МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка компьютерной игры в жанре «Аркада» для платформы Android

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-143.ВКР

Научный руководитель,
профессор кафедры СП, д.ф.-м.н.,
доцент
_____ Т.А. Макаровских

Автор работы,
студент группы КЭ-404
_____ Г.Ю. Сырцева

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студентке группы КЭ-404
Сырцевой Генриетте Юрьевне,
обучающейся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2023 г. № 764-13/12)

Разработка компьютерной игры в жанре «Аркада» для платформы Android.

2. **Срок сдачи студентом законченной работы:** 05.06.2024 г.

3. **Исходные данные к работе**

3.1. Unity и C#. Геймдев от идеи до реализации. [Электронный ресурс] URL: https://coollib.net/b/670499-dzheremi-gibson-bond-unity-i-cn-geymdev-ot-idei-do-realizatsii/read (дата обращения: 01.02.2024 г.).

3.2. Разработка мобильных приложений на C# для iOS и Android. [Электронный ресурс] URL: https://coollib.com/b/660136-vyacheslav-chernikov-razrabotka-mobilnyih-prilozheniy-na-cn-dlya-ios-i-android/readp?p=5&cnt=9000 (дата обращения: 01.02.2024 г.).

3.3. Изучаем C# через разработку игр на Unity. [Электронный ресурс] URL: https://coollib.net/b/673744-ferrone-harrison-izuchaem-cn-cherez-razrabotku-igr-na-unity/read (дата обращения: 01.02.2024 г.).

4. Перечень подлежащих разработке вопросов

4.1. Разработать игровой процесс игры.

4.2. Разработать дизайн приложения.

4.3. Спроектировать структуру игры под Android.

4.4. Реализовать и протестировать приложение.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
профессор кафедры СП, д.ф.-м.н., доцент Т.А. Макаровских

Задание принял к исполнению Г.Ю. Сырцева

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В настоящее время игровая индустрия находится в постоянном развитии, и с каждым годом появляются новые технологии и тренды. Однако, несмотря на все новшества, аркадные игры остаются популярными среди игроков всех возрастов. Unity – одна из самых популярных платформ для создания игр, именно здесь разрабатываются и выпускаются множество успешных проектов.

Актуальность аркадных игр на Unity заключается в их простоте, доступности и универсальности. Такие игры могут быть созданы даже небольшой командой разработчиков или даже одним человеком, что делает их разработку более экономически выгодной. При этом можно создать захватывающую анимацию и подобрать звуковые эффекты, что позволит увеличить погруженность в игровой процесс. Кроме того, аркадные игры обладают высокой степенью зависимости от геймплея, что делает их привлекательными для широкой аудитории.

Также стоит отметить, что аркадные игры на Unity могут быть успешно адаптированы для мобильных устройств, что является важным фактором в современном мире игровой индустрии. Благодаря простому управлению и небольшим требованиям к аппаратному обеспечению, такие игры могут привлечь большое количество пользователей.

Постановка задачи

Целью выпускной квалификационной работы является разработка компьютерной игры в жанре «Аркада» для платформы Android. Для достижения поставленной цели необходимо решить следующие задачи:

1) разработать игровой процесс игры;
2) разработать дизайн приложения;
3) спроектировать структуру приложения под Android;
4) реализовать и протестировать приложение.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 33 страницы, объем списка литературы – 15 источников.

В первой главе «Анализ предметной области» приводится анализ аналогичных проектов со всеми их особенностями, которые необходимо учитывать при создании нового игрового приложения. Так же рассматриваются все средства реализации проекта, и производится анализ предметной области. Есть обзор на современную литературу, которая использовалась при создании проекта.

Во второй главе «Проектирование» производится анализ требований функциональных и нефункциональных требований к системе, описывается общий сценарий игры. Приведены в качестве примера макеты игры и описан дизайн, используемые шрифты и цвета.

В третьей главе «Реализация» определены основные компоненты системы и подробно представлена их реализация. Показаны листинги, демонстрирующие управление кнопками, поведение главного игрока и платформ всех категорий сложности.

В четвертой главе «Тестирование» произведена проверка соответствия приложения всем установленным функциональным требованиям, показаны результаты функционального и юзабилити-тестирования игрового приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области

Целью данной работы является разработка компьютерной 2D игры в жанре «Аркада» на платформе Unity под Android. Компьютерные игры – программы, предназначенные в основном для обучения или развлечения пользователя. 2D (2 dimension) игры имеют плоскую графику (спрайты), а камера не имеет перспективы [1].

В процессе разработки компьютерной игры использовались некоторые книги, посвященные геймдеву, среди которых можно выделить следующие: «Разработка мобильных приложений на C# для iOS и Android», «Изучаем C# через разработку игр на Unity» и «Unity и C#/ Геймдев от идеи до реализации». Первая книга представляет собой подробное руководство по изучению языка программирования C# с использованием популярного игрового движка Unity. Автор книги, Вячеслав Черников, в сжатой форме описывает весь процесс создания приложений для смартфонов и планшетов. Объясняется шаг за шагом процесс создания игры, что является отличным ресурсом для начинающих разработчиков.

Следующая книга «Изучаем C# через разработку игр на Unity» содержит полную информацию о языке C# и всех его особенностях в данной среде. Харрисон Ферроне, автор книги, демонстрирует пример создания простого приложения. Вся информация о платформе написана доступным языком. Объясняются основы программирования, все главные концепции и лучшие практики, позволяющие повысить собственные знания в данной области и вывести их на новый уровень.

Автором книги «Unity и C#/ Геймдев от идеи до реализации» является Джереми Гибсон Бонд. Он дает необходимую информацию обо всех современных методах и профессиональных инструментах разработки дизайна игры и дальнейшего проектирования с использованием Unity и языка программирования C#. Автор книги охватывает все этапы

разработки игры, начиная от создания концепции идеи до завершения проекта. Он подробно объясняет основные концепции программирования и поэтапно помогает читателям создавать свои собственные игры. Книга полезна для тех, кто хочет узнать, как применить знания по программированию и геймдизайну для создания собственного приложения под Android.

Исследование этих источников помогает получить все необходимые знания и навыки для успешного завершения проекта по разработке игры под Android с использованием C# и Unity.

Unity – это ведущая на данный момент межплатформенная среда, предоставляющая большое количество инструментов для разработки игр. Данная платформа позволяет создать приложения под более чем 20 операционными системами, включающими персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и пр. Данный движок – это крайне производительный визуальный рабочий процесс и сильная межплатформенная поддержка. Сама разработка привязана к тщательно продуманному визуальному редактору. Он позволяет быстро и рационально создавать профессиональные игры, обеспечивая высокую продуктивность разработчиков и предоставляет в их распоряжение исчерпывающий список самых современных технологий в области видеоигр [2]. Теперь разработка собственной игры доступна каждому желающему. Большинство необходимых средств бесплатно [5]. Данная платформа была выбрана для разработки игры, так как в ней много средств, благодаря которым можно быстро создать интерфейс, разработать графику, физику, звук и дизайн. Есть встроенная поддержка 2D игр и работы с 2D компонентами [14].

Одной из главных особенностей Unity 2D является возможность создания игр без необходимости знания программирования. В среде Unity 2D есть визуальный редактор, который позволяет создавать игровые объекты, настраивать их свойства и создавать игровую логику без

написания кода.

Кроме того, Unity 2D поддерживает множество форматов файлов и имеет встроенную систему физики, что делает процесс разработки более удобным и быстрым. Unity – один из тех движков, который обеспечивает поддержку как 2D, так и 3D-игр без лишних проблем для разработчиков [15].

В качестве языка программирования был выбран C#, так как это язык со строгой типизацией, с ним гораздо сложнее допустить ошибку, так же в нем много способов сократить код, не нарушая логику программы. Код игры легко привязать к Unity, и работать через Visual Studio, параллельно отслеживая все изменения в динамике игры. C# является наиболее широко используемым и поддерживаем в Unity и позволяет большинству разработчиков применить уже имеющиеся знания [8].

В наши дни мобильные игры существуют трех различных вариантов.

1. Простая игра с тщательно подобранными взаимодействиями, графикой и ограниченной сложностью, потому что эти аспекты лучше всего соответствуют архитектуре игр.

2. Более сложные игры, доступные для широкого круга устройств, от специализированных игровых консолей до смартфонов.

3. Мобильные версии игр, первоначально создававшиеся для консолей и персональных компьютеров [7].

Игра – это попытка добиться определенного положения игроком, используя только средства, разрешенные правилами, где правила запрещают использовать более эффективные средства в пользу менее эффективных и принимаются только потому, что они делают возможной такого рода деятельность [4].

Игры жанра Аркады характеризуются относительной легкостью игрового процесса, но при этом его интенсивностью. Был выбран этот жанр, так как он сочетает простоту процесса и захватывающий сюжет. Аркады распространены среди приложений под Android, которые набирает

популярность на данный момент в сфере IT.

В данном проекте рассматривается разработка простой 2D игры под Android на платформе Unity.

1.2. Анализ аналогичных проектов

В современном мире все сильнее заметна востребованность приложений, разрабатываемых под Android. Современные телефоны имеют высокую мощность, оснащены большим объемом памяти, что позволяет разработчикам создавать все более красивые, захватывающие игры с широким функционалом. За последние 15 лет смартфоны прошли большой путь. Они превратились в по-настоящему массовый продукт, который лежит в кармане буквально каждого жителя развитой страны [3]. Разработчиком Android является Google [13].

В качестве схожих аркад можно привести «Happy Hop» (рисунок 1) и «A Jumpers Journey».

Рисунок 1 – Главное меню «Happy Hop»

Первая игра оснащена хорошей анимацией. Присутствует звуковое сопровождение. Также в игре видно максимальное количество набранных очков и текущее количество очков (рисунок 2). В игре можно отключить звуковое сопровождение, как с главного меню, так и после запуска игры.

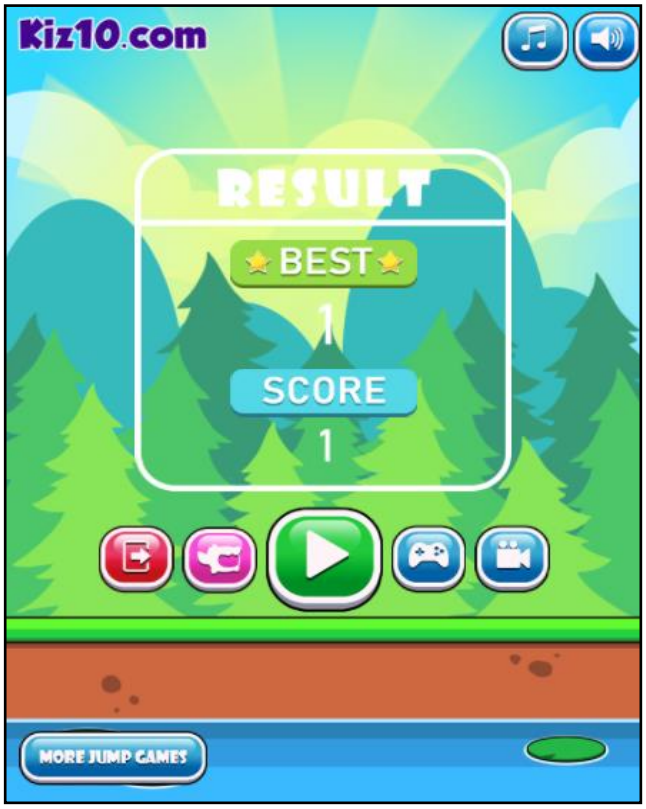

Рисунок 2 – Запись набранных очков в «Happy Hop»

Игрок перемещается по полю (рисунок 3), осуществляя прыжки по платформам. На некоторых платформах можно собирать звезды, которые дадут ему дополнительные баллы. За полученные баллы игрок может изменить внешний вид своего игрока. Так же необходимо время от времени пополнять здоровье своего персонажа. В нижней части видна полоса здоровья. Она уменьшается, до осуществления успешного попадания на платформу. Сложность заключается в том, что время, за которое нужно осуществить прыжок, ограничено. Через кнопку настроек можно отключить звуковое сопровождение, продолжить текущую игру, начать игру заново или выйти на главный экран.

Рисунок 3 – Игровое поле «Happy Hop»

Суть второй игры (рисунок 4) заключается в осуществлении прыжков на соседние доступные поверхности. К кубу подключена интересная анимация.

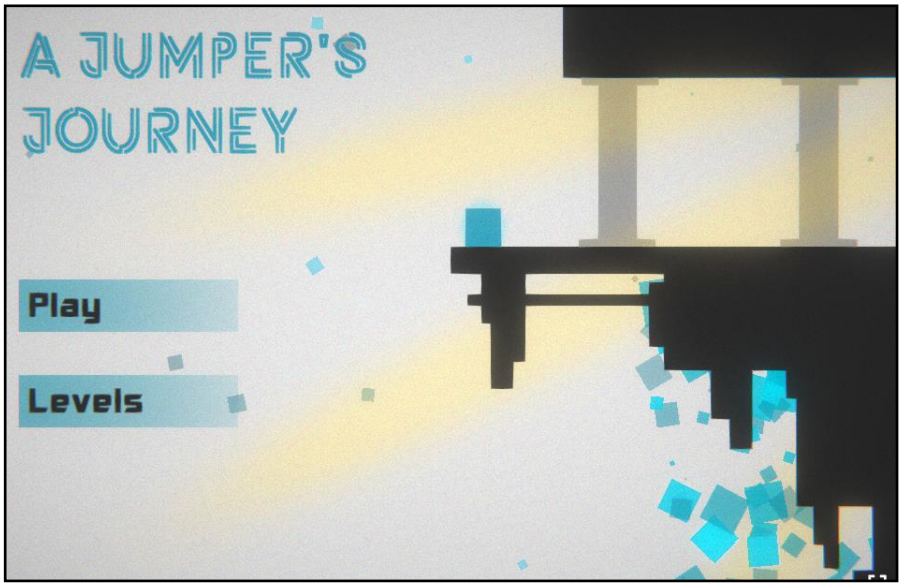

Рисунок 4 – Главное меню «A Jumpers's Journey»

Цвет куба меняется после проигрыша. Также от него исходят небольшие кубы при перемещении по игровому полю. Прыжок осуществляется в ту точку, в которую щелкает игрок. При щелчке в верхней части экрана куб осуществляет прыжок вверх.

В игре присутствуют подсказки, в которых указано, что необходимо делать, для успешного продолжения игры (рисунок 5). Задача пройти как можно больше доступных платформ и дойти до нового уровня, который обозначается желтым кубом. В игре есть шипы, которые необходимо обходить, или игра завершается.

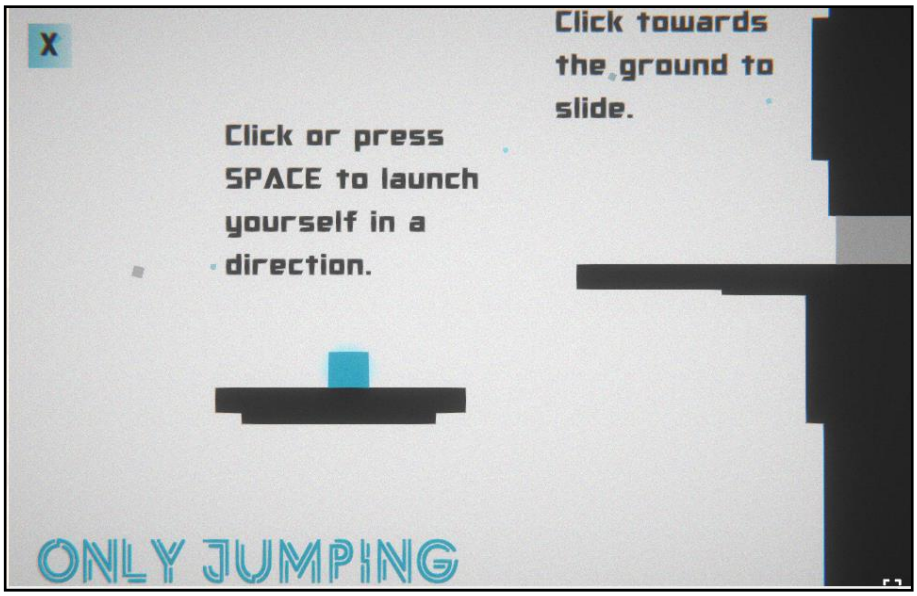

Рисунок 5 – Подсказки в «A Jumpers's Jour»

Подобраны спокойные цвета, как у фона, так и у всех надписей. При игре глаза не устают. Все выполнено в одном стиле и выглядит очень гармонично. В игре есть меню всех платформ. При прохождении уровня, они становятся доступными для запуска. Отсутствует возможность отключить звуковое сопровождение.

В обеих играх есть звуковые эффекты и простая анимация. Сам игровой процесс затягивает. Управление очень простое и интуитивно понятно любому начинающему игроку.

Перечисленные выше игры интересны, но в них не хватает самого развития сюжета по мере прохождения, что может значительно усилить

вовлеченность игрока в процесс. Большинство бонусов открыты как в начале, так и в конце. Развития не происходит, хотя оно может быть легко внедрено даже в аркадные игры, учитывая современные возможности.

В данной работе рассматривается процесс создания аркадной игры с развитием сюжета, но сохранением атрибутов данного жанра. Можно отметить, что звуковые эффекты иногда скорее создают неудобство, чем делают прохождение более захватывающим.

Жанр «Аркада» отличается простотой и в какой-то мере ненавязчивостью. Сам игровой сюжет несложный, поэтому все должно выглядеть лаконично. Излишнее количество анимаций тоже зачастую мешает погрузиться в процесс и отвлекает. Если игра простая, минимальная анимация делает ее гораздо красивее и динамичнее, а излишняя перегружает.

Выводы по первой главе

В данной главе были рассмотрены основные инструменты создания аркадных игр и выбраны наиболее эффективные из них. Также были проанализированы некоторые игры этого жанра и выявлены определенные нюансы, которые необходимо учесть при разработке проекта. Было принято решение создать простую игру с развивающимся сюжетом и отсутствием отвлекающей анимации и звуков.

2. ПРОЕКТИРОВАНИЕ

2.1. Концепция игры

На этапе проектирования самым важным является продумывание пользовательского интерфейса. Если сразу начать с дизайна, то придется постоянно переделывать его для согласования с заказчиком [11]. Ядром игры является игровая механика или правила, которые в значительной степени зависят от оболочки или тематики игры [6].

Игровое приложение реализовано с помощью средства Unity 2D. Это средство разработки игр, которое позволяет создавать двухмерные игры для различных платформ, включая ПК, мобильные устройства и консоли. Оно предоставляет широкий набор инструментов и функций, которые помогают разработчикам создавать игры быстро и эффективно. Разработчики игры должны определить четкие правила, обеспечить привлекательное художественное оформление, чтобы вызвать у игроков желание сыграть в игру [12].

Суть игры заключается в том, что игрок перемещается по полю прыжками на выезжающие платформы, при этом возможно совершить только один прыжок без перемещения платформы, на которую приземлился игрок. С каждым новым прыжком, платформа меняет свое положение, и игрок может оказаться за пределами поля. Выезжающие платформы различаются по параметрам. Они встают на разное расстояние относительно игрока. Чем меньше и дальше от игрока платформа, тем сложнее на нее попасть. Одновременно может выехать только одна. Набрать очки можно только при условии успешного приземления игрока на нее. Можно использовать небольшую хитрость, и если рядом не оказалось подходящей платформы, на которую есть возможность прыгнуть, игрок может подпрыгнуть и осуществить быстрое перемещение в тот момент, пока все платформы будут находиться в движении. Есть возможность регулировать силу прыжка. При более длительном нажатии прыжок становится сильнее. При достижении определенного количества

очков, отсчет которых показывается в верхней части экрана, игра становится сложнее и выезжающие блоки меняют свою форму.

2.2. Функциональные и нефункциональные требования

В ходе проектирования приложения были определены функциональные возможности, предоставляемые пользователю, а также нефункциональные требования к разрабатываемому приложению.

Функциональные требования к проектируемой системе:

1) игра должна выводить фразу об окончании игры;
2) игра должна выводить кнопку запуска уровня заново;
3) в игре должна быть возможность отключать звуковые эффекты через главное меню;
4) на экране должно отображаться максимальное полученное количество очков;
5) на экране должно отображаться текущее количество очков;
6) в начале игры должны появляться фразы-подсказки, информирующие о том, какие действия нужно предпринять.

Нефункциональные требования к проектируемой системе:

1) игровое приложение должно быть написано на языке C#;
2) приложение должно быть разработано на платформе Unity;
3) фразы-подсказки должны быть выполнены в одном стиле;
4) количество различных платформ по уровню сложности должно быть не менее 5;
5) в игре цвет платформ меняется;
6) выезжающие платформы должны отличаться по уровню сложности и иметь разный внешний вид;
7) в игре должны присутствовать небольшие анимации при запуске и при осуществлении прыжка;
8) игровое приложение должно быть разработано под Android.

2.3. Дизайн игры

Проекты в Unity поделены на сцены. Каждая сцена содержит коллекцию игровых объектов. Сцену можно рассматривать как один уровень в игре, но сцены так же помогают разделить игру на управляемые объекты. Например, главное меню игры обычно реализуется в виде отдельной сцены, как и каждый ее уровень [7].

При запуске игры загружается сцена главного меню (рисунок 6), с основным персонажем, находящимся на платформе, с которой осуществляется первый прыжок. Посередине экрана надпись «Начать игру», над ней располагается название игры.

Рисунок 6 – Макет главного меню

При запуске, играет анимация, и персонаж перемещается вместе с платформой на задний план (рисунок 7). Дизайн игры продуман так, чтобы все детали были видны на экране. Цвета подобраны таким образом, чтобы

глаза не уставали и не напрягались, для этого они должны быть приглушенными. В верхней части показано количество текущих очков и максимальное количество очков. В нижней части располагается кнопка отключения звуков. Выезжающие платформы выполнены в разных цветах. Все они имеют различные размеры. Изначально выезжают большие платформы. Они самые легкие. На более маленькие платформы тяжелее попасть. Они появляются после того, как игрок достигнет определенного числа очков. Группы платформ уезжают плавно за пределы экрана и меняют свое положение на одинаковое расстояние каждый раз, после осуществления прыжка. Платформы удаляются за пределами поля. Игрок тоже должен автоматически удаляться при выходе за пределы поля игры.

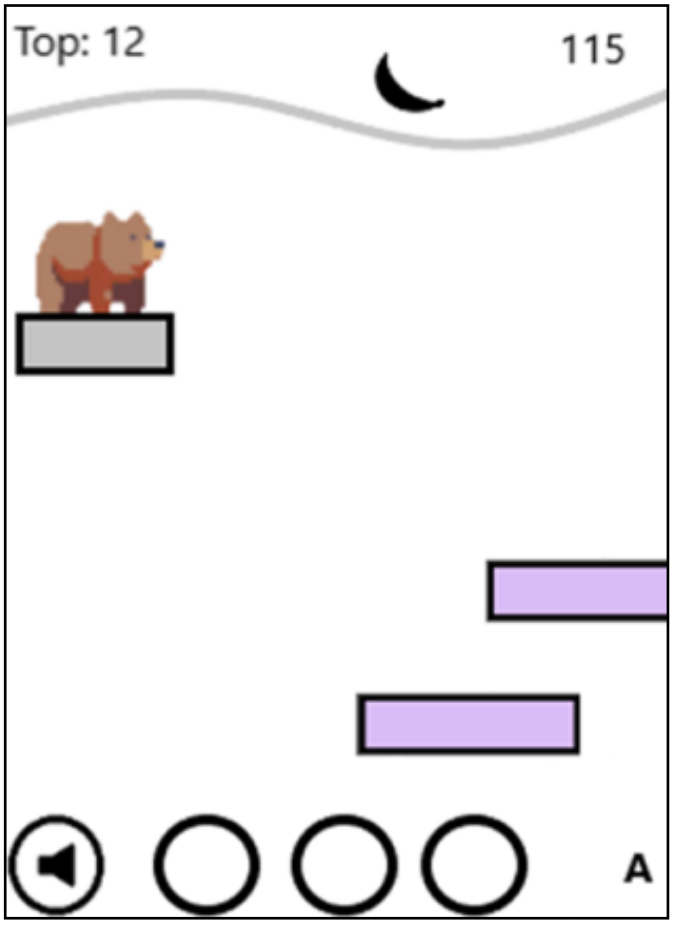

Рисунок 7 – Макет дизайна игрового поля

Под каждую платформу создан отдельный материал. Всего существует пять различных типов платформ. Каждая платформа отличается по цвету.

В игре присутствуют звуковые эффекты при успешном попадании игрока на платформу. К основному игроку должна быть подключена анимация, при осуществлении прыжка. Игрок уезжает с первой платформой на задний план, уменьшаясь в размерах. При проигрыше на экране появляется кнопка, с помощью которой можно запустить игру заново.

При наборе десяти очков, игрок автоматически переходит на следующий уровень сложности и попадание на платформы становится более сложным. Дальше его задача набрать двадцать очков. Каждый раз, когда количество будет увеличиваться на некоторое количество очков, платформы меняются визуально и по форме. Чем больше баллов набрал игрок, тем сложнее игра становится, так как платформы изменяются в размере. Цвет платформ отличается, чтобы было понятно, что уровень поменялся. Для открытия последнего вида платформ необходимо набрать сто пятьдесят баллов.

Выводы по второй главе

Во второй главе были определены функциональные и нефункциональные требования и разработан дизайн игры. Должны присутствовать анимации, подключенные к основному игроку и при запуске приложения. От выбранной цветовой палитры не должны уставать глаза. Был придуман сценарий и развитие в игре. Основная цель сделать игру простой, но с интересным дизайном. Все используемые эффекты должны быть спокойными и нейтральными, чтобы не препятствовать погружению в игровой процесс и не отвлекать.

3. РЕАЛИЗАЦИЯ

3.1. Реализация компонентов игры

Для реализации 2D игры была выбрана платформа Unity. Визуальный рабочий процесс – достаточно уникальная вещь, выделяющая Unity из большинства сред разработки. Весь рабочий процесс привязан к тщательно продуманному визуальному редактору [10]. Среда обладает всеми необходимым инструментарием для создания 2D игр, а также поддержкой широкого круга платформ, подробной официальной документацией и официальными уроками, а также материалами по разработке продуктов. Для реализации скриптов игры использовался язык программирования C#. Для написания и отладки кода использовалась интегрированная среда программирования Visual Studio Code.

Ниже (рисунок 8) представлена диаграмма компонентов игрового приложения. Данные компоненты содержат в себе классы, описывающие поведение, свойства, параметры и зависимости игровых объектов.

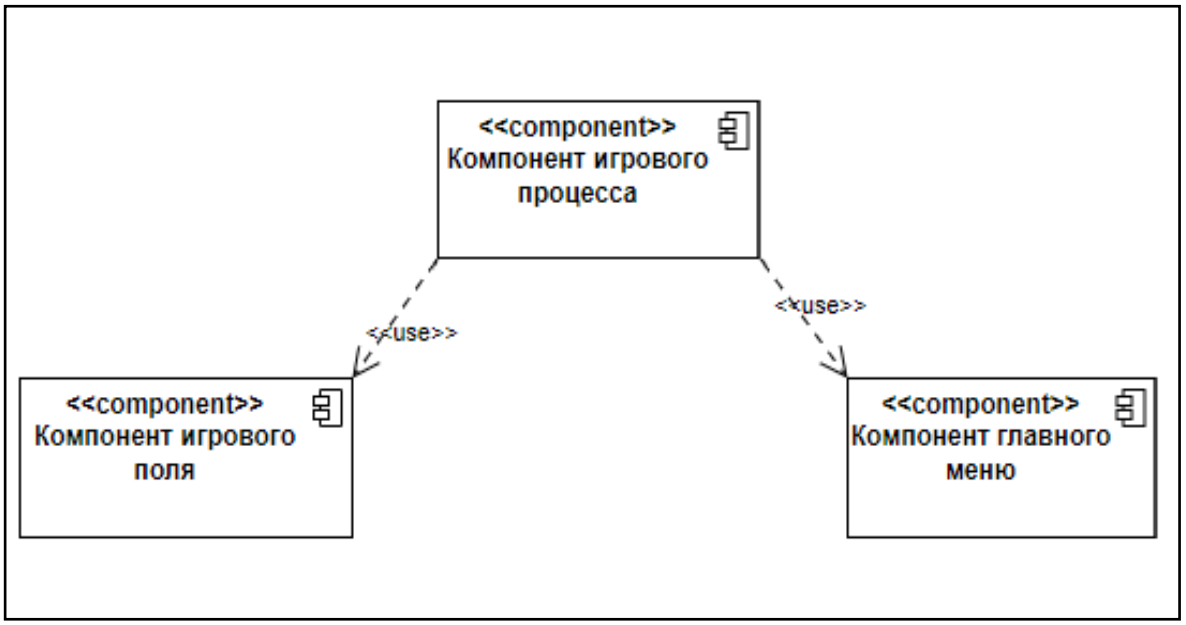

Рисунок 8 – Диаграмма компонентов

Компонент игрового процесса содержит набор классов, реализующих игровой процесс системы. Прыжки игрока, всплывание новых платформ и создание их.

Компонент главного меню содержит набор классов, представляющих главное меню и меню настроек.

Компонент игрового поля содержит набор методов, представляющих игровое поле, по которому передвигается игрок.

Интерфейс системы состоит из главного меню (рисунок 9) и игровой сцены (рисунок 10). Главное меню представлено в отдельной сцене Main, которая является начальной при запуске приложения. Оно состоит из панели, на которой есть кнопка настроек, через которую можно отключить звуковые эффекты или включить их. Скрипты разделены на две папки: Game и Main Scene. В папке Game хранится все, что относится к полю игры, управлению персонажем и платформами. В папке Main Scene находятся скрипты, которые относятся к главному меню и кнопкам.

Рисунок 9 – Панель запуска игры

На игровом поле сверху указано количеств полученных баллов. Баллы начисляются при успешном попадании на выезжающие платформы.

Рисунок 10 – Игровое поле

Также в игре присутствуют фразы-подсказки, которые смогут сориентировать игрока, что необходимо делать, для успешного продолжения игры (рисунок 11). Для названия игры был выбран шрифт SimpleSquare. Он же используется для подсчета количества набранных очков при прохождении. Для фраз-подсказок используется шрифт Mister-Brush. Надпись при проигрыше имеет красную обводку, а количество очков зеленую.

Были созданы отдельные материалы для всех платформ, чтобы они отличались по цвету, и было понятно, что игрок дошел до нового уровня.

Рисунок 11 – Подсказки

Чтобы определить логику игры или поведение ее персонажей, необходимо написать сценарии. Разработка сценариев в Unity начинается с создания нового файла сценария – обычного текстового файла, добавляемого в проект [8]. Классы – это шаблоны. Они устанавливают правила для любого объекта, созданного с помощью этого шаблона [9].

В этом коде (листинг 1) запускается корутина `BearCanJump` в методе `Start`, которая позволяет персонажу передвигаться по платформам. В `Fixed Update`, когда игрок нажимает на клавишу, персонаж сжимается, создавая небольшую анимацию, благодаря тому, что изменяются его размеры. Если на клавишу нажать дольше, сжатие персонажа будет более заметным. Когда игрок совершает неудачный прыжок на платформу, и персонаж уходит за пределы поля игры, он автоматически удаляется с помощью функции `Destroy`.

Листинг 1 – Управление персонажем

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class BearJumps : MonoBehaviour
{
    public static bool jump, nextPLatform;
    public GameObject bear;
    private bool jumpReady, lose; //successJump;
    private float squeezeSpeed = 0.5f, pushTime, bearPosY; //pushtime
время нажатия на кнопку
    public static int points; //количество набранных очков
    public Text lostTheGame;
    void Start () {
        StartCoroutine (BearCanJump());
    }
    void  FixedUpdate() { // сжимание персонажа при нажатии клавишы
        if (jumpReady && bear.transform.localScale.y > 0.45f && bear!=null)
{
            BearSqueeze (-squeezeSpeed);
        } else if (!jumpReady&& bear!=null) {
            if (bear.transform.localScale.y <0.5f) {
            BearSqueeze (squeezeSpeed * 4f);
        } else if (bear.transform.localScale.y !=0.5f)
            bear.transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);
        }
        if (bear!=null) {
            if (bear.transform.position.y< -12f){ // удаляем игрока при
выходе с поля игры
                Destroy(bear, 1f);
                lose = true;
                lostTheGame.gameObject.SetActive (true);
            }
        }
```

В методе `OnMouseUp` (листинг 2) отслеживается сила нажатия на клавишу. В зависимости от того, насколько долго зажата кнопка, изменяется сила прыжка. Сила подобрана таким образом, что невозможно вылететь за пределы поля, даже при очень долгом зажатии.

Листинг 2 – Осуществление прыжка

```
void OnMouseUp() { // в данной функции осуществляется прыжок
        if (bear.GetComponent<Rigidbody> () && nextPLatform) {
        jumpReady = false;
        jump = true;
        //successJump = true;
        float power, squeezePeriod;
        squeezePeriod = Time.time - pushTime;
        if (squeezePeriod < 2f)
            power = 120*squeezePeriod;
        else
            power = 150f;
        if (power<50f)
            power = 50f;
        bear.GetComponent<Rigidbody>().AddRelativeForce (bear.transform.up
* power, 0f);
```

```
        bear.GetComponent<Rigidbody>().AddRelativeForce
(bear.transform.right * power, 0f);
        StartCoroutine (checkCurrentBearPos ());
        nextPLatform = false;
        }
    }
```

В листинге 3 происходит проверка положение игрока на платформах. Игрок может прыгнуть только один раз на одну платформу. Но, при условии, что платформа находится в движении, можно успеть прыгнуть еще. Если игрок попадает на одну и ту же платформу, находящуюся в покое, он проигрывает. Так же игрок считается проигравшим, если не попал ни на одну платформу, после совершения прыжка. В таком случае, на экране появляется надпись «You lost the game». Данная надпись подсвечивается.

Листинг 3 – Динамика игры

```
IEnumerator checkCurrentBearPos () {
        yield return new WaitForSeconds (1f); // если попал на ту же
платформу то проиграл
        if (bearPosY == bear.transform.localPosition.y){
         points--;
         lose = true;
         lostTheGame.gameObject.SetActive (true);
        }
        else {
            while (!bear.GetComponent<Rigidbody> ().IsSleeping ()) {
                yield return new WaitForSeconds(0.1f);
                if (bear == null)
                    break;
            }
            if (!lose) {
                nextPLatform = true;
                points ++;
                print ("next platform");
                bear.transform.localEulerAngles = new Vector3 (0f,
bear.transform.eulerAngles.y, 0f);
            }
        }
    }
```

В листинге 4 представлен класс `PlatformMovement`, который управляет передвижением платформ по полю. Платформы формируются за пределами игры. Они выезжают из нижней части экрана. Когда игрок осуществляет успешный прыжок, платформы, которые находятся на поле, передвигаются на одинаковое расстояние по x и y. Движение по

координате z не происходит.

Листинг 4 – Динамика игры

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UIElements;

public class PlatformMovement : MonoBehaviour
{
    private bool moved = true;
    public GameObject bearAndPlatform;
    private Vector3 platformTarget;
    void Start()
    {
       platformTarget = new Vector3 (-1.18f, 1.45f, -9.6f);
    }
    void Update () {
         if (BearJumps.nextPLatform) {
             if (transform.position!=platformTarget) {
                 transform.position = Vector3.MoveTowards (trans-
form.position, platformTarget, Time.deltaTime * 5f);
             }
              else if (transform.position == platformTarget && !moved) {
                 platformTarget = new Vector3 (transform.position.x - 1.2f,
transform.position.y + 1.2f, transform.position.z);
                 BearJumps.jump = false;
                 moved = true;
         }
             if (BearJumps.jump)
                moved = false;
    }
}
}
```

Класс `PlatformGenerator`, который занимается созданием платформ разного уровня (листинг 5). При достижении определенного числа очков, параметры платформ меняются, так же изменяется их положение на поле и цвет, чтобы можно было сразу видеть уровень сложности игры. Легче всего попасть на большие платформы.

Листинг 5 – Создание новых платформ

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlatformGenerator : MonoBehaviour
{
    public GameObject platformA, platformB, everyPLatform;
    private GameObject platformCreated;
    private Vector3 platformTarget;
    private float speed = 4f;
    private bool onNewPLatform;
    void Start () {
      platfsGeneratorLa();
    }
```

```
     void Update () {
        if (platformCreated.transform.position!= platformTarget && !onNew-
PLatform)
            platformCreated.transform.position = Vector3.MoveTowards (plat-
formCreated.transform.position, platformTarget, Time.deltaTime * speed);
        else if (platformCreated.transform.position == platformTarget)
            onNewPLatform = true;
        if (BearJumps.jump && BearJumps.nextPLatform && BearJumps.points<2)
{
            platfsGeneratorLa();
            onNewPLatform = false;
        }
        else if (BearJumps.jump &&BearJumps.nextPLatform && Bear-
Jumps.points % 50 == 0) {
                PlatformGeneratorLb();
        }
     }
```

Класс `PlatformGenerator`, который занимается созданием платформ разного уровня (листинг 6). При достижении определенного числа очков, параметры платформ меняются, так же изменяется их положение на поле и цвет, чтобы можно было сразу видеть уровень сложности игры. Каждая платформа изменяется по параметру x и становится меньше. Платформы каждого типа различны по этому параметру в некоторых пределах.

Листинг 6 – Платформы разного уровня

```
void platfsGeneratorLa () {
       platformTarget = new Vector3 (Random.Range (0.97f, 1.44f), Ran-
dom.Range (-2.4f, -1.33f), -8.6f);
       platformCreated = Instantiate (platformA, new Vector3 (4.5f, -5.8f,
-8.6f), Quaternion.identity) as GameObject;// начальная точка создания
       platformCreated.transform.localScale = new Vector3 ( Random.Range
(2.6f, 2.9f), platformCreated.transform.localScale.y, platformCreat-
ed.transform.localScale.z);
       platformCreated.transform.parent = everyPLatform.transform;
}
void PlatformGeneratorLb () {
       platformTarget = new Vector3 (Random.Range (0.97f, 1.44f), Ran-
dom.Range (-2.4f, -1.33f), -8.6f);
       platformCreated = Instantiate (platformB, new Vector3 (4.5f, -5.8f,
-8.6f), Quaternion.identity) as GameObject;       platformCreat-
ed.transform.localScale = new Vector3 ( Random.Range (1.6f, 1.9f), plat-
formCreated.transform.localScale.y, platformCreat-
ed.transform.localScale.z);
       platformCreated.transform.parent = everyPLatform.transform;
}
void PlatformGeneratorLc () {
       platformTarget = new Vector3 (Random.Range (0.97f, 1.44f), Ran-
dom.Range (-2.4f, -1.33f), -8.6f);
       platformCreated = Instantiate (platformB, new Vector3 (4.5f, -5.8f,
-8.6f), Quaternion.identity) as GameObject;
       platformCreated.transform.localScale = new Vector3 ( Random.Range
(1.3f, 1.0f), platformCreated.transform.localScale.y, platformCreat-
ed.transform.localScale.z);
       platformCreated.transform.parent = everyPLatform.transform;
```

Класс `DetectClicks` занимается управлением всех кнопок и запуска объектов, участвующих в игре (листинг 7). Так же здесь хранятся ссылки на надписи-подсказки. Первая подсказка отображается на экране, когда проиграла анимация главного персонажа и первой платформы. Вторая подсказка возникает, когда первая уже бездействует, после того, как было осуществлено первое нажатие. Данные подсказки имеют анимации. Они изменяют свой цвет. Подсказки автоматически исчезают, когда игрок прыгает успешно.

Листинг 7 – Начальный запуск игры

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class DetectClicks : MonoBehaviour
{
    public GameObject buttons, bear, platform_generator, bearAndPlatform;
    public Text gameName, playText, recomOne, recomTwo, points;
    private bool clickedAlready;
    void   OnMouseDown() {
        if (!clickedAlready) {
        clickedAlready = true;
        playText.gameObject.SetActive (false);
        gameName.gameObject.SetActive (false);
        points.gameObject.SetActive (true);
        buttons.GetComponent <ScrollObjs> ().speed = -10f;
        buttons.GetComponent <ScrollObjs> ().checkPosition = -130f;
        bearAndPlatform.GetComponent <Animation>
().Play("BearAndPlatform");
        recomOne.gameObject.SetActive (true);
        StartCoroutine (bearGetsComponent ());
        }
        else if (clickedAlready && recomOne.gameObject.activeSelf) {
            recomOne.gameObject.SetActive (false);
            recomTwo.gameObject.SetActive (true);
        }
        else if (clickedAlready && recomTwo.gameObject.activeSelf) {
            recomTwo.gameObject.SetActive (false);
        }
        platform_generator.GetComponent <PlatformGenerator> ().enabled =
true;
    }
    IEnumerator bearGetsComponent () {
        yield return bear.AddComponent<Rigidbody> ();
    }
}
```

Класс `Points` занимается подсчетом очков, набранных в игре (листинг 8). Подсчитывается максимальное количество набранных очков и

текущее количество очков. После запуска игры заново, лучший результат обновляется, если получено больше очков.

Листинг 8 – Подсчет очков

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class Points : MonoBehaviour
{
    private bool startthegame;
    private Text currentPoint;
    public Text bestPoint;
    void Start()
    {
        bestPoint.text = "Best: " + PlayerPrefs.GetInt
("BestPoints").ToString ();
        currentPoint = GetComponent <Text> ();
        BearJumps.points = 0;
    }
    void Update(){
      if (currentPoint.text == "0")
          startthegame = true;
      if (startthegame)
          currentPoint.text = BearJumps.points.ToString ();
            if (PlayerPrefs.GetInt ("BestPonts") < BearJumps.points) {
                PlayerPrefs.SetInt ("BestPoints", BearJumps.points);
                bestPoint.text = PlayerPrefs.GetInt ("BestPoints").ToString
();
            }
    }
}
```

Выводы по третьей главе

В третьей главе было разработано игровое приложение в соответствии с заранее продуманным сценарием и дизайном на платформе Unity на языке C#. Данное приложение имеет интересное развитие сюжета, простые анимации, не перегружающие игру и интуитивно понятный интерфейс. Так же внутри игры есть подсказки, благодаря которым обучение происходит еще проще. В игре присутствуют звуковые эффекты, которые можно отключать при желании.

4. ТЕСТИРОВАНИЕ

4.1. Функциональное и юзабилити-тестирование

Во время разработки игры и после ее завершения был проведен ряд тестов, чтобы определить, что все работает так, как задумано и исправить ошибки, если они есть (таблица Таблица 1 – Тестирование).

Таблица 1 – Тестирование

№	Название теста	Действия	Ожидаемый результат	Пройден ли тест?
1.	Запуск приложения.	Запустить приложение.	Игра запустилась	Да
2.	Проверка анимации игрока и платформы.	Запустить игру, нажав на панели запуск игры.	Главный игрок и платформа под ним уезжают на дальний план и	Да
3.	Проверка анимации фразы, оповещающей о проигрыше.	Запустить игру и прыгнуть не на платформу или попасть на одну и ту же платформу два раза.	Появляется надпись, оповещающая о проигрыше	Да
4.	Проверка возможности осуществления прыжков во время движения платформ.	Запустить игру, проверить двойные прыжки по одной платформе во время движения.	Игрок не проиграл	Да
5.	Начисление очков за попадание на платформу.	Запустить игру и успешно прыгнуть на платформу.	Начислен балл. Он отображается в верхней части экрана.	Да
6.	Проверка анимации при прыжке.	Запустить игру и осуществить прыжок. Повторить то же, удерживая клавишу дольше.	Игрок сжимается. Чем дольше зажата клавиша, тем больше.	Да
7.	Проверка фразы-подсказки о необходимости совершить прыжок.	Запустить игру.	Появляется фраза до прыжки и исчезает после.	Да
8.	Проверить фразу-подсказку о зажатии клавиши.	Запустить игру и прыгнуть.	Появляется фраза после осуществления прыжка. Затем исчезает.	Да
9.	Проверить изменение размера и материала выезжающих платформ разной сложности.	Запустить игру и совершить необходимое количество успешных прыжков.	Платформы меняют размер и материал.	Да

Все ошибки, найденные при осуществлении тестирования вручную, были найдены и исправлены.

Было проведено юзабилити-тестирование, чтобы выяснить возможные нюансы и ошибки, которые были не учтены при разработке приложения. В данном тестировании участвовало пять человек. Были внесены некоторые изменения в дизайн игры и увеличена сила, с которой игрок может попадать на разные платформы. Сила прыжка для успешного попадания на сложные платформы была недостаточной. Были изменены материалы некоторых платформ. Для подсказок были использованы новые шрифты, чтобы вид стал интереснее. Добавлена анимация для спуска кнопок главного меню вниз и изменено перемещение основного персонажа на задний план вместе с первой платформой, не участвующей в игре. Степень сжатия основного персонажа при осуществлении прыжка была подкорректирована, чтобы меньше искажать изображение. Так же первая платформа была уменьшена в размерах, чтобы отличаться от платформ первого уровня сложности и было понятно, что они имеют разные свойства.

Выводы по четвертой главе

В данной главе было проведено функциональное и юзабилити-тестирование разработанного приложения и найдены ошибки и недочеты в игре. Все ошибки были исправлены, а пожелания пользователей учтены. Тестирование помогло создать более качественное и интересное игровое приложение.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана компьютерная игра в жанре «Аркада» для платформы Android. Разработка была осуществлена на платформе Unity на языке C#. Игровое приложение имеет ненавязчивую простую анимацию, звуковые эффекты и интуитивно понятный интерфейс. Сюжет развивается при наборе определенного количества баллов, максимально набранное число которых, игрок видит на экране. Игра оснащена подсказками, для более легкого прохождения. В результате разработки приложения были осуществлены следующие задачи.

1. Разработан игровой процесс игры.
2. Спроектировано приложение.
3. Разработан дизайн приложения.
4. Спроектирована структура игры под Android.
5. Реализовано и протестировано приложение.

Для дальнейшего развития проекта необходимо:

1) усложнить концепцию игры;
2) добавить анимацию;
3) добавить возможность менять внешний вид персонажа;
4) добавить выбор уровня сложности.

Опыт, который был получен в результате работы над данным проектом, будет очень полезен при разработке новых игровых приложений на платформе Unity

ЛИТЕРАТУРА

1. Бонд Д.Г. Unity и C#. Геймдев от идеи до реализации. // 2-е издание СПб: Питер, 2021. – 928 с.
2. Дарвин Я. Android. Сборник рецептов. Задачи и решения разработчиков приложений. // Издательство Вильямс, 2018. – 768 с.
3. Зобнин Е.Е. Android глазами хакера. // Интеллектуальная издательская система Ridero, 2016. – 337 с.
4. Корнилов А.В. Unity. Полное руководство. // СПб: Наука и Техника, 2021. – 496 с.
5. Ларкович С.Н., Купрянова А.В. Привет, Unity! Моя первая книга по созданию игр. // СПб: Наука и Техника, 2021. – 288 с.
6. Линовес Дж. Виртуальная реальность в Unity. // CommonsWare, 2019. – 412 с.
7. Мэннинг Д., Батфилд-Эддисон П. Unity для разработчика. Мобильные мультиплатформенные игры. // СПб: Питер, 2018. – 304 с.
8. Торн А. Искусство создания сценариев в Unity. // ДМК Пресс, 2016. – 360 с.
9. Ферроне Х. Изучаем C# через разработку игр на Unity. // 5-е издание СПб: Питер, 2022. – 400 с.
10. Хокинг Д. Unity в действии. Мультиплатформенная разработка на C#. // 2-е издание СПб: Питер, 2019. – 352 с.
11. Черников В.Н. Разработка мобильных приложений на C# для iOS и Android / под ред. Мовчана Д. – ДМК Пресс, 2020 – 188 с.
12. Lewis S., Dunn M. Нативная разработка мобильных приложений. Перекрестный справочник для iOS и Android. // ДМК Пресс, 2020. – 377 с.
13. Meier R., Lake I. Professional Android. // Wrox, 2018. – 911 с.
14. Sapio F. Getting Started with Unity 5.x 2D Game Development. // Pearson Technology Group, 2022. – 688 с.
15. Shekar S., Karim W. Mastering Android Game Development. // Packt Publishing Ltd, 2017. – 330 с.