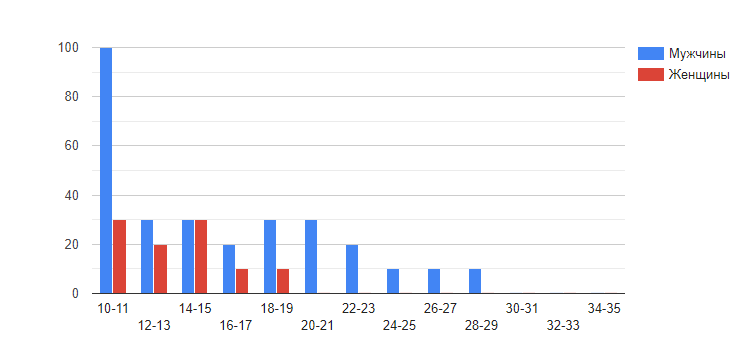


Рисунок 1.4 — Результаты ответа на вопрос «Играете ли вы в игры в жанре аркада?»

Возраст людей, прошедших опрос, был от 10 до 35 лет.

Анализ разных возрастных групп (см. Рисунок 1.5):

 Рисунок 1.5 — Результаты опроса с выборкой по возрасту

В результате анализа ответов респондентов, было решено, что основная целевая аудитория игры: пользователи мужского пола в возрасте от 10 до 20 лет.

Опрос был проведён среди пользователей форума на сайте Gamedev.ru [13]

## 1.4 Анализ игр в жанре платформер

### 1.4.1 Общая информация и примеры наиболее известных игр

Платформеры стали популярными в 1980-х годах. Тогда же были сформулированы основные принципы этих игр. Они сводятся к следующему:

* персонаж перемещается по уровням, причём по мере продвижения прохождение вперёд становится всё более сложным;
* у уровней, как правило, есть много секретов для завлечения игроков, любящих исследовать миры.
* время от времени персонаж может получать новые способности, позволяющие проходить туда, куда раньше было не попасть (также является элементом метроидвании)
* для взаимодействий с объектами достаточно к ним прикоснуться
* большое количество разнообразных и интересных с разных точек зрения игроков. Контакт с ними может привести к смерти или же к уменьшению здоровья. Игрок может избегать их, используя бег, прыжки или иные специальные возможности.
* появление ловушек и препятствий, которые нужно обходить или же преодолевать с помощью специальных сил;
* движущиеся платформы, падение с которых приводит к потере здоровья или жизни;
* возможность собирать деньги или другие ресурсы, иногда просто очки.

**Super Meat Boy**

Игра в жанре «платформер» (см. Рисунок 1.6)

Главный герой, мясной кубик с ножками, умеющий прыгать и карабкаться по стенам, преодолевает под контролем игрока сотни препятствий — раскаленные лавовые шары, лабиринты из циркулярных пил, самонаводящиеся лазеры и ракетницы. При правильных действиях редкий уровень здесь проходится дольше, чем за минуту, — но это при правильных действиях. Ни единой ошибки не прощается. Жестоко карается любой промах: упали в пропасть — начинайте заново; напоролись на острое лезвие — заново; коснулись зловредного белого слизня… да-да, заново. [10]



Рисунок 1.6 – Скриншот из игры Super Meat Boy.

**Hollow Knight**

Игра в жанре «платформер» (см. Рисунок 1.7)

Hollow Knight — это захватывающая, красивая приключенческая игра в огромном подземном царстве Халлоунест. Двухмерная игра с акцентом на умения и исследования позволяет вам сражаться с огромным множеством смертоносных существ, избегая сложных ловушек и разгадывая древние загадки, прокладывая себе путь через отходы грибков, лесные массивы и разрушенные подземные города.

Атмосфера жутковата, а иногда и нервирует, но в игре присутствует частица юмора и легкомыслия, особенно при общении со всеми странными и удивительными персонажами, которых вы встретите на своём пути. Hollow Knight обладает красивой графикой, удобным и отзывчивым управлением, сложным, но увлекательным геймплеем. [11]



Рисунок 1.7 – Скриншот из игры Hollow Knight.

**Ori and The Blind Forest**

Игра в жанре «платформер» (см. Рисунок 1.8)

Ori and The Blind Forest (с англ. — «Ори и непроглядный лес») — компьютерная игра в жанрах платформер и метроидвания, разработанная студией Moon Studios на базе движка Unity и изданная Microsoft Studios.

Игра представляет собой двухмерный платформер. Игрок управляет персонажем по имени Ори (сказочное существо белого цвета, что-то среднее между лисицей, белкой и дикой кошкой) и защищающим его духом по имени Сейн (англ. Sein) — сгусток энергии, который следует за Ори. С помощью Сейна можно атаковать врагов, в которых он (по нажатии левой кнопки мыши, клавиши «X» на клавиатуре или кнопки геймпада) выпускает заряды «духовного пламени». Кроме того, Сейн может сделать кратковременный мощный выброс энергии (на что тратится ресурс), поражающий всех врагов поблизости от Ори и разрушающий некоторые объекты. Сам Ори изначально умеет только прыгать, но в процессе игры он сможет научиться карабкаться по стенам, нырять под воду, парить в воздухе, совершать двойные-тройные прыжки и использовать энергию, чтобы выстреливать собой или отталкивать врагов и предметы.

Игровая карта, представляющая различные участки леса, загружается целиком, и игрок волен идти туда, куда ему хочется, однако, не открыв определённую для данного участка леса способность героя, игрок в любом случае не сможет туда попасть. [12]



Рисунок 1.8 – Скриншот из игры Ori and The Blind Forest.

### 1.4.2 Анализ целевой аудитории

Теперь необходимо ответить на вопрос: какие группы геймеров играют сейчас в платформеры и что конкретно привлекает ту или иную группу. Во-первых, очевидно, что хардкорные игроки хотят увидеть достойный вызов своим навыкам, поэтому им подойдут платформеры-аркады (Например, Super Meat Boy, Celeste). Также стоит упомянуть, что платформеры часто содержат в себе элементы других жанров, например, метроидвании, где упор в геймплее ведётся на исследование игрового мира и поиск секретов, и игрокам, ищущим именно это, подойдут платформеры-метроидвании (Hollow Knight, Ori and The Blind Forest)

Таким образом, делаем вывод: в зависимости от того, элементы каких жанров включены в конкретную игру и на что делается упор в геймплее, игра может быть предназначена как для казуальных игроков, так и для хардкорных

Также для анализа целевой аудитории игроков был проведен опрос, в котором респонденты указывали свой возраст, пол, а также играют ли они в игры в жанре аркада или нет.

При проведении опроса было опрошено 1000 человек, среди них 500 человек женского пола и 500 человек мужского пола.

Из 500 человек женского пола, на вопрос «Играете ли Вы в игры в жанре платформер?» положительно ответили 350 человека, что составляет 70 %.

Из 500 человек мужского пола, на вопрос «Играете ли Вы в игры в жанре платформер?» положительно ответили 100 человек, что составляет 20 %.

Из этого можно сделать вывод, что основная аудитория, играющая в игры подобного жанра, состоит из людей мужского пола.

Таким образом, результаты анализа (см. Рисунок 1.9)

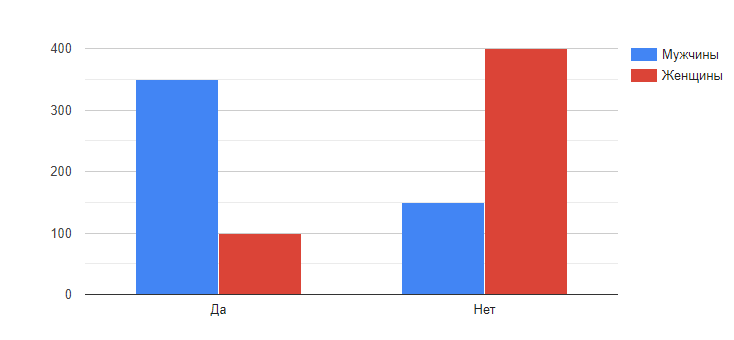
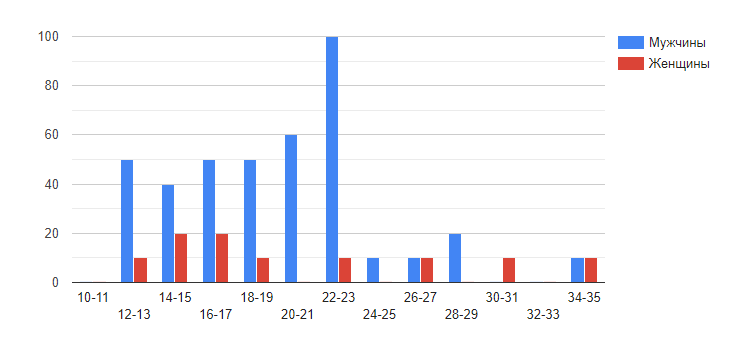


Рисунок 1.9 — Результаты ответа на вопрос «Играете ли вы в игры в жанре платформер?»

Возраст людей, прошедших опрос, был от 10 до 35 лет.

Анализ разных возрастных групп (см. Рисунок 1.10):

 Рисунок 1.10 — Результаты опроса с выборкой по возрасту

В результате анализа ответов респондентов, было решено, что основная целевая аудитория игры: пользователи мужского пола в возрасте от 12 до 22 лет.

Опрос был проведён среди пользователей форума на сайте Gamedev.ru [13]

## 1.5 Анализ возможных видов монетизации

Ниже будут описаны самые распространённые способы монетизации игр

1. **Разместить игру в Google Play или ином похожем сервисе и сделать её платной.** Это позволит получать прибыль при каждом скачивании, однако игра должна быть высокого качества, чтобы её действительно покупали. В рамках выполнения дипломной работы была произведена попытка разместить созданные игры как независимый разработчик, однако это требует разового взноса в 25$ (оформление подписки разработчика), что является неприемлемой тратой для студента, выполняющего эту работу.
2. **Реклама.** Является, пожалуй, самым известным и распространённым видом монетизации среди инди-разработчиков. Если обобщить – при определённых условиях выскакивает реклама, которую можно или нельзя пропустить. Реклама уместна во многих играх, однако, если её будет слишком много, игроки уйдут из игры. Для её внедрения и корректной также необходима подписка разработчика, также не представляется возможной реализация.
3. **Продажа внутриигровых предметов за реальные деньги (или же – донат).** Это могут быть как косметические предметы (например, скин на персонажа или оружие), так и сущности, дающие игрокам новые игровые возможности (например, новый играбельный персонаж с уникальными способностями, или какой-нибудь крайне полезный предмет). Очень популярная форма монетизации в играх, распространяющихся по модели Free To Play (условно-бесплатные игры). Однако, очень важно сделать платные предметы действительно желанными для игрока, не навязывать это и сделать необязательным для прохождения основного контента. Внедряется, как правило, опытными разработчиками в составе как минимум небольших студий.

ГЛАВА 2. РАЗРАБОТКА ИГРЫ FLAPPY BIRD SIMULATOR

## 2.1 Конкретное описание игры

На основе вышеописанного анализа было решено создать игру для Android под названием Flappy Bird Simulator. Это игра-платформер для мобильных устройств, в которой игрок с помощью касаний экрана должен контролировать полёт птицы между рядами зелёных труб, не задевая их.

Задачи:

* Реализовать главное меню с названием игры (Flappy Bird Simulator) и с кнопками Play (при её нажатии игра начинается) и Quit (при её нажатии происходит выход из игры)
* Реализовать игровой уровень, а именно:
  + Нижнюю границу уровня, пол (когда птичка касается её, она умирает)
  + Движение птицы (прыжок, гравитация, движение вправо)
  + Столкновение объектов друг с другом (трубы и пол не сталкиваются друг с другом, а птичка с трубами или полом – сталкивается)
  + Генерация новых препятствий и удаления старых, а также увеличение сложности по мере продвижения по уровню
  + Появление новых облаков и удаление старых
  + Смерть птички (когда она сталкивается с трубой или полом, она умирает, и появляется меню с надписью Game Over, рекордом и текущим результатом, а также кнопки Retry (при нажатии на неё уровень перезапускается) и Main Menu (при нажатии на неё происходит возврат на главное меню)

## 2.2 Технологическая часть

1.Ресурсы разработки

Игра была разработана с помощью туториала CodeMonkey [4], Unity, Visual Studio.

2.Описание системных требований

Игра запустится на любом современном компьютере или Android-е

3. Описание пользовательского интерфейса

В главном меню игры есть два пункта: Play и Quit. Play позволяет начать игру, Quit – выйти из неё. После нажатия на кнопку Start игра сразу же начинается. В игре вы можете нажать Space, чтобы прыгнуть, и Esc, чтобы выйти в главное меню. Также в игре есть интерфейс, показывающий результаты и текущий рекорд (он сохраняется при выходе из игры) (см. рисунок 2.1)



Рисунок 2.1 – Результат и рекорд

Также в игре есть интерфейсное меню, появляющееся после смерти персонажа (см рисунок 2.2), с вариантами:

1)Retry (перезапуск уровня)

2)Main menu (вернуться на главное меню)



Рисунок 2.2 – Меню, появляющееся после смерти персонажа

4. Руководство к использованию и эксплуатации

Игру можно запустить с помощью ярлыка на телефоне (см рисунок 2.3)



Рисунок 2.3 - Ярлык игры

После запуска файла вы увидите окно (см рисунок 2.4)



Рисунок 2.4 - Меню игры

5. Результаты экспериментальной проверки

При проверке проекта никаких отклонений от поставленной задачи не наблюдалось. В целом игра готова к эксплуатации. В результате экспериментальной проверки никаких ошибок выявлено не было. Игра работает без сбоев и нарушений.

## 2.3 Базовая настройка

Для того, чтобы создать игровой проект, необходимо: находясь в главном окне Unity, создать новый проект, назвать его Flappy Bird Simulator (название игры), из Templates выбрать 2D, выбрать местоположение и нажать Create (см. Рисунок 2.5)

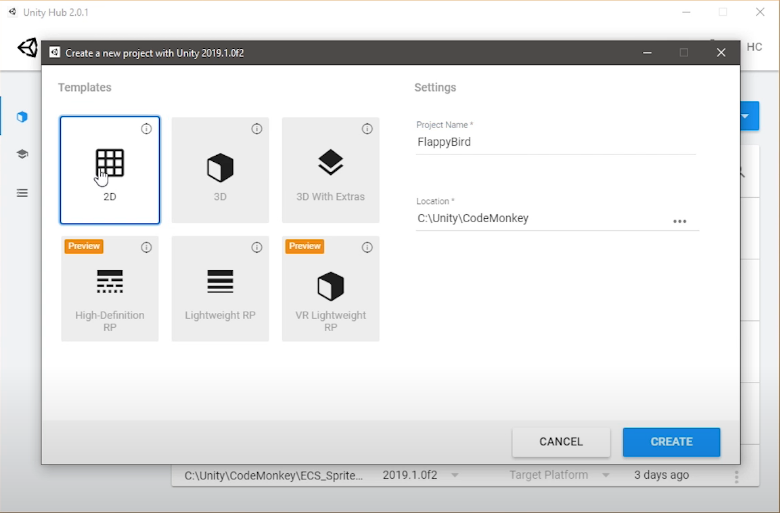


Рисунок 2.5 – Создание нового проекта в Unity

Далее был создан объект GameHandler и к нему был привязан скрипт GameHandler.cs (cм. Приложение А1), отвечающий за отладочную загрузку в начале

Далее была выполнены импорт и настройка текстур объектов, использующихся в игре (см. Рисунок 2.6)

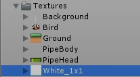


Рисунок 2.6 – Текстуры объектов, использующихся в игре

Также на GameScene был добавлен спрайт птицы (см. Рисунок 2.7)



Рисунок 2.7 – Спрайт птицы

Далее был создан скрипт GameAssets.cs (см. Приложение А2) для хранения всех объектов для лёгкого доступа, привязанный к новому пустому объекту GameAssets

## 2.4 Управление птицей, её физика, анимация

Был создан и привязан к объекту Bird скрипт управления и физики птицы Bird.cs, отвечающий за перемещение птицы, физику и логику столкновений (если птица сталкивается с полом или препятствием, то появляется интерфейс Game Over, с результатом и меню с вариантами: 1) Restart, 2) Return to the main menu:

Также был разработан скрипт SpriteAnimator.cs, отвечающий за анимацию птицы

## 2.5 Генерация уровня, сложность

Был создан объект Level в GameScene и к нему был привязан скрипт Level.cs, отвечающий за генерацию новых препятствий, облаков в поле зрения камеры, а также удаление тех, что находятся вне поля зрения камеры

## 2.6 Интерфейс в игре

В редакторе Unity в GameScene были созданы следующие объекты (см. рисунки 2.8, 2.9, 2.10)

На объект ScoreWindow был наложен скрипт ScoreWindow.cs, который отвечает за изменение числа очков вверху экрана, а если оно становится выше, чем HIGHSCORE, то HIGHSCORE меняется после смерти персонажа на это число.

Также был создан файл Score.cs, который содержит основную логику ScoreWindow.cs

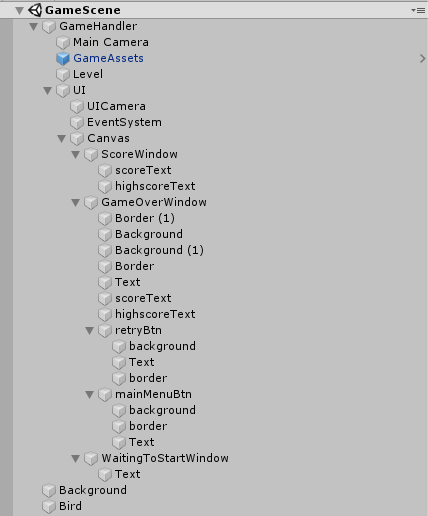


Рисунок 2.8 – Начальные игровые объекты

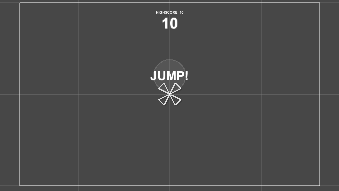


Рисунок 2.9 – Интерфейс игрового уровня

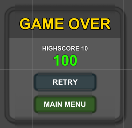


Рисунок 2.10 – Экран смерти

## 2.7 Экран загрузки

Была создана сцена Loading (для того, чтобы было видно, когда игра грузится) (см. Рисунок 2.11)

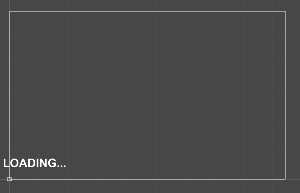


Рисунок 2.11 – Экран загрузки

Далее были разработаны скрипты Loader.cs и LoaderUpdate.cs, отвечающие за логику экрана загрузки

Скрипт LoaderUpdate.cs был привязан к объекту LoaderUpdate.cs

## 2.8 Главное меню

Была создана сцена MainMenu (см. Рисунок 2.12)



Рисунок 2.12 – Главное меню

К объекту MainMenuWindow был привязан скрипт MainMenuWindow.cs, отвечающий за логику главного меню (если нажимаешь Play, игра начинается, если Quit, происходит выход из игры)

## 2.9 Звук

Также был создан скрипт SoundManager.cs, который отвечает за весь звук в игре

## 2.10 Ожидание игрока

Для большего удобства был добавлен экран ожидания игрока и был разработан скрипт WaitingToStartWindow.cs (время в игре как бы остановлено, пока игрок не нажмёт кнопку прыжка

## 2.11 Диаграмма классов

С помощью средств Visual Studio была получена диаграмма классов приложения (см. Рисунок 2.13)

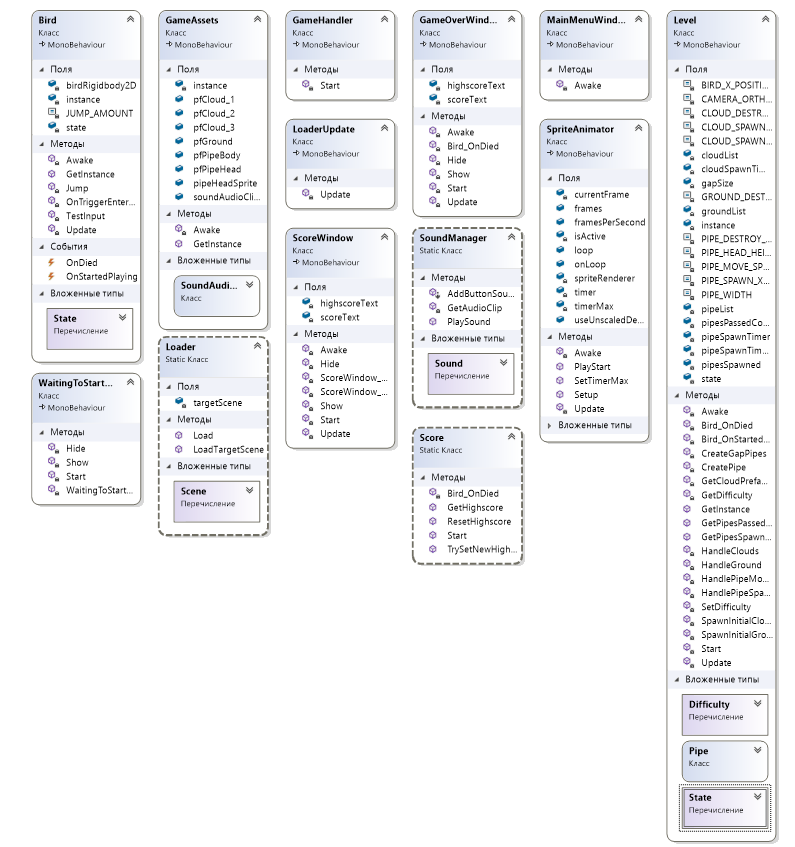


Рис. 2.13 – Диаграмма классов

## 2.12 Итоговый результат

Вот как выглядит итоговый продукт (см. Рисунки 2.14, 2.15, 2.16):



Рисунок 2.14 – Меню игры

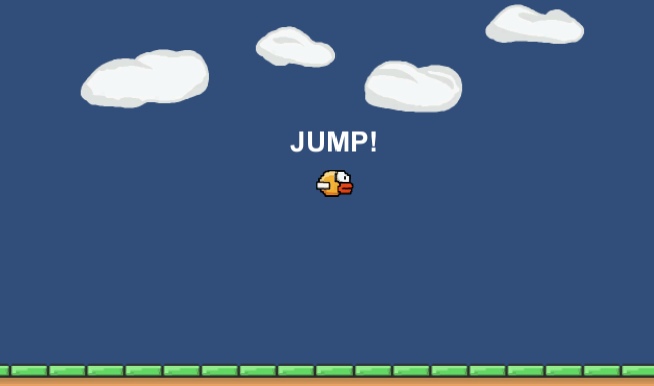


Рисунок 2.15 – Начало игры (необходимо нажать на экран)

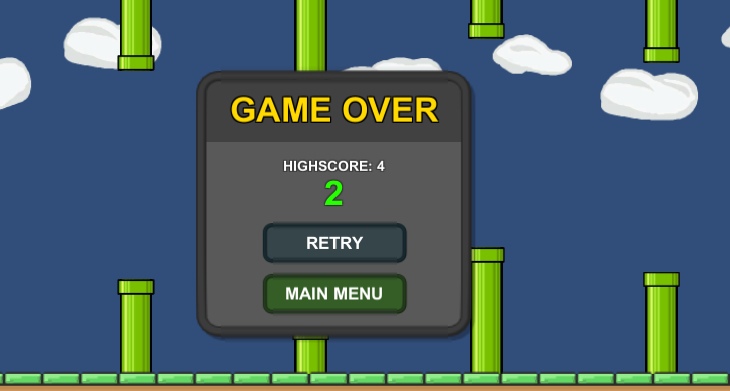


Рисунок 2.16 – Конец игры и повтор или возвращение в главное меню

## 2.13 Тестирование игры

Тестирование на совместимость позволяет проверить продукт на компьютерах с разными техническими характеристиками, разными операционными системами, с различным программным обеспечением, работающим параллельно с запущенной игрой.

Во время тестирования было проверено, насколько стабильно идёт игра (количество кадров), были ли какие-то ошибки.

Тестирование проводилось с помощью приложения Unity Remote, на устройстве Mi A2 Lite. Игра выдаёт стабильные 30 кадров в секунду, багов нет, все механики работают.

Людьми, тестирующими игру, были: мои папа, мама, а также большая часть моей группы. В целом, они остались довольны игрой и не высказали критических претензий.

## 2.13 Дальнейшие возможности развития игры

В дальнейшем можно монетизировать игру, сейчас из-за причин, описанных в пункте 1.5 Главы 1, сделать это затруднительно. Однако, если будут найдены инвестиции, все описанные виды монетизации возможны, а продажу предметов за реальные деньги (третий вид монетизации) можно реализовать через покупку энергии, необходимой для игры.

С точки зрения функциональности, на взгляд автора, ничего добавлять не требуется.

ГЛАВА 3. РАЗРАБОТКА ИГРЫ ABOVETALE

## 3.1 Конкретное описание игры

На основе вышеописанного анализа было решено создать игру для Android под названием AboveTale. Это игра-платформер для мобильных устройств, в которой игрок c помощью специального управления должен исследовать игровой мир, выполнять задания, связанные цельным сюжетом, и решать головоломки. В рамках данной работы в виду огромных масштабов задумки, было решено реализовать демоверсию игры, а также проанализировать, как именно в будущем её можно улучшить.

Задачи:

* Реализовать главное меню с названием игры (Android) и с кнопками New Game (при её нажатии игра начинается) и Quit (при её нажатии происходит выход из игры)
* Реализовать первый уровень, а именно:
  + Нижнюю границу уровня, пол (персонаж может по ней ходить)
  + Движение персонажа (прыжок, гравитация, движение вправо, влево, анимации)
  + Задний фон, представляющий собой набор из нескольких слоёв, перемещающихся вследствие движений игрока (эффект параллакса)
  + Систему инвентаря, добавления и удаления его элементов, а также систему квестов, завязанных на этих механиках (будет специальный NPC, давая предметы которому, можно получить предмет для продвижения вперёд)
  + «Умную» камеру, которая следует за игроком и не выходит за пределы уровня
  + Смену сцен в пределах одного уровня
  + Вторая сцена должна содержать область для перехода на второй уровень
  + Систему диалогов с NPC как альтернативный способ пояснения задачи игроку.
* Реализовать второй уровень, а именно:
  + Небольшой домик изнутри
  + Область для возвращения на первый уровень

## 3.2 Технологическая часть

1.Ресурсы разработки

Игра была разработана полностью с нуля с использованием Unity, спрайты были нарисованы с помощью Adobe Photoshop.

2.Описание системных требований

Игра запустится на любом современном компьютере или Android-е

3. Описание пользовательского интерфейса

В главном меню игры есть два пункта: New Game и Quit. New Game позволяет начать игру, Quit – выйти из неё. После нажатия на кнопку New Game игра сразу же начинается. В игре вы можете использовать джойстик, чтобы двигаться влево-вправо, а также кнопку прыжка, чтобы, очевидно, прыгать, есть кнопка, позволяющая выйти в главное меню. Также в игре есть интерфейс (Кнопка «Рюкзак», при нажатии на которую появляются слоты Инвентаря, каждый и которых показывает предмет, содержащийся в нём, а также под каждым слотом есть кнопка «Выбросить» (см. Рисунок 3.1)

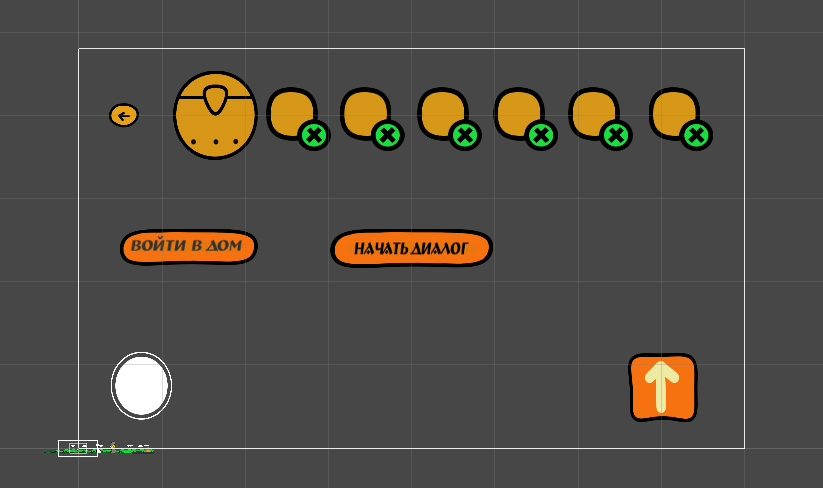


Рисунок 3.1 – Весь интерфейс

4. Руководство к использованию и эксплуатации

Игру можно запустить с помощью ярлыка на телефоне (см. Рисунок 3.2)



Рисунок 3.2 - Ярлык игры

После запуска файла вы увидите окно (см. Рисунок 3.3)

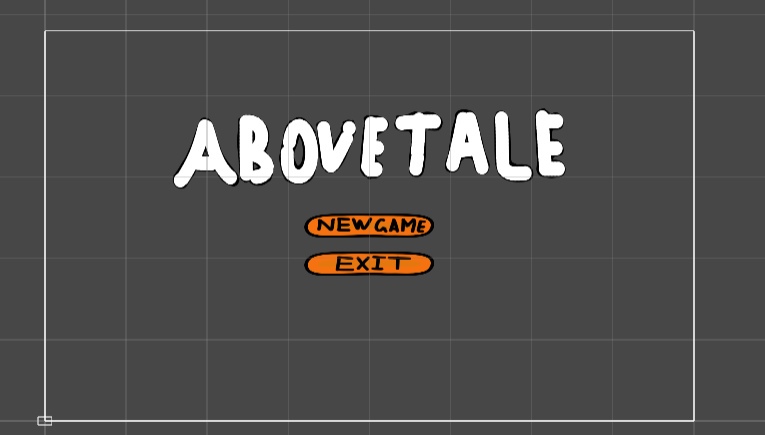


Рисунок 3.3 - Меню игры

5. Результаты экспериментальной проверки

При проверке проекта никаких отклонений от поставленной задачи не наблюдалось. В целом игра готова к эксплуатации. В результате экспериментальной проверки никаких ошибок выявлено не было. Игра работает без сбоев и нарушений.

## 3.3 Введение

Так как базовый процесс создания был разобран в предыдущей главе, в этой главе этому внимание уделяться не будет.

## 3.4 Игровой уровень и движение персонажа

Для создания всех спрайтов в игре была использована программа Adobe Photoshop 2020, а также скрипт Export Layers To Files для удобного превращения файлов в спрайты

Демонстрация части первого уровня (см. Рисунок 3.1):

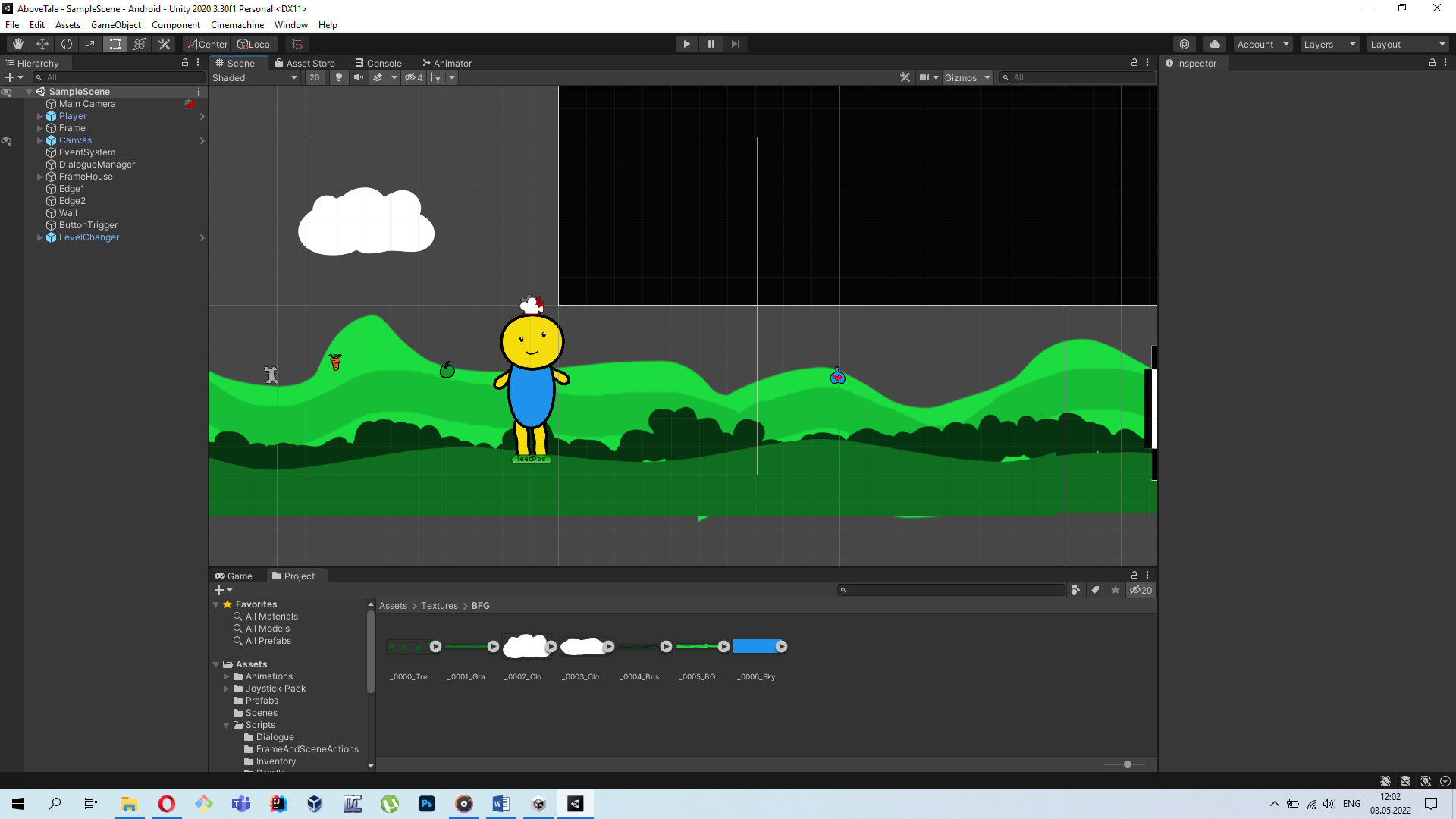


Рисунок 3.4 – Часть сцены с вручную нарисованными спрайтами

Задний и передний фон, а также управляемый игроком персонаж, являются объектами, состоящими из нескольких отдельных спрайтов (данная особенность необходима для анимаций и эффекта параллакса).

Объекту Player (см. Рисунок 3.5) были добавлены: объект Rigidbody 2D для реализации гравитации, Capsule Collider 2D для хитбокса. Объекту Grass (см. Рисунок 3.6) также были добавлен хитбокс, чтобы управляемый персонаж не проваливался сквозь пол.



Рисунок 3.5 – Иерархия объекта Player

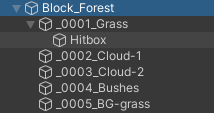


Рисунок 3.6 – Иерархия объекта Block\_Forest (первый игровой уровень)

Далее был создан скрипт PlayerController.cs, отвечающий за движение игрока. Следует упомянуть, что у игрока есть возможность прыгать, если объект feetPos, являющийся частью игрока, соприкасается с поверхностью на слое Ground (эта настройка необходима, чтобы игрок прыгал только один раз и не прыгал в воздухе)

## 3.5 Анимации персонажа (бездействие, ходьба, прыжок)

Для объекта Player были созданы следующие анимации: idle, jump, takeOf, jump, land, run (см. Рисунок 3.7)



Рисунок 3.7 – Анимация idle

В случае объекта Player вся суть анимаций состоит в том, что при их создании за несколько десятков кадров разные части тела игрока перемещаются или изменяются в размере. А благодаря инструменту Animator настраивается логика проигрывания этих анимаций (в каком порядке они проигрываются, при каких условиях, каково время выполнения и т.д.) (см. Рисунок 3.8)

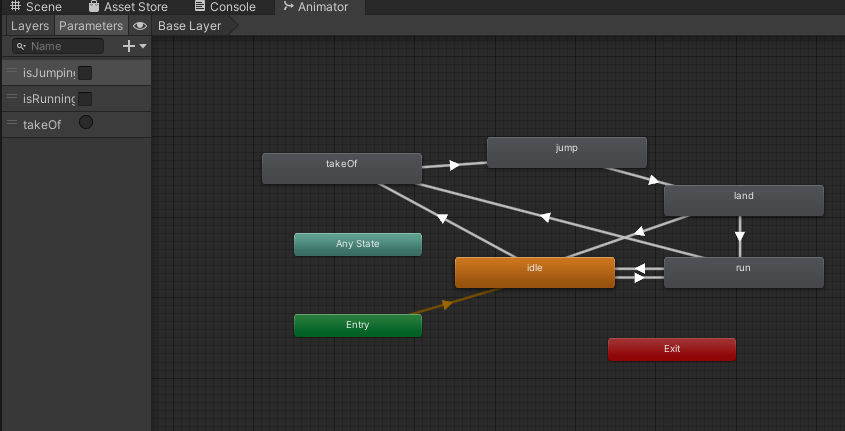


Рисунок 3.8 – Инструмент Animator

Также для анимации был дополнен скрипт PlayerController.cs. Если обобщить суть, после изменения при прыжке и беге переменные isRunning и takeOf меняют своё значение, и это приводит к последовательной смене анимаций.

## 3.6 Движение камеры и параллакс эффект

Далее при разработке возникла следующая проблема: персонаж двигается по локации с анимациями, но камера не следует за ним, а фон выглядит, как однородная статичная картинка, а так быть не должно. Чтобы решить эти проблемы, во-первых, было реализовано движение камеры. Для этого был использован встроенный в Unity модуль Cinemachine. Если обобщить суть, необходимо импортировать модуль, вставить в проект улучшенную камеру из этого модуля, объединить эту камеру с нашей оригинальной с помощью настроек объектов, сделать триггер-хитбокс для всей локации, чтобы камера не выходила за границы уровня, а также написать три скрипта: ParallaxBackground.cs (привязывается к игровому уровню или к его части, в которой находятся слои, которые мы будем перемещать), ParallaxCamera.cs (привязывается к камере), ParallaxLayer.cs (привязывается к перемещаемым слоям) (см. Рисунки 3.9, 3.10, 3.11)

Параллакс-эффект состоит в том, что некоторые слои перемещаются с движениями персонажа, чтобы в игре всё выглядело естественно.

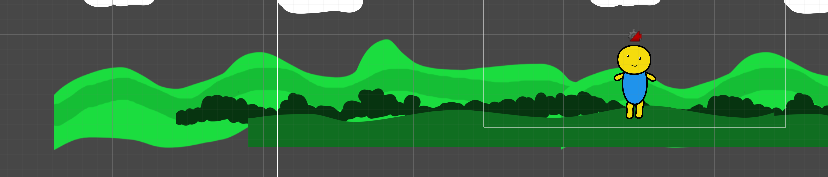


Рисунок 3.9 – Слои переместились…

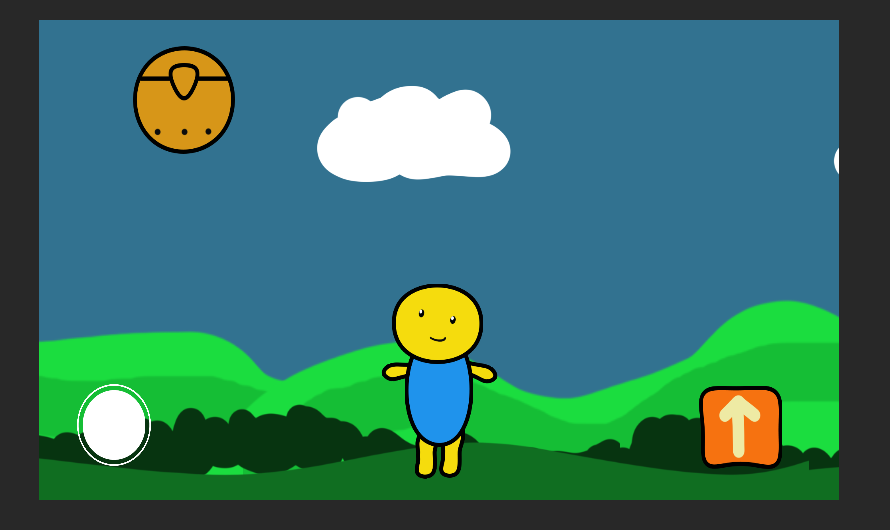


Рисунок 3.10 – Но в игре всё выглядит естественно



Рисунок 3.11 – Перемещение камеры (камера центрирована на персонаже, но когда персонаж подходит к краям уровня, камера останавливается)

## 3.7 Инвентарь, подбор и сброс предметов

Одна из главных механик игры – это подбор предметов в инвентарь и использование их для взаимодействия с NPC и другими объектами вроде стен, чтобы продвигаться дальше.

Для этого был реализован инвентарь. Были нарисованы спрайты для объектов интерфейса, связанного с инвентарём. (см. Рисунки 3.12, 3.13, 3.14)

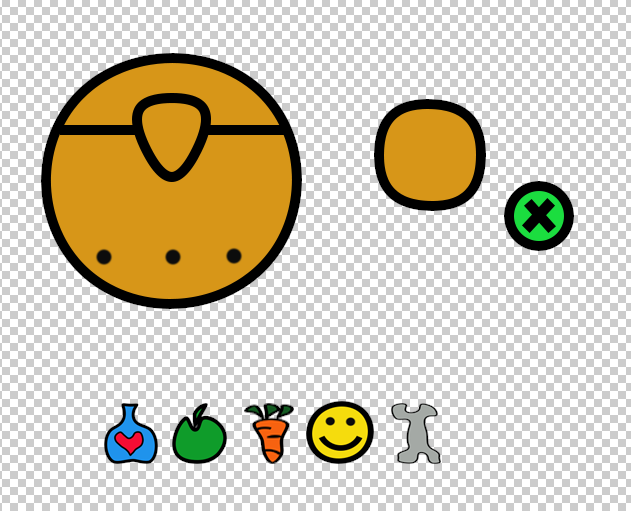


Рисунок 3.12 – Спрайты инвентаря

Также был создан объект Canvas, который и хранит в себе весь настоящий интерфейс

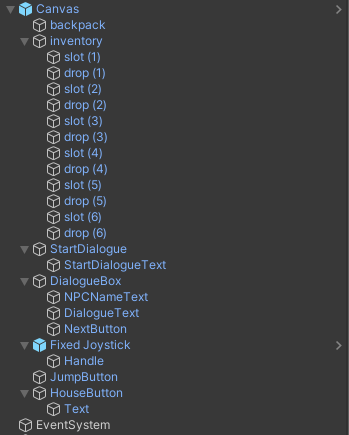


Рисунок 3.13 – Иерархия Canvas (UI)



Рисунок 3.14 – Раскрытый инвентарь

Также были реализованы 4 скрипта: Inventory (привязывается к игроку и отвечает за заполнение слота и за то, чтобы в одном слоте был один предмет), Pickup (привязывается к подбираемым предметам и отвечает за их подбор), Slot (привязывается к слотам в UI и отвечает за логику слота), Spawn (ни к чему не привязывается и хранит логику выбрасывания предмета)

Также следует отметить, что все объекты, которые может подбирать игрок, перемещены в специальную папку Prefabs, из которой они могут быть использованы на сцене в любое время со всеми свойствами, в том числе из кода.

## 3.8 Диалоги

Вторая важная механика в игре – это система диалогов с NPC (см. Рисунок 3.15)

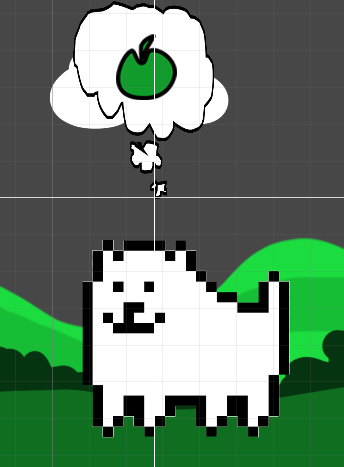


Рисунок 3.15 – NPC

Для этого, во-первых, были сделаны новые спрайты интерфейса, перенесены в сам интерфейс и заполнены надписями (см. Рисунок 3.16):

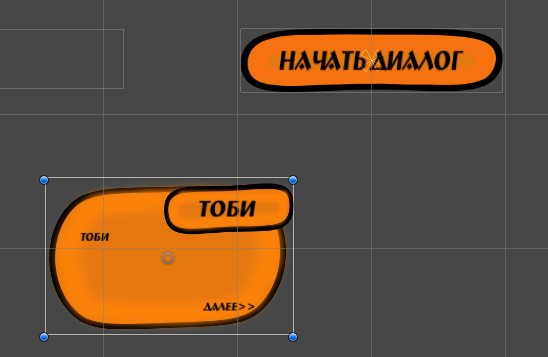


Рисунок 3.16 – Спрайты интерфейса

Далее были сделаны анимации для этих кнопок: BoxClose, BoxOpen, StartDialogueOpen, StartDialogClose. Если обобщить всю суть, если подойти к NPC, то появляется кнопка «Начать диалог», и появляется диалоговое окно с именем персонажа и тем, что он говорит, и при нажатии на кнопку «Далее» текст последовательно меняется на тот, что мы задали заранее в свойствах объекта StartDialog. (см. Рисунок 3.17)

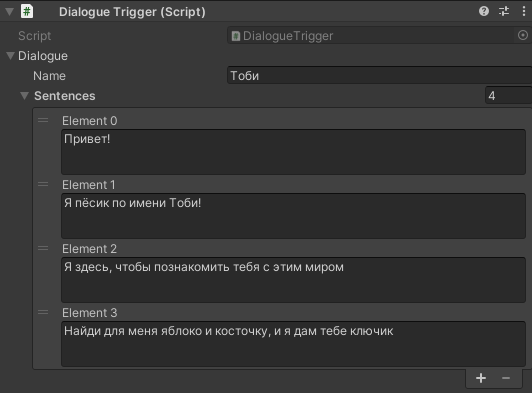
 

Рисунок 3.17 – Текст сменяется при нажатии кнопки «Далее»

Были реализованы четыре скрипта: Dialogue.cs (абстракция для DialogueTrigger), DialogueAnimator.cs (анимации диалогового окна, привязан к объекту NPC), DialogueManager.cs (см. Приложение B2) (начало диалога и смена предложений, привязан к объекту DialogueManager), DialogueTrigger.cs (привязывается к объекту StartDialogue и отвечает за появление диалогового окна)

## 3.9 Управление под Android

В пунктах 3.2 и 3.3 было реализовано движение по нажатию WASD и кнопки Space, что не подходит для управления на Android. Поэтому на UI были реализованы следующие кнопки: (см. Рисунок 3.18)

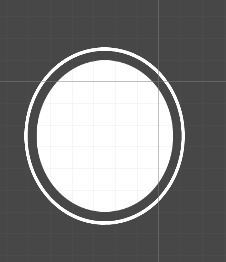


Рисунок 3.18 – Джойстик и кнопка «Прыжок»

Для создания джойстика c Asset Store был импортирован объект Joystick.

Также скрипт PlayerController.cs был отредактирован и был подключён к этим кнопкам.

## 3.10 Квесты

Главная механика AboveTale – это квесты. Если обобщить суть, они заключаются в том, что NPC последовательно получает предметы и что-то происходит. В нашем случае есть NPC по имени Тоби, который запрашивает (как в диалогах с ним, так и посредством облачка над ним) сначала яблоко (предоставление которого убирает невидимое препятствие), потом кость (даёт ключ). Также есть стена, которая не исчезнет, пока не предоставить ключ, который даёт Тоби.

Для этого были реализованы следующие объекты: QuestCloud (облака над Тоби, которые меняются по мере предоставления предметов), скрипты: Quests.cs (см. Приложение B1) (привязан к NPC и отвечает за логику скриптов), QuestCloud.cs (привязан к объекту QuestCloud и отвечает за смену облаков), QuestItemsId.cs (enum для id), QuestWall.cs (привязан к стене и отвечает за её логику).

## 3.11 Смена сцен, локаций

Также в игре были реализованы:

1.новая локация, Town (см. Рисунок 3.19)

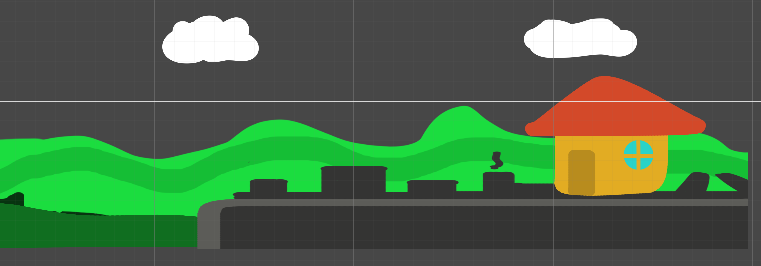


Рисунок 3.19 – Локация Town

2. переход между этой и предыдущей локацией. Для этого были созданы объекты Edge1 и Edge2, а также скрипт FrameSwitch.cs (привязан к этим объектам). Если обобщить всю суть, то при касании определённого Edge происходит переход на соответствующую локацию)

3. переход между сценой, на которой находятся две вышеупомянутые локации, и сценой внутри дома. (см. Рисунок 3.20)

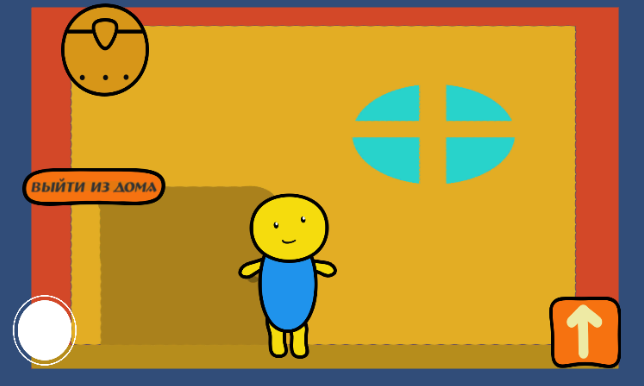


Рисунок 3.20 – Сцена HouseScene

Это было реализовано с помощью скриптов LevelChanger.cs (привязан к объекту LevelChanger.cs и отвечает за смену уровней и затемнение/освещение), ButtonTrigger.cs (привязан к кнопке выхода/входа в дом) и объекта PlayerPosition (для сохранения позиции игрока при входе/выходе)

4. сцена для главного меню (см. Рисунок 3.21) c кнопками начала и выхода из неё, а также кнопка возврата на сцену:

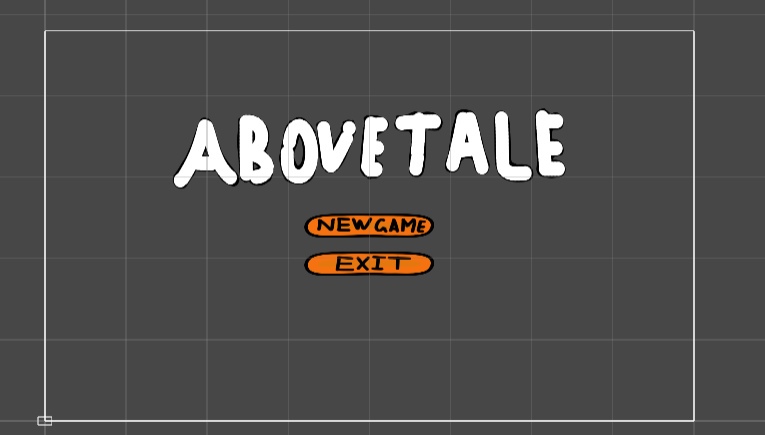


Рисунок 3.21 – Сцена Menu

## 3.12 Работа со звуком

Для каждого уровня (меню, первый и второй) была создана отдельная музыкальная тема (с помощью программы FL Studio 20) (см. Рисунок 3.22)

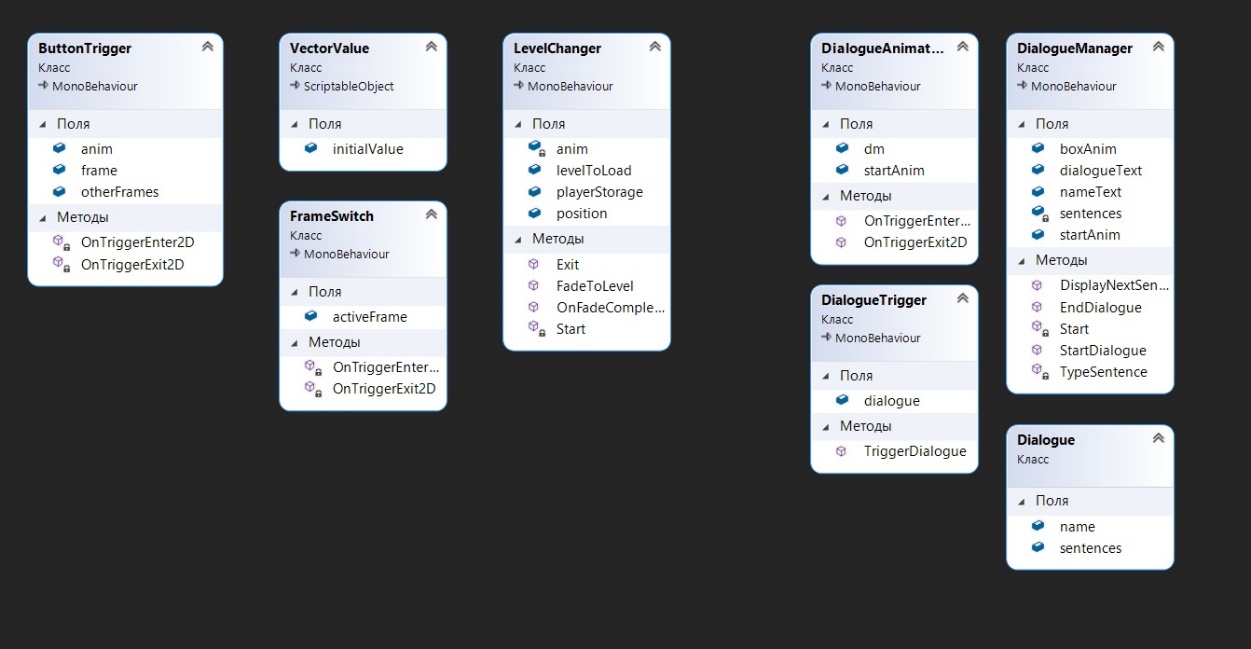


Рисунок 3.22 – Once Upon A Time (тема главного меню)

В целом, процесс создания семплов в FL Studio 20 можно описать следующим образом: берутся разные простые семплы из других источников из Интернета (например, с сайта FreeSound) или создаются сами, потом эти семплы редактируются програмно, помещаются на разные дорожки и получившееся сливается в один семпл.

## 3.13 Диаграмма классов

Диаграмма классов, созданная средствами Visual Studio, выглядит следующим образом (см. Рисунок 3.23)



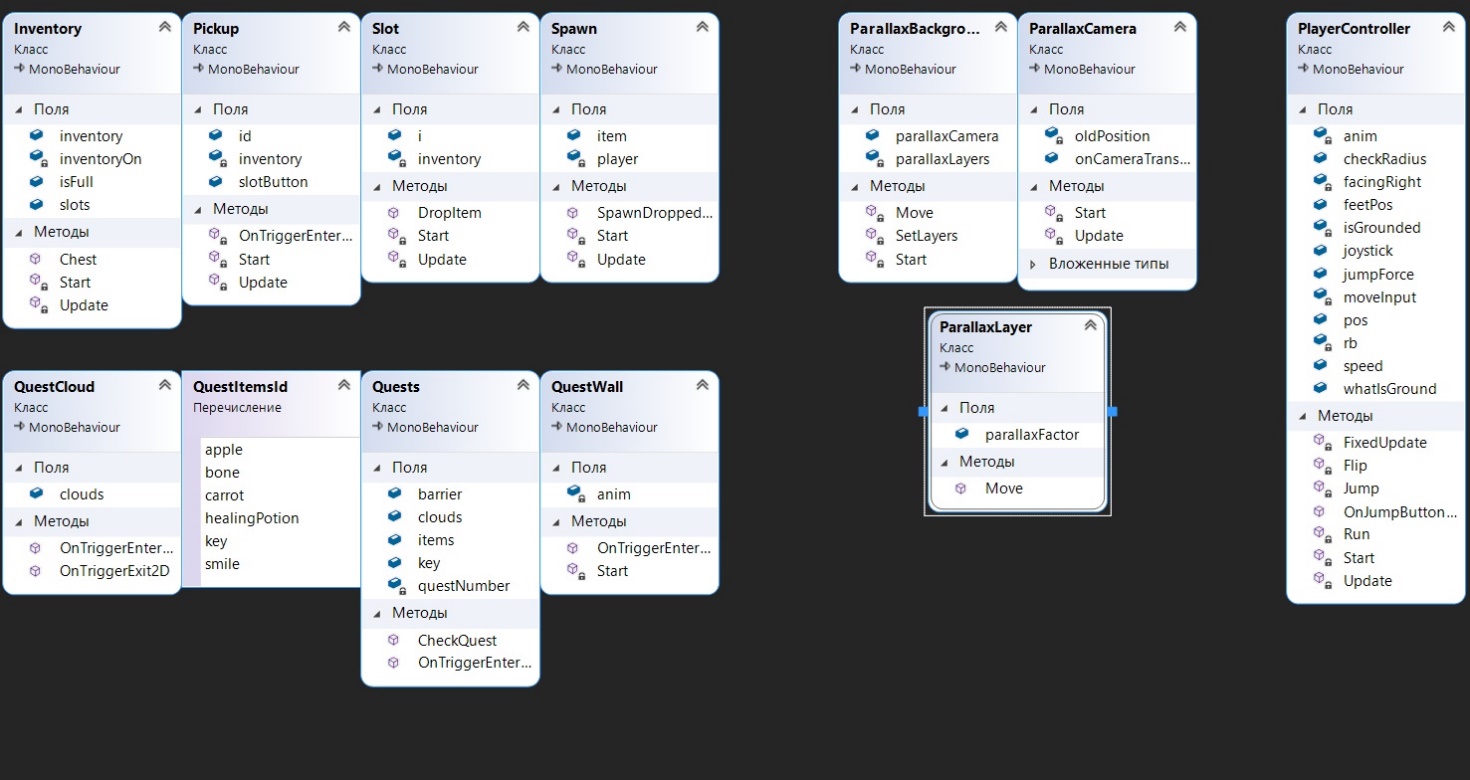


Рисунок 3.23 – Диаграмма классов

## 3.14 Иерархия сцен

HouseScene: SampleScene: MenuScene:

(см. Рисунок 3.24)

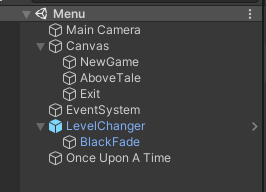
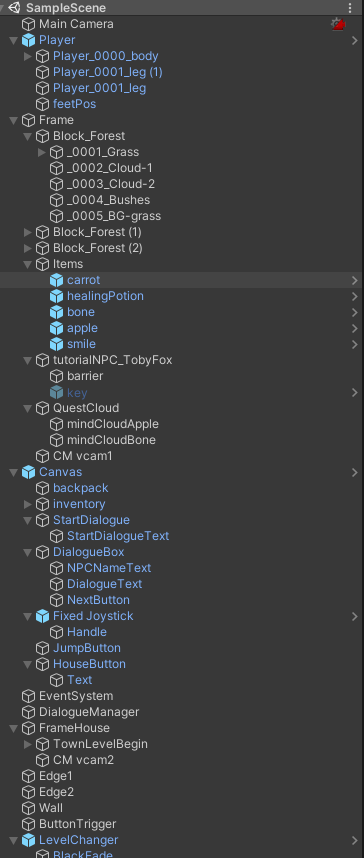
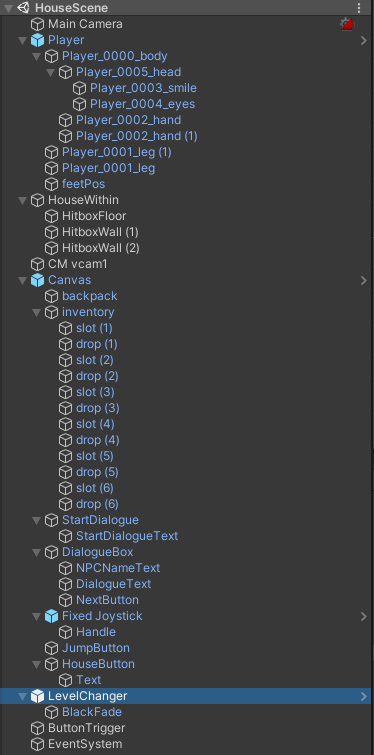


Рисунок 3.24 – Иерархия объектов Сцен

## 3.15 Тестирование игры

Для того, чтобы разместить игру на Google Play, необходимо единоразово заплатить 25$, поэтому была выполнена ручная сборка на Android с помощью средств Unity.

Тестирование на совместимость позволяет проверить продукт на компьютерах с разными техническими характеристиками, разными операционными системами, с различным программным обеспечением, работающим параллельно с запущенной игрой.

Во время тестирования было проверено, насколько стабильно идёт игра (количество кадров), были ли какие-то ошибки.

Тестирование проводилось с помощью приложения Unity Remote, на устройстве Mi A2 Lite. Игра выдаёт стабильные 30 кадров в секунду, багов нет, все механики работают.

Людьми, тестирующими игру, были: мои папа, мама, а также большая часть моей группы. В целом, они остались довольны игрой и не высказали критических претензий.

## 3.16 Дальнейшие возможности развития игры

В дальнейшем можно монетизировать игру, сейчас из-за причин, описанных в пункте 1.5 Главы 1, сделать это затруднительно. Однако, если будут найдены инвестиции, все описанные виды монетизации возможны, а продажу предметов за реальные деньги (третий вид монетизации) можно реализовать через покупку энергии, необходимой для игры.

С точки зрения функциональности, необходимо развить такие механики, как сохранение состояния уровней после перехода на другой и возвращения обратно, разнообразить диалоговую систему (добавить варианты диалога и разные ответы у НПС для них), добавить миниигры или говоломки для разнообразия игрового процесса, добавить сюжет, а также боссов вместе с их уникальными музыкальными темами.

ЗАКЛЮЧЕНИЕ

В настоящей дипломной работе были (-а):

* Проанализированы разные gameEngine и выбран конкретный для разработки игр
* Проанализированы разные жанры мобильных игр и выбраны конкретные для разрабатываемых игр
* Проанализированы особенности выбранных жанров и классических представителей этих жанров.
* Проведены исследования и даны характеристики целевой аудитории игр в выбранных жанрах
* Разработаны игры в выбранных жанрах
* Проведено тестирование разработанных игр

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. CRYENGINE | The complete solution for next generation game development by Crytek [Электронный ресурс]. – Режим доступа: https://www.cryengine.com –Дата доступа: 20.03.2022.
2. Unity [Электронный ресурс] – Режим доступа: https://unity3d.com/ru –Дата доступа: 20.03.2022.
3. Unity Public Relations [Электронный ресурс] – Режим доступа:https://unity3d.com/ru/public-relations – Дата доступа: 20.03.2022.
4. What is Unreal Engine 4 [Электронный ресурс] – Режим доступа: <https://www.unrealengine.com/en-US/what-is-unreal-engine-4> – Дата доступа: 20.03.2022.
5. Официальный сайт Google Play. [Электронный ресурс] – Режим доступа: https://play.google.com/store/apps/details?id=com.halfbrick.jetpackjoyride – Дата доступа: 20.03.2022.
6. Официальный сайт Google Play. [Электронный ресурс] – Режим доступа: https://play.google.com/store/apps/details?id=com.DefiantDev.SkiSafari – Дата доступа: 20.03.2022.
7. Официальный сайт Google Play. [Электронный ресурс] – Режим доступа: https://play.google.com/store/apps/details?id=com.kiloo.subwaysurf – Дата доступа: 20.03.2022.
8. Жанры игр — какие они бывают [Электронный ресурс]. – Режим доступа:<https://dmagame.ru/zhanry-igr-kakie-oni-byvajut/#:~:text=Жанры%20игр%20—%20какие%20они%20бывают> – Дата доступа: 20.03.2022
9. Making Flappy Bird in Unity 2019 [Электронный ресурс]. – 2019. – Режим доступа: <https://unitycodemonkey.com/video.php?v=b5Wpni9KPik> – Дата доступа: 20.03.2022.
10. Super Meat Boy [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Super_Meat_Boy> – Дата доступа: 24.05.2022.
11. Hollow Knight [Электронный ресурс]. – Режим доступа: <https://hollowknight.fandom.com/ru/wiki/Hollow_Knight> – Дата доступа: 24.05.2022.
12. Ori and The Blind Forest [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Ori_and_the_Blind_Forest> – Дата доступа: 24.05.2022.
13. Насколько велика аудитория платформеров и аркад [Электронный ресурс]. – Режим доступа: <https://gamedev.ru/flame/forum/?id=174431&page=3> – Дата доступа: 24.05.2022.

# ПРИЛОЖЕНИЕ А

**Код программы**

## А1 Скрипт GameHandler.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using CodeMonkey;

using CodeMonkey.Utils;

public class GameHandler : MonoBehaviour {

private void Start() {

Debug.Log("GameHandler.Start");

Score.Start();

}

}

## А2 Скрипт GameAssets.cs

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GameAssets : MonoBehaviour {

private static GameAssets instance;

public static GameAssets GetInstance() {

return instance;

}

private void Awake() {

instance = this;

}

public Sprite pipeHeadSprite;

public Transform pfPipeHead;

public Transform pfPipeBody;

public Transform pfGround;

public Transform pfCloud\_1;

public Transform pfCloud\_2;

public Transform pfCloud\_3;

public SoundAudioClip[] soundAudioClipArray;

[Serializable]

public class SoundAudioClip {

public SoundManager.Sound sound;

public AudioClip audioClip;

}

}

# ПРИЛОЖЕНИЕ B

**Код программы**

## B1 Скрипт Quests.cs

public class Quests : MonoBehaviour

{

private int questNumber;

public int[] items;

public GameObject[] clouds;

public GameObject barrier;

public GameObject key;

public void OnTriggerEnter2D(Collider2D other)

{

if (other.tag!= "Player" && other.gameObject.GetComponent<Pickup>().id == items[questNumber])

{

questNumber++;

Destroy(other.gameObject);

CheckQuest();

}

}

public void CheckQuest()

{

for (int i = 0; i < clouds.Length; i++)

{

if(i == questNumber)

{

clouds[i].SetActive(true);

}

else

{

clouds[i].SetActive(false);

}

}

if(questNumber == (int)QuestItemsId.apple + 1)

{

barrier.SetActive(false);

}

if (questNumber == (int)QuestItemsId.bone + 1)

{

key.SetActive(true);

}

}

}

## B1 Скрипт DialogueManager.cs

public class DialogueManager : MonoBehaviour

{

public Text dialogueText;

public Text nameText;

public Animator boxAnim;

public Animator startAnim;

private Queue<string> sentences;

private void Start()

{

sentences = new Queue<string>();

}

public void StartDialogue(Dialogue dialogue)

{

boxAnim.SetBool("boxOpen", true);

startAnim.SetBool("startOpen", false);

nameText.text = dialogue.name;

sentences.Clear();

foreach(string sentence in dialogue.sentences)

{

sentences.Enqueue(sentence);

}

DisplayNextSentence();

}

public void DisplayNextSentence()

{

if(sentences.Count == 0)

{

EndDialogue();

return;

}

string sentence = sentences.Dequeue();

StopAllCoroutines();

StartCoroutine(TypeSentence(sentence));

}

IEnumerator TypeSentence(string sentence)

{

dialogueText.text = "";

foreach(char letter in sentence.ToCharArray())

{

dialogueText.text += letter;

yield return null;

}

}

public void EndDialogue()

{

boxAnim.SetBool("boxOpen", false);

}

}