Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Модели данных и системы управления базами данных

К защите допустить:

И.О. Заведующего кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_\_С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**БИБЛИОТЕКА**

БГУИР КП 1-40 04 01 011 ПЗ

Студент Е. А. Киселёва

Руководитель В. С. Плиска

Минск 2024

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc176219843)

[1 Общая характеристика компании 4](#_Toc176219844)

[1.1 Информация о компании 4](#_Toc176219845)

[1.2 Методология разработки 5](#_Toc176219846)

[1.3 Процесс разработки программного кода 6](#_Toc176219847)

[2 Обзор используемых технологий 8](#_Toc176219848)

[2.1 C# 8](#_Toc176219849)

[2.2 Unity 9](#_Toc176219850)

[2.3 Unity UI 10](#_Toc176219851)

[2.4 Unity Animator 11](#_Toc176219852)

[2.5 Git 12](#_Toc176219853)

[2.6 Firebase 14](#_Toc176219854)

[3 Индивидуальное задание 16](#_Toc176219855)

[3.1 Имплементация навыка персонажа 16](#_Toc176219856)

[3.2 Имплементация экрана покупки навыка 19](#_Toc176219857)

[3.3 Возможность удаленного включения или выключения функционала 20](#_Toc176219858)

[3.4 Имплементация индикаторов отслеживания объектов нового типа 21](#_Toc176219859)

[Заключение 24](#_Toc176219860)

[Список литературных источников 25](#_Toc176219861)

****Введение****

В условиях цифровой трансформации библиотеки сталкиваются с необходимостью модернизации процессов хранения, учета и предоставления информации. Это обуславливает актуальность разработки современных информационных систем, которые обеспечивают централизованное управление данными и удобный доступ пользователей к библиотечным ресурсам.

Для эффективного функционирования сети библиотек требуется современная информационная система, которая позволяет управлять данными о книгах, читателях и их взаимодействии с библиотекой.

Целью курсового проекта является разработка базы данных (БД) согласно выбранной теме. К задачам курсового проекта относятся:

– определить сущности проектируемой БД и их связи;

– нормализовать БД до третьей нормальной формы;

– создать пул запросов;

– создать индексы для наиболее часто используемых сущностей;

– создать пул триггеров и хранимых процедур для работы с БД.

С учетом выбранной темы необходимо разработать базу данных для веб-приложения сети библиотек, которая обеспечит централизованное управление данными и автоматизацию ключевых процессов.

Для достижения поставленной цели в работе решаются следующие задачи:

1 Проведение анализа литературных источников и обзор существующих аналогов информационных систем для библиотек, выявление их преимуществ и недостатков, формулировка требований к разрабатываемой системе.

2 Определение функциональных требований к базе данных и выбор оптимальной системы управления базами данных на основе анализа доступных инструментов и их характеристик.

3 Разработка проектной документации, включая инфологическую и даталогическую модели, модель миграции данных, IDEF-диаграммы и описание форматов хранения данных.

4 Создание физической базы данных, включая реализацию скриптов для её построения, настройку индексов, триггеров, процедур, функций и других элементов, а также, при необходимости, разработку ORM.

5 Проведение тестирования базы данных, включая проверку на корректность ограничений, тесты на аномалии, проверку работы процедур, триггеров и других компонентов системы.

В результате выполнения поставленных задач будет создана база данных, которая обеспечит надежное хранение данных, эффективный поиск и управление библиотечными ресурсами, а также удовлетворит потребности пользователей сети библиотек.

Курсовая записка составлена в соответствии со стандартом предприятия СТП 01-2024. [1]

# Анализ литературных источников и обзор             существующих аналогов

Для разработки базы данных, которая станет основой сайта сети библиотек, важно проанализировать существующие решения, обеспечивающие управление библиотечными ресурсами через веб-приложения. На этом этапе рассматриваются современные подходы к созданию сайтов для библиотек, анализируются их функциональные возможности, архитектура и используемые технологии.

Целью данного анализа является выявление сильных и слабых сторон существующих аналогов, что позволит сформулировать обоснованные требования к создаваемому сайту. На основе этих требований впоследствии будут определены задачи базы данных, включая структуру хранения данных, функционал обработки запросов и поддержку пользовательских операций.

Результатом данного этапа станет перечень функциональных требований к веб-приложению сети библиотек, который станет отправной точкой для проектирования базы данных и дальнейшей реализации системы.

## Анализ литературных источников

Системы управления данными в библиотеках предоставляют возможность автоматизировать ключевые процессы, повышая доступность ресурсов и удобство их использования для читателей. Они также играют важную роль в обеспечении эффективного управления фондами и взаимодействия с пользователями через цифровые платформы.

Можно выделить следующие преимущества использования систем управления базами данных в библиотеках:

1 Централизованное хранение данных: предоставление централизованного хранилища информации о библиотечных фондах, читателях и услугах, что позволяет избежать дублирования данных, обеспечивает их согласованность и упрощает доступ к информации.

2 Автоматизация процессов: системы управления данными помогают автоматизировать рутинные операции, такие как учет книг, регистрация читателей, обработка запросов на бронирование и продление сроков сдачи. Это сокращает время выполнения задач и снижает нагрузку на персонал.

3 Гибкость и масштабируемость: адаптация к изменениям в потребностях библиотеки, включая увеличение объемов данных, добавление новых функций и интеграцию с внешними сервисами.

4 Улучшение пользовательского опыта: обеспечение доступа к информации о книгах, услугах и мероприятиях в режиме онлайн, что делает работу с библиотекой более удобной для пользователей. Например, они могут бронировать книги, просматривать свои читательские записи и получать уведомления о сроках возврата.

5 Круглосуточный доступ: цифровые платформы позволяют пользователям получать доступ к библиотечным ресурсам в любое время и из любого места, что значительно повышает удобство использования.

Системы управления данными в библиотеках необходимы для веб-приложений библиотек, а тем более для библиотек, объединенных в целую сеть, так как помогают автоматизировать процессы, лучше структурировать информацию и предоставить пользователям больше возможностей.

База данных является основой таких систем, обеспечивая хранение и обработку больших объемов информации. Это систематизированное хранилище данных, структурированное таким образом, чтобы они могли быть легко найдены и обработаны. База данных является неотъемлемой частью такой системы, так как с помощью нее можно автоматизировать множество процессов.

База данных – это представленные в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью ЭВМ.

В современных приложениях работа с базой данных ведется посредством готового программного обеспечения, которое содержит в себе все необходимые инструменты и компоненты для успешного взаимодействия с хранилищем данных. Такое программное обеспечение называется системой управления базой данных (СУБД). Основными функциями СУБД являются:

– создание баз данных, изменение, удаление и объединение их по определенных признакам;

– хранение данных, в том числе больших массивов, в структурированном виде и нужном формате;

– защита данных от взлома и нежелательных изменений при помощи распределенного доступа;

– выгрузка и сортировка данных по заданным фильтрам при помощи SQL-запросов;

– поддержка целостности баз данных, резервное копирование и восстановление после сбоев. [2]

СУБД классифицируются на разные типа в зависимости от моделей используемых данных, способов предоставления доступа к БД, а также по уровню распределенности. Классифицируются СУБД по следующим признакам:

– модель данных (иерархические, сетевые, реляционные, объектно-ориентированные, объектно-реляционные);

– степени распределенности (локальные, распределенные);

– способ доступа к БД (файл-серверные, клиент-серверные, встраиваемые). [3]

Логическое и физическое представление БД – это два разных способа описания и организации данных в БД. Логическое представление отражает, как данные понимаются и используются пользователями и приложениями. Физическое представление отражает, как данные хранятся и обрабатываются системой управления БД.

Логическое представление базы данных состоит из логических объектов, таких как таблицы, представления, индексы. Эти объекты определяют структуру, свойства и отношения данных, а также правила и ограничения, которые обеспечивают целостность и согласованность данных. Логическое представление базы данных может быть описано с помощью концептуальной схемы, которая показывает все элементы данных и их связи в виде графической диаграммы. Логическое представление базы данных может быть разделено на несколько уровней абстракции, таких как внешний, логический и внутренний.

Внешний уровень представляет данные в формате, понятном конкретному пользователю или приложению. Логический уровень представляет данные в формате, понятном всем пользователям и приложениям. Внутренний уровень представляет данные в формате, понятном системе управления базой данных.

Физическое представление базы данных состоит из физических объектов, таких как файлы, блоки, страницы и сегменты. Эти объекты определяют способ хранения, доступа и обработки данных на физическом носителе, таком как диск, память или сеть.

Физическое представление базы данных может быть описано с помощью физической схемы, которая показывает расположение и размер физических объектов, а также параметры и настройки, которые влияют на производительность и эффективность базы данных.

Физическое представление базы данных может быть оптимизировано с помощью различных методов и техник, таких как сжатие, шардирование, кэширование, индексирование и т.д.

Разделение логического и физического представления базы данных позволяет достичь логической и физической независимости данных. Логическая независимость означает, что изменения в логической структуре данных не влияют на прикладные программы и пользователей. Физическая независимость означает, что изменения в физической организации данных не влияют на логическую структуру данных и прикладные программы.

В процессе работы с базой данных могут возникать аномалии – ошибки, определяющие несоответствие модели данных и модели предметной области. Выделяют три вида аномалий:

1 Аномалии вставка: аномалии вставки определяются, когда при добавлении новой записи в таблицу отсутствует часть данных и появляется необходимость либо оставить поле пустым, либо заполнить фиктивными данными, либо не добавлять записи вовсе.

2 Аномалии удаления: аномалии удаления определяются тогда, когда при удалении ненужного блока данных происходит потеря полезных данных, все еще необходимых приложению.

3 Аномалии обновления: аномалии обновления определяются тогда, когда происходит сбой обновления данных в таблице, то есть не проходит корректное обновление всех данных, либо не проходит обновление данных вовсе.

Для устранения аномалий принято производить нормализацию таблицы базы данных. Нормализация – это группировка и/или распределение атрибутов по отношениям с целью устранения аномалий операций с БД, обеспечивая целостность данных и оптимизации модели БД.

Существуют несколько требований к нормализации, которые, в свою очередь, могут противоречить друг другу. Для этого необходимо выбирать тот минимальный набор требований, который поможет решить проблему в конкретном случае и не повлечет серьезных изменений архитектуры приложения.

При приведении БД к первой нормально форме необходимо все элементы составных данных разнести по разным полям, иначе в процессе работы с данными могут появиться ошибки и аномалии. Таким образом в БД не должно быть составных данных, а также дубликатов.

При приведении БД ко второй нормальной форме необходимо, чтобы все ее таблицы были приведены к первой нормальной форме, а также в каждой таблице отсутствовали частичные зависимости. Другими словами, у таблиц не должно быть атрибутов, зависящих только от части первичного ключа. Отказ от составного первичного ключа имеет один неоспоримый плюс, а именно: упрощается процесс приведения таблицы ко второй нормальной форме. Доказательство находится на поверхности: если первичный ключ не является составным, то нет и частичных зависимостей.

При приведении БД к третьей нормальной форме необходимо, чтобы все ее таблицы были приведены ко второй нормальной форме, а также чтобы в каждой таблице отсутствовали транзитивные зависимости.

Помимо третьей нормальной формы, специалисты выделяют усиленную разновидность третьей нормальной формы – нормальную форму Бойса-Кодда. Смысл усиления в том, что во время формулирования первоначальных требований к третьей нормальной форме Кодд не предусмотрел вероятность того, что в нормализуемом отношении может существовать более одного потенциального ключа, указанные ключи окажутся составными и эти ключи станут обладателями хотя бы одного общего атрибута. Вероятность совместного возникновения перечисленных событий крайне невысока, но все-таки не исключена.

Если третья нормальная форма призвана бороться с транзитивными зависимостями, то четвертая нормальная форма состоит в конфронтации с другой проблемой реляционной модели – многозначными зависимостями. Многозначная зависимость – это не что иное, как связи «многие ко многим». Таким образом, БД, соответствующая четвертой ступени нормализации, обязана избавиться от многозначных зависимостей между атрибутами отношений.

Для обычного разработчика БД пятая нормальная форма представляет скорее теоретический, нежели практический интерес. Пятая нормальная форма требует обеспечения беспрепятственной возможности перестройки данных в нормализованных таблицах. Приведение таблицы к высшей степени нормализации – это крайне редкий случай. Это действие имеет смысл в том случае, если таблицы содержит зависимые сочетания. Зависимые сочетания – это свойство декомпозиции, которое вызывает генерацию ложных строк при обратном соединении декомпозированных отношений с помощью операции естественного соединения.

Нормализация осуществляется на этапе логического проектирования БД и представляет собой вариант восходящего подхода, который начинается с установления связей между атрибутами. На практике для построения приемлемой логической модели БД следует пройти только три или четыре ступени нормальных форм:

- все поля в таблицах неделимы и не содержат повторяющихся групп

- все не ключевые поля в таблицах зависят от первичного ключа

- в таблицах отсутствуют избыточные не ключевые поля

- в таблицах устранены многозначные зависимости

Таким образом, можно определить, что проектирования БД является, наряду с выбором платформы и архитектуры, ключевым элементом в планировании и разработке приложения.

## Существующие аналоги приложения

Для анализа существующих аналогов сайтов библиотек были рассмотрены несколько популярных решений, используемых в различных библиотеках. Эти сайты предоставляют функционал для управления ресурсами, взаимодействия с пользователями и автоматизации основных библиотечных процессов.

**1.2.1** Сайт национальной библиотеки Республики Беларусь

Одним из ключевых аспектов, который выделяет Суперлевел среди конкурентов, является пристальное внимание к визуальной составляющей игр. Каждая деталь тщательно прорабатывается, чтобы создать миры, которые не только выглядят красиво, но и погружают игрока в атмосферу. Например, боевые сцены в играх компании славятся динамикой и эффектностью, что делает каждый поединок захватывающим и зрелищным.

Суперлевел использует современные технологии, чтобы графика их игр выглядела на высшем уровне, даже на мобильных устройствах. Это позволяет достичь высокого уровня погружения и делает игры привлекательными для широкой аудитории.

Суперлевел известна не только визуальной составляющей, но и своим подходом к разработке игровых механик. Команда ищет новые и интересные способы взаимодействия с игрой, стремясь сделать каждую новую игру лучше предыдущей. Например, интегрируются сложные боевые системы, которые остаются интуитивно понятными для новичков, но в то же время предлагают глубину и стратегические возможности для более опытных игроков.

Эти инновации помогают Суперлевел выделяться в насыщенном рынке мобильных игр. Игроки ценят те неожиданные повороты и уникальные элементы, которые они находят в играх компании.

Основная цель Суперлевел заключается в создании игр, к которым пользователи хотят возвращаться снова и снова. Благодаря захватывающему геймплею, детализированной графике и продуманным механикам, игры Суперлевел не только привлекают внимание, но и удерживают его.

Таким образом, Суперлевел предлагает игрокам глубокий, яркий и запоминающийся опыт, который вызывает желание вновь и вновь открывать приложение и погружаться в созданные ими виртуальные миры. Это стремление к постоянному совершенствованию и инновациям позволило Суперлевел занять уверенное место среди лидеров мобильного геймдева.

## Требования к приложению

Компания Суперлевел работает по методологии Kanban, сосредотачиваясь на гибкости и эффективности процессов.

Kanban – это методология управления проектами и процессами, которая фокусируется на визуализации работы, управлении потоком задач и непрерывном улучшении процессов. Изначально разработанная в Toyota для оптимизации производственных процессов, Kanban был адаптирован для использования в разработке программного обеспечения и других областях.

Основные принципы Kanban:

1 Визуализация работы: основной инструмент Kanban доска, на которой визуализируется весь процесс работы. Обычно доска разделена на колонки, каждая из которых соответствует этапу процесса (например, «Запланировано», «В процессе», «Готово»). Задачи представляются в виде карточек, которые перемещаются по доске от одной колонки к другой по мере выполнения.

2 Ограничение незавершенной работы (WIP – Work In Progress): одним из ключевых аспектов Kanban является ограничение количества задач, которые могут находиться в работе на каждом этапе, что помогает избежать перегрузки команды и улучшает фокус на текущих задачах. WIP-лимиты способствуют выявлению и устранению узких мест в процессе.

3 Управление потоком: в Kanban важен непрерывный поток работы. Целью является минимизация времени, затрачиваемого на выполнение каждой задачи, и оптимизация процесса так, чтобы задачи быстро перемещались по доске. Постоянное отслеживание и улучшение потока задач помогает повышать производительность команды.

4 Ясные и прозрачные правила процесса: в Kanban должны быть четко определены правила для каждой стадии процесса. Это помогает команде лучше понимать, как выполняется работа и какие задачи являются приоритетными. Все участники команды должны знать эти правила и следовать им.

5 Непрерывное улучшение: kanban поощряет регулярное пересмотрение и оптимизацию процессов. Команда должна постоянно искать способы улучшить эффективность, устранить узкие места и повысить качество работы. Регулярные ретроспективы и анализ метрик, таких как цикл выполнения задач и производительность, помогают вносить улучшения.

Kanban-доска является центральным элементом методологии. Она может быть физической (на стене с использованием карточек) или виртуальной (в специальных инструментах, таких как Jira, Trello, Asana и др.). В компании Суперлевел используется инструмент Jira. Колонки на доске представляют этапы процесса, через которые проходит задача (например, «To Do», «In Progress», «Code Review», «Done»). Каждая карточка на доске представляет отдельную задачу или элемент работы. Карточки перемещаются по доске по мере выполнения работы. WIP-лимитами называются ограничения на количество задач в каждой колонке. Например, если лимит для колонки «In Progress» установлен на 3, это значит, что одновременно может выполняться не более трех задач.

Преимущества Kanban:

1 Гибкость: kanban можно внедрять в существующие процессы без необходимости радикальных изменений. Подходит как для крупных, так и для небольших команд.

2 Прозрачность: вся работа команды видна и доступна для анализа, что упрощает управление проектом.

3 Улучшение производительности: ограничение WIP и фокус на управлении потоком задач помогают команде работать более эффективно и избегать перегрузки.

4 Быстрое реагирование на изменения: kanban легко адаптируется к изменениям в требованиях или приоритетах, так как задачи могут добавляться и изменяться в любой момент времени.

Недостатки Kanban:

1 Отсутствие строгой структуры: kanban предоставляет много гибкости, что может стать проблемой для команд, которые привыкли к строгим процессам или не обладают высокой самоорганизацией.

2 Зависимость от дисциплины: для успешного использования Kanban требуется высокая дисциплина и самоконтроль команды, особенно в соблюдении WIP-лимитов и регулярном обновлении доски.

3 Сложность в масштабировании: хотя Kanban хорошо работает для небольших команд, его может быть трудно масштабировать на уровень крупных проектов или организаций без дополнительных инструментов и адаптаций.

Kanban – это мощный и гибкий инструмент для управления процессами, который может быть полезен в самых разных сферах, включая разработку программного обеспечения. Успешное внедрение Kanban требует понимания его принципов и готовности команды к постоянному улучшению.

## Процесс разработки программного кода

Процесс разработки в компании Суперлевел организован поэтапно и включает в себя:

1 Анализ задачи и технического задания, формирование плана решения, на котором происходит разбор целей и выбор подходящих методов для их достижения, формулировка задачи в формате входных и выходных данных, ожидаемых результатов, выявление дополнительных требований и возможных рисков.

2 Проектирование решения задачи, включающее в себя оценку результатов анализа и выявление ограничений, определение критических моментов, проектирование ключевых элементов, включая процессы и код, выбор инструментов разработки.

3 Программирование требуемой функциональности, включающее в себя реализацию плана разработки с учётом проектной архитектуры и стандартов качества кода, внесение изменений в код в соответствии с возникающими потребностями проекта.

4 Тестирование функциональности программного обеспечения, включающее в себя проведение тестирования и возврат задач на доработку при выявлении ошибок.

5 Интеграция изменений в систему проекта, включающих в себя внедрение разработанных решений и проверка их взаимодействия с уже существующими компонентами системы.

Таким образом, процесс разработки в компании Суперлевел представляет собой полный и тщательно организованный цикл, направленный на достижение высокого качества и успешности проекта. Этот процесс охватывает все ключевые аспекты разработки, начиная с глубокого анализа задачи и продуманного планирования, что позволяет заложить прочную основу для дальнейшей работы. Продвижение от проектирования и программирования до тестирования и интеграции изменений осуществляется с особым вниманием к деталям и стандартам качества. Такой системный подход гарантирует, что каждое решение будет тщательно проверено и оптимизировано, а внедрение новых функциональных возможностей пройдет гладко и без сбоев, что в конечном итоге приводит к созданию надежного и эффективного программного обеспечения.

# Обзор используемых технологий

В ходе прохождения производственной практики были исследованы и использованы следующие технологии, используемые при разработке проектов компании:

– C#;

– Unity;

– Unity UI;

– Unity Animator;

– Git;

– Firebase.

Перечисленные выше технологии распространены в современном мире, а умение их использовать – надежная точка для профессионального роста.

## C#

C# – это современный объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он был представлен в 2000 году как часть платформы .NET и с тех пор стал одним из самых популярных языков для разработки приложений, особенно в экосистеме Microsoft. C# поддерживает принципы объектно-ориентированного программирования, такие как инкапсуляция, наследование, полиморфизм и абстракция, что делает его мощным инструментом для создания сложных и масштабируемых приложений. Язык является строго типизированным, что помогает избежать множества ошибок, связанных с типами данных, и делает код более надежным. Важным аспектом C# является автоматическое управление памятью с помощью сборки мусора, что устраняет необходимость в ручном управлении памятью и снижает вероятность утечек памяти.

C# тесно интегрирован с платформой .NET, что предоставляет разработчикам обширную библиотеку классов и инструментов для создания различных типов приложений, включая веб-приложения, настольные приложения, мобильные приложения и игры. Изначально разработанный для Windows, с появлением .NET Core и более поздних версий, таких как .NET 5/6, C# стал кросс-платформенным языком, что позволяет разрабатывать приложения для Windows, macOS, Linux и других платформ. Асинхронное программирование с использованием ключевых слов `async` и `await` упрощает создание высокопроизводительных приложений, способных эффективно выполнять множество задач одновременно. C# также обеспечивает высокую степень интероперабельности, позволяя взаимодействовать с кодом, написанным на других языках, таких как C++, F#, и даже с кодом, работающим с неуправляемыми API Windows.

C# активно используется в различных областях программирования. В веб-разработке он применяется для создания высокопроизводительных и масштабируемых веб-приложений с помощью фреймворка ASP.NET. В разработке настольных приложений C# используется с такими технологиями, как Windows Forms и WPF (Windows Presentation Foundation), что позволяет создавать удобные и функциональные приложения для Windows. В игровой индустрии C# является основным языком для разработки игр на движке Unity, одном из самых популярных игровых движков, благодаря чему на языке создаются игры для различных платформ, включая ПК, консоли и мобильные устройства. Также C# широко используется в разработке облачных решений, особенно на платформе Microsoft Azure, что упрощает создание и развертывание облачных приложений. С использованием Xamarin, фреймворка на базе .NET, C# можно применять для создания мобильных приложений под Android и iOS, что позволяет разработчикам писать единый код и развертывать его на разных платформах.

C# обладает множеством преимуществ, включая активную поддержку со стороны Microsoft, богатую библиотеку классов и инструментов, а также отличную документацию и большое сообщество разработчиков. Однако, несмотря на все эти плюсы, язык тесно связан с экосистемой Microsoft, что может стать недостатком для тех, кто работает вне этой среды. Кроме того, из-за своей мощности и сложности C# может оказаться избыточным для простых или небольших проектов, где более легковесные языки могут быть предпочтительнее. В целом, C# – это мощный и универсальный язык программирования, который идеально подходит для разработки широкого спектра приложений, от небольших утилит до крупных корпоративных систем. [2]

## Unity

Unity – это популярный игровой движок, разработанный компанией Unity Technologies, который используется для создания видеоигр и других интерактивных приложений. Он был впервые выпущен в 2005 году и с тех пор стал одним из самых востребованных инструментов в индустрии разработки игр. Unity предоставляет разработчикам мощный набор инструментов и функций для создания игр, включая средства для работы с 2D и 3D графикой, физикой, анимацией, звуком и пользовательскими интерфейсами. Одним из ключевых преимуществ Unity является его кросс-платформенность – игры, созданные на этом движке, могут быть легко развернуты на различных платформах, таких как Windows, macOS, Linux, Android, iOS, PlayStation, Xbox, Nintendo Switch и даже в браузерах благодаря поддержке WebGL.

Unity известен своей простотой в использовании и доступностью, что делает его привлекательным как для новичков, так и для опытных разработчиков. Одной из особенностей движка является возможность визуального редактирования через Unity Editor, где разработчики могут создавать и настраивать сцены, объекты и их взаимодействия без необходимости писать код. Однако, для более сложных и кастомизированных проектов, Unity предоставляет мощный API, который можно использовать с языками программирования, такими как C# и JavaScript (поддержка последнего постепенно прекращается). C# стал основным языком для разработки на Unity, благодаря его интеграции с .NET и возможностям объектно-ориентированного программирования, что упрощает создание сложных игровых механик и систем.

Unity поддерживает множество современных технологий и методов разработки, таких как реалистичная физика, сложные системы освещения и теней, шейдеры, постобработка и многое другое. Движок также предлагает средства для работы с искусственным интеллектом, сетевыми функциями, а также интеграцию с различными внешними библиотеками и сервисами, что делает его гибким инструментом для создания разнообразных проектов.

Одним из значительных преимуществ Unity является его активное сообщество и обширная экосистема, включающая Unity Asset   
Store – платформу, где разработчики могут приобретать и продавать готовые ресурсы, такие как модели, текстуры, звуковые эффекты и даже готовые скрипты. Это позволяет значительно ускорить процесс разработки и сосредоточиться на уникальных аспектах игры, не тратя время на создание базовых элементов с нуля.

Unity широко используется не только в игровой индустрии, но и в других сферах, таких как архитектурная визуализация, разработка виртуальной и дополненной реальности (VR/AR), анимация и симуляции. Возможность создания интерактивных 3D-приложений для различных устройств и платформ сделала Unity популярным выбором среди разработчиков, работающих в этих областях. Движок также используется в образовательных целях, помогая студентам и начинающим разработчикам осваивать основы программирования и создания игр.

Благодаря своей универсальности, мощным возможностям и поддержке множества платформ, Unity стал одним из ведущих инструментов в разработке игр и интерактивных приложений, предлагая как начинающим, так и профессиональным разработчикам все необходимое для реализации своих идей. [3]

## Unity UI

Unity UI (User Interface) – это система, встроенная в Unity и позволяющая разработчикам создавать пользовательские интерфейсы для своих игр, приложений. Эта система предоставляет обширный набор инструментов для разработки и управления интерфейсами, включая кнопки, текстовые поля, изображения и другие элементы. Основной компонент для работы с интерфейсом в Unity – это Canvas. Он представляет собой область, на которой размещаются все UI-элементы. Canvas можно настроить для рендеринга на экране камеры, пространства мира или для рендеринга на экране с заданным масштабом.

Canvas работает в связке с другими важными компонентами, такими как RectTransform. RectTransform является расширением стандартного компонента Transform и используется для определения положения, размера и привязки UI-элементов на экране. С помощью RectTransform можно задавать, как элементы будут адаптироваться под разные размеры и соотношения сторон экрана, что особенно важно для создания кроссплатформенных интерфейсов.

Ещё одним ключевым аспектом Unity UI является система Event System, которая управляет взаимодействием пользователя с интерфейсом. Event System отвечает за обработку событий, таких как нажатия кнопок, ввод текста и перемещение элементов. Важной частью Event System является Input Module, который интерпретирует пользовательский ввод и превращает его в события, такие как Pointer Click или Drag.

Для создания и стилизации интерфейсов Unity предоставляет разнообразные UI-элементы, включая Text, Image, Button, Toggle, Slider и другие. Эти элементы можно комбинировать и настраивать для создания сложных и интерактивных интерфейсов. Например, Button может быть настроен так, чтобы реагировать на различные состояния, такие как нажатие, наведение курсора или его отпускание, с возможностью изменения визуального отображения в зависимости от состояния.

Unity UI также поддерживает работу с анимациями и переходами. Это позволяет анимировать UI-элементы, создавая плавные переходы между различными состояниями интерфейса. Визуальные эффекты могут значительно улучшить восприятие интерфейса пользователем, делая его более живым и динамичным.

Кроме того, Unity UI предоставляет мощные инструменты для адаптивного дизайна, позволяя UI-элементам автоматически изменять размер и положение в зависимости от разрешения экрана или ориентации устройства. Это достигается с помощью механизмов, таких как Anchor и Pivot, которые определяют, как элементы будут привязаны к родительскому RectTransform и как они будут масштабироваться.

В целом, Unity UI – это гибкая и мощная система, которая позволяет создавать интерфейсы различной сложности для игр и приложений, обеспечивая при этом удобство работы как для разработчиков, так и для конечных пользователей. [4]

## Unity Animator

Unity Animator – это система анимации, встроенная в Unity и позволяющая разработчикам создавать, управлять и воспроизводить анимации для различных объектов в игре. Система Animator используется для анимирования не только персонажей и объектов, но и интерфейсов, предоставляя обширные возможности для создания сложных и реалистичных движений.

В основе Unity Animator лежит компонент Animator, который привязывается к игровому объекту и управляет его анимациями. Этот компонент работает в связке с Animator Controller – активом, который задает логику переходов между различными анимациями и управляет их воспроизведением. Animator Controller содержит состояния (states), которые представляют собой отдельные анимации, и переходы (transitions), определяющие, когда и как объект переходит из одного состояния в другое. Переходы могут зависеть от различных параметров (parameters), в частности таких как числа, булевые значения, триггеры или векторы, которые задаются динамически в ходе игры.

С помощью Animator Controller можно создавать сложные анимационные схемы, включая ветвление логики, циклические анимации и последовательные переходы. Это особенно полезно при создании анимаций для персонажей, где, например, состояние «идти» может переходить в состояние «бежать», а затем в состояние «прыгать», в зависимости от действий игрока или условий в игре.

Animator также поддерживает использование Blend   
Trees – инструментов, которые позволяют плавно переходить между несколькими анимациями в зависимости от одного или нескольких параметров. Это позволяет создавать более реалистичные и плавные движения, например, переход от ходьбы к бегу в зависимости от скорости персонажа.

Еще одним важным аспектом Unity Animator является поддержка механики Inverse Kinematics (IK). С помощью IK можно сделать так, чтобы конечности персонажа, например, руки и ноги, автоматически двигались в нужное положение, основываясь на заданных целевых точках. Это упрощает создание реалистичных анимаций, таких как взаимодействие персонажа с объектами в мире или следование взглядом за определенной целью.

Animator также предоставляет инструменты для создания и редактирования анимаций прямо в Unity с помощью Animation Window. В этом окне можно вручную анимировать любые свойства объектов, создавая ключевые кадры (keyframes) и настраивая их параметры. Эта возможность позволяет напрямую в редакторе создавать и корректировать анимации, не прибегая к сторонним инструментам.

Кроме того, Unity Animator поддерживает слои анимации, что позволяет накладывать различные анимации друг на друга и комбинировать их. Это особенно полезно для персонажей, когда, например, основная анимация бега может быть дополнена анимацией стрельбы, играющей одновременно.

В целом, Unity Animator – это инструмент, предоставляющий огромные возможности для создания анимаций любой сложности. Он позволяет управлять анимациями с высокой степенью гибкости, создавая плавные и реалистичные движения, которые обогащают игровой процесс и делают его более захватывающим для игроков. [5]

## Git

Git – это распределенная система управления версиями, которая позволяет разработчикам отслеживать изменения в исходном коде и координировать работу над проектами. Созданный Линусом Торвальдсом в 2005 году, Git стал стандартом в индустрии программного обеспечения благодаря своей эффективности, гибкости и возможности поддержки командной работы.

Одной из ключевых особенностей Git является его распределенная природа. В отличие от централизованных систем управления версиями, таких как Subversion (SVN), в Git каждый разработчик имеет полную копию репозитория на своей локальной машине. Это означает, что все операции, такие как коммиты, ветвление и слияние, выполняются локально, без необходимости подключения к центральному серверу. Это повышает скорость работы и позволяет разработчикам работать офлайн, а затем синхронизировать свои изменения с другими участниками проекта, когда это необходимо.

Git хранит историю изменений в виде цепочки коммитов, где каждый коммит представляет собой снимок состояния проекта в определенный момент времени. Коммиты связаны между собой, образуя дерево изменений, что позволяет отслеживать и возвращаться к предыдущим версиям проекта при необходимости. Эта структура данных обеспечивает высокую целостность и надежность хранения изменений.

Одной из основных концепций Git является ветвление (branching). Ветки позволяют разработчикам работать над различными частями проекта одновременно, без риска нарушить основной код. Каждый разработчик может создать свою ветку для работы над новой функциональностью или исправлением ошибки, а затем объединить (слиять) свои изменения с основной веткой (чаще всего называемой «main» или «master») после завершения работы. Это упрощает процесс интеграции новых функций и исправлений, а также поддерживает чистоту и стабильность основной версии проекта.

Git также предоставляет мощные инструменты для слияния веток и разрешения конфликтов. Когда несколько разработчиков вносят изменения в одни и те же части кода, могут возникать конфликты, которые нужно разрешать вручную. Git предлагает различные стратегии слияния и инструменты для автоматического и ручного разрешения конфликтов, что упрощает процесс интеграции изменений.

Важной частью экосистемы Git является поддержка удаленных репозиториев, которые позволяют разработчикам обмениваться изменениями и работать совместно. Платформы, такие как GitHub, GitLab и Bitbucket, предоставляют хостинг для Git-репозиториев и дополнительные функции для управления проектами, такие как системы отслеживания задач, обзоры кода и непрерывная интеграция. Эти платформы способствуют лучшему взаимодействию между членами команды и помогают организовать рабочие процессы.

Git также поддерживает ряд инструментов и команд для управления репозиториями, такими как «git clone» для создания локальной копии репозитория, «git pull» для получения обновлений из удаленного репозитория, «git push» для отправки изменений в удаленный репозиторий, «git status» для проверки состояния рабочего дерева, и «git log» для просмотра истории коммитов. Эти команды позволяют эффективно управлять проектом и следить за его развитием.

Одним из больших преимуществ Git является его гибкость и мощные возможности, которые позволяют адаптировать систему управления версиями под различные рабочие процессы и стили разработки. Его высокая скорость, распределенная природа и поддержка ветвления делают его отличным инструментом для как малых, так и крупных проектов, а также для команд любой размерности. Git стал стандартом де-факто для управления версиями в разработке программного обеспечения и имеет огромное сообщество пользователей и разработчиков, которые способствуют его дальнейшему развитию и улучшению. [6]

## Firebase

Firebase – это платформа для разработки мобильных и веб-приложений, предоставляемая компанией Google. Она была приобретена Google в 2014 году и с тех пор значительно расширилась и улучшилась, став одной из самых популярных платформ для разработки приложений благодаря своему широкому набору инструментов и услуг. Firebase предназначен для упрощения разработки приложений и ускорения процесса выхода на рынок, предлагая разработчикам интегрированные решения для самых разных задач.

Одним из ключевых компонентов Firebase является Firebase Realtime Database. Это облачная база данных, которая позволяет хранить и синхронизировать данные между клиентами в реальном времени. Это значит, что любые изменения данных автоматически обновляются на всех подключенных устройствах, что идеально подходит для приложений с активным пользовательским взаимодействием, таких как чаты или социальные сети. Firebase Realtime Database поддерживает масштабирование и позволяет разработчикам легко работать с данными без необходимости в сложной настройке серверной части.

Еще одним важным компонентом является Firebase Firestore – новая версия базы данных, которая предлагает более сложные функции по сравнению с Realtime Database. Firestore предоставляет гибкую, масштабируемую базу данных с поддержкой запросов, транзакций и оффлайн-доступа. Она лучше подходит для сложных приложений с богатой функциональностью и сложными структурами данных.

Для аутентификации пользователей Firebase предлагает Firebase Authentication, который упрощает управление пользователями и их регистрацией. С помощью этого сервиса можно реализовать регистрацию и вход через электронную почту и пароль, а также через социальные сети, такие как Google, Facebook и Twitter. Firebase Authentication интегрируется с другими сервисами Firebase и предоставляет средства для защиты данных пользователей и управления их сеансами.

Firebase Cloud Messaging (FCM) позволяет отправлять уведомления и сообщения пользователям на устройствах iOS, Android и веб-браузерах. Это решение идеально подходит для создания приложений с функцией отправки пуш-уведомлений и маркетинговых кампаний. FCM поддерживает как простые уведомления, так и сообщения с данными, что позволяет реализовывать более сложные сценарии взаимодействия с пользователями.

Firebase Hosting предоставляет возможность хостинга статических сайтов и приложений с использованием простого и надежного решения. Это включает поддержку SSL, быструю доставку контента через CDN и возможность легкого развертывания с помощью командной строки или интеграции с другими сервисами Firebase.

Firebase Analytics, также известный как Google Analytics для Firebase, позволяет отслеживать поведение пользователей и анализировать взаимодействие с приложением. Он предоставляет подробные отчеты и аналитику, что помогает разработчикам лучше понимать, как пользователи взаимодействуют с приложением и какие функции наиболее популярны. Эти данные помогают принимать обоснованные решения для улучшения приложения и повышения его эффективности.

Для разработки и тестирования приложений Firebase предлагает Firebase Test Lab, который позволяет тестировать приложение на различных устройствах и конфигурациях в облаке. Это позволяет находить и исправлять ошибки на разных устройствах, не требуя наличия физического доступа к ним.

Firebase Performance Monitoring предоставляет инструменты для отслеживания производительности приложения и выявления узких мест, таких как медленные запросы или долгие загрузки. Эти данные помогают оптимизировать приложение и обеспечить более плавную работу для конечных пользователей.

Firebase Crashlytics – это инструмент для отслеживания и анализа сбоев в приложении. Он предоставляет подробную информацию о причинах сбоев, что упрощает процесс их устранения и улучшает стабильность приложения.

В целом, Firebase предлагает комплексное решение для разработки, тестирования и мониторинга мобильных и веб-приложений, объединяя в себе разнообразные инструменты и услуги, которые упрощают процесс разработки и обеспечивают высокое качество конечного продукта. Благодаря интеграции с экосистемой Google и широким возможностям настройки, Firebase становится ценным инструментом для разработчиков, стремящихся создать современные и функциональные приложения. [7]

# Индивидуальное задание

Целью данного индивидуального задания является разработка и интеграция новых функциональных возможностей и контента в существующую систему, обеспечивая улучшение пользовательского опыта и расширение функциональности приложения. Цель включает в себя следующие задачи:

– Имплементация навыка персонажа;

– Имплементация экрана покупки навыка;

– Добавление возможности удаленного включения или выключения функционала;

– Имплементация индикаторов отслеживания объектов нового типа.

Выполнение этих задач позволит приобрести опыт работы с реальным проектом в команде с другими разработчиками, поможет развить навыки в интеграции новых функций и контента, управлении приложениями, а также работе с визуализацией данных и настройкой функционала.

## Имплементация навыка персонажа

В рамках задачи был реализован навык для персонажа Dead X, позволяющий ему стрелять в бочку, стоящую на локации и разрушать ее.

В первую очередь была реализован поиск бочки в пределах определенной дистанции вокруг персонажа, код которого представлен на рисунке 3.1.



Рисунок 3.1 – Поиск бочки поблизости

Далее необходимо было реализовать проверку на доступность бочки для атаки, код которой представлен на рисунке 3.2.

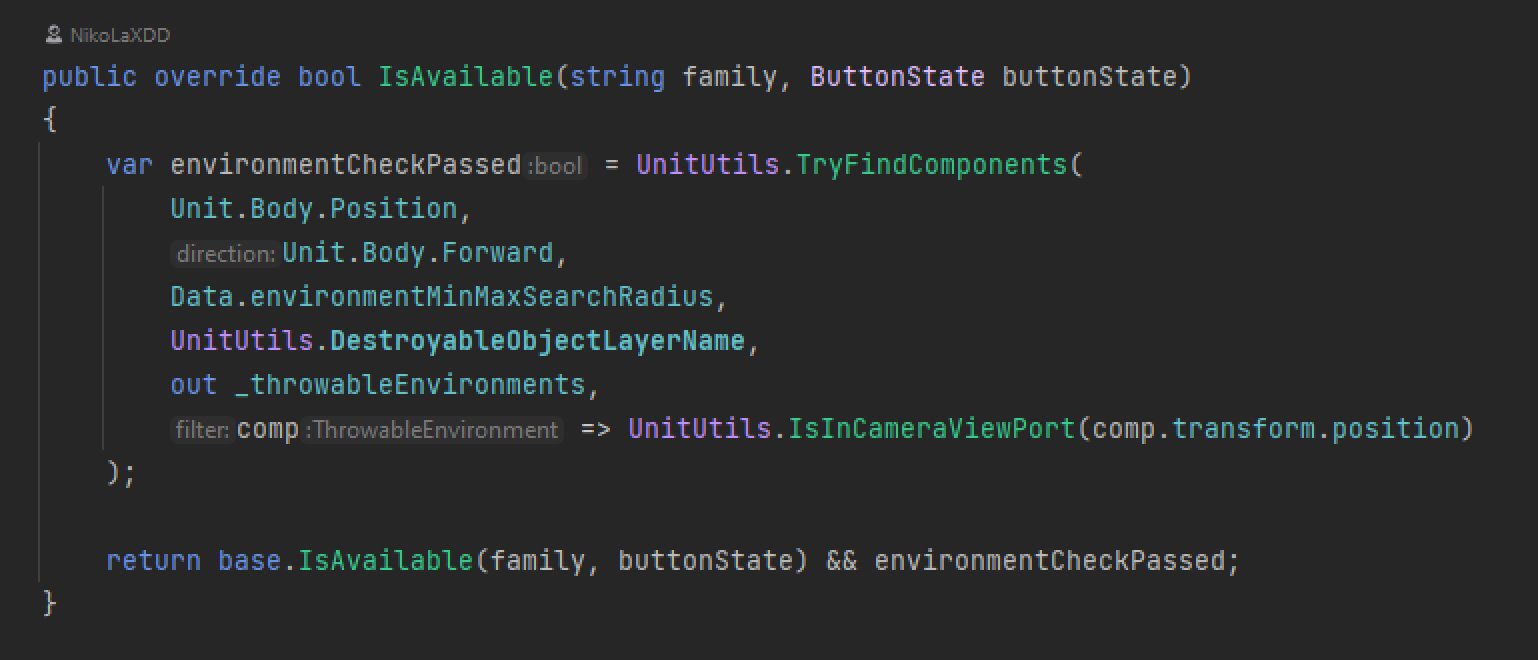


Рисунок 3.2 – Проверка доступности бочки

После этого необходимо было реализовать основной поток навыка, код представлен на рисунке 3.3, связывающий воедино логику работы навыка.



Рисунок 3.3 – Основной поток навыка

Еще одним важным элементом была инициализация выстрела, рассчитывающая траекторию полета снаряда, его скорость, код реализации представлен на рисунке 3.4.

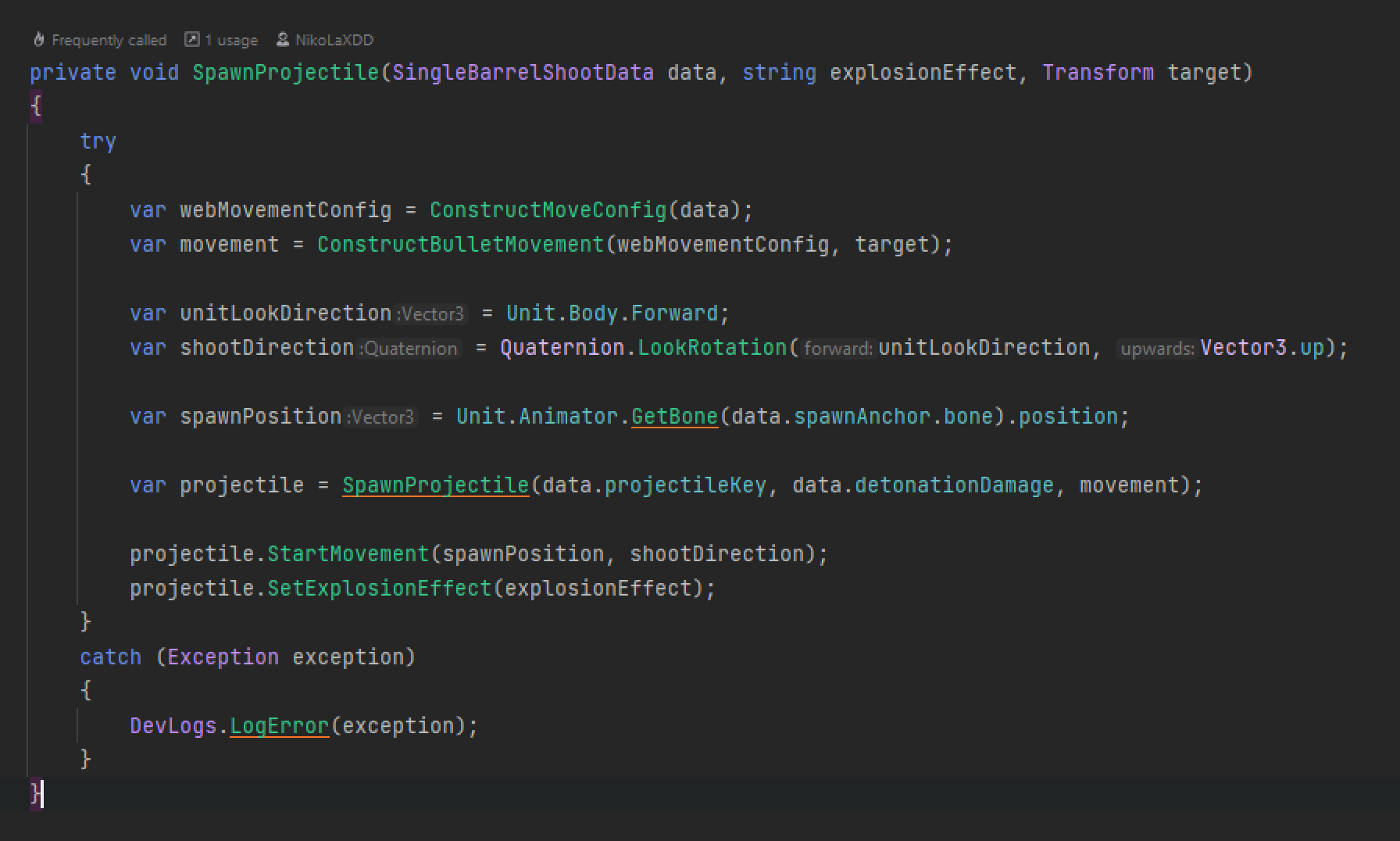


Рисунок 3.4 – Инициализация выстрела

Необходимо было использовать определенный звук выстрела, код представлен на рисунке 3.5.

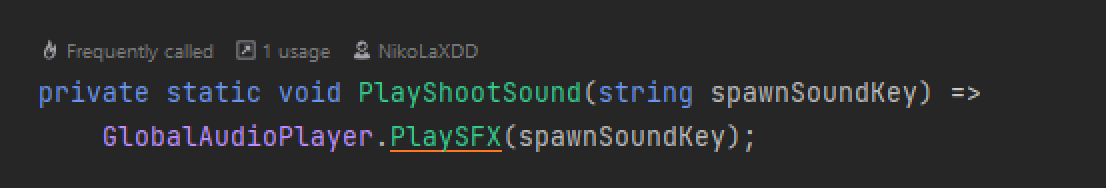


Рисунок 3.5 – Имплементация звука выстрела

Результат работы навыка представлен на рисунке 3.6.



Рисунок 3.6 – Результат работы навыка

По итогу разработки в рамке задачи один из персонажей игры получил возможность стрелять по бочкам, представляющим собой объекты, расставленные на локации.

## Имплементация экрана покупки навыка

В рамках задачи был реализован экран, который появляется перед игроком и позволяет ему приобрести навык за игровую валюту.

Экран связывается с моделью, то есть логикой того, что должно случиться в игре после нажатия на экран, через медиатор. Эта связь основана на принципах MVC.

При запуске экрана происходит заполнение его содержимого необходимой информацией о первом в списке на покупку навыке – название, иконка, превью видео, стоимость, а также о балансе игровой валюты у игрока. Код, выполняющийся при запуске экрана, представлен на рисунке 3.7.

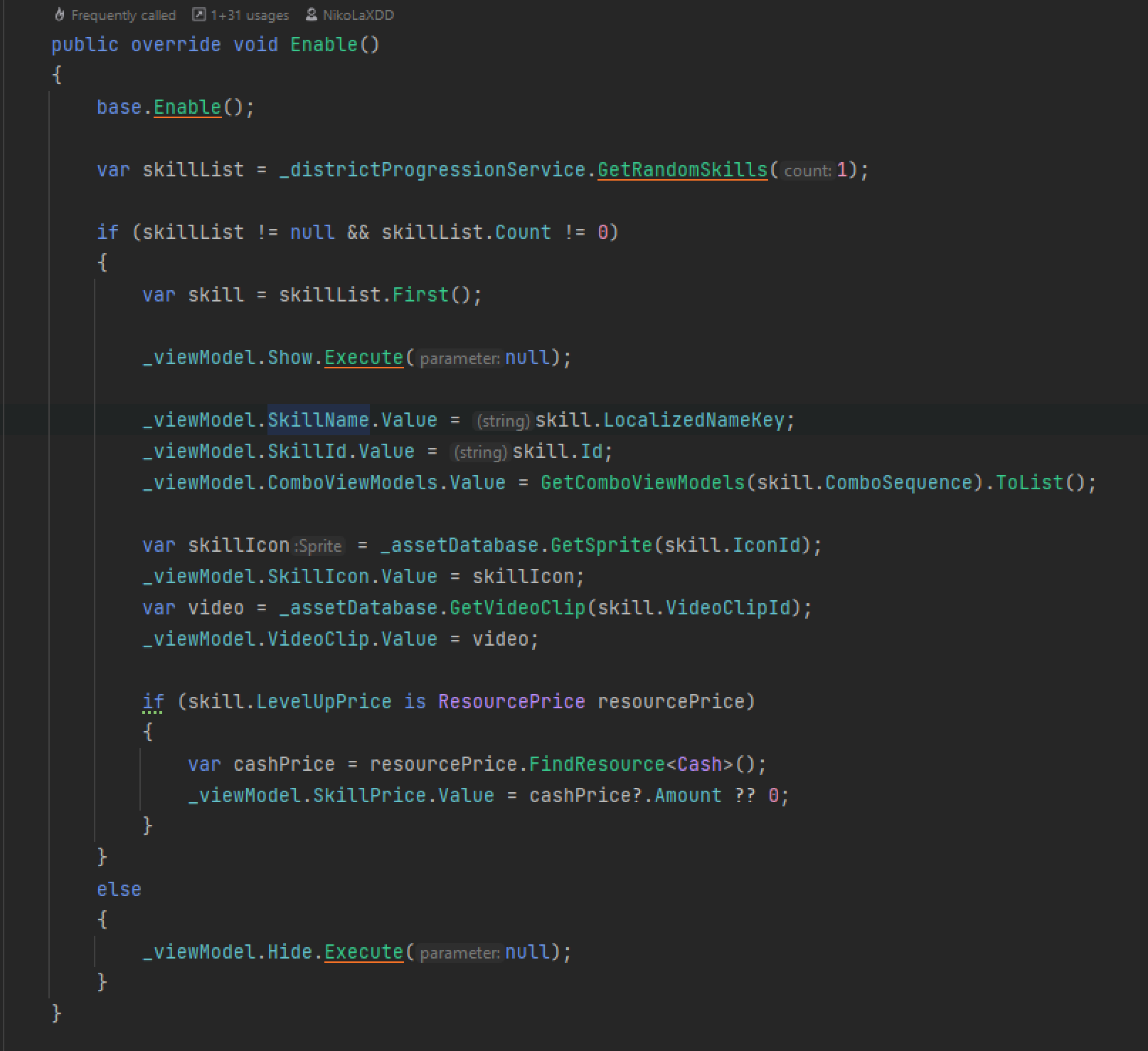


Рисунок 3.7 – Метод, выполняемый при запуске экрана

При нажатии на кнопку в центре экрана вызывается метод покупки навыка, код которого представлен на рисунке 3.8. Если у игрока хватает игровой валюты, чтобы купить навык, то тот приобретается, если нет, то игрока перенаправляет в магазин.



Рисунок 3.8 – Метод покупки навыка

Для того, чтобы видео превью навыка воспроизводилось на экране беспрерывно, необходимо было запустить его и зациклить его воспроизведение. Код соответствующего метода представлен на рисунке 3.9.

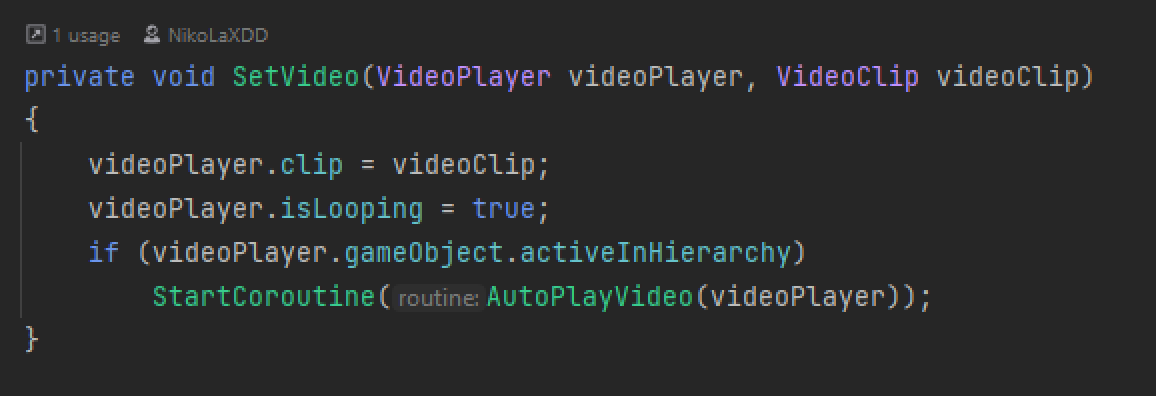


Рисунок 3.9 – Метод запуска превью видео

Сам экран был сверстан по предоставленному в техническом задании макету с использованием Unity UI, а анимация его появления, также по макету, была сделана с использованием Unity Animator. Результат работы экрана покупки навыка представлен на рисунке 3.10.



Рисунок 3.10 – Результат работы экрана покупки навыка

По итогу разработки в рамках задачи был сделан появляющийся перед игроком экран с возможностью покупки навыка на нём. На экране игрок может увидеть всю необходимую информацию о покупаемом навыке: его иконку, название, стоимость, превью, а также баланс игровой валюты у игрока.

## Возможность удаленного включения или выключения  функционала

В компании Суперлевел практикуются эксперименты с геймплеем, которые проводятся на новых игроках, устанавливающих игру. Для этого используются возможности Firebase с проведением А/В тестов и возможность приложения принимать значения из конфигураций, отправляемых на устройство игрока удаленно через конфигурации Firebase.

Необходимо было реализовать в проекте возможность удаленного включения и выключения выпадения нового экспериментального типа   
наград – чемоданов. Код реализации представлен на рисунке 3.11.

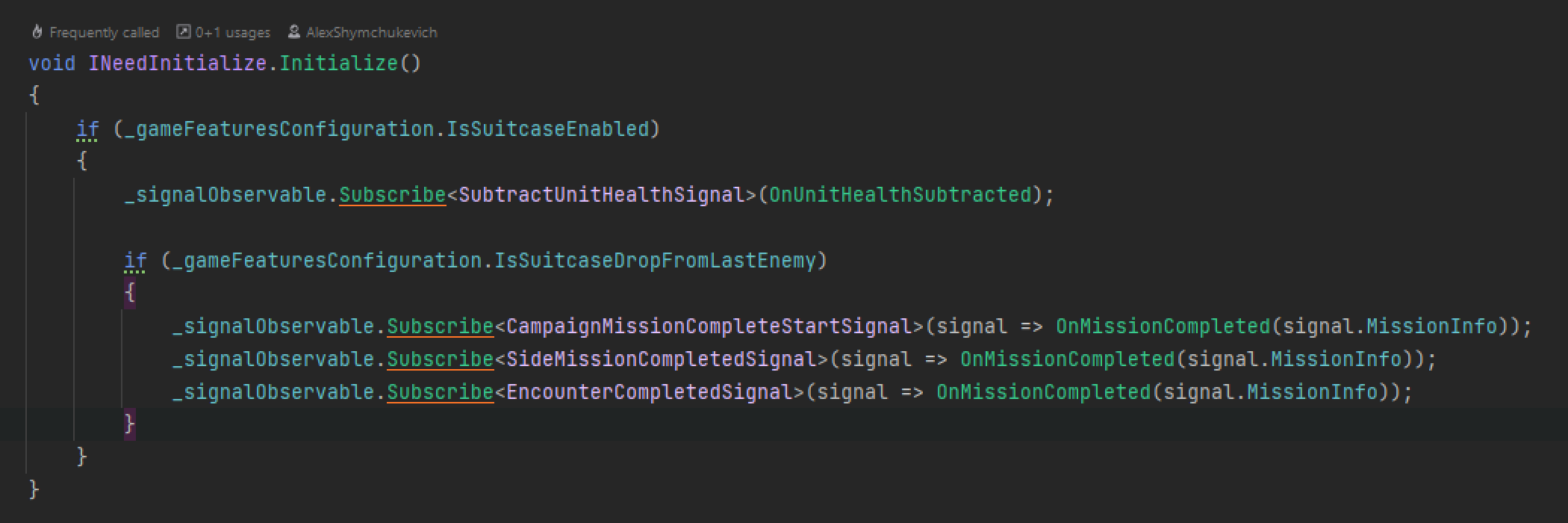


Рисунок 3.11 – Метод, позволяющий удаленно включать или выключать выпадение чемоданов

В результате разработки в рамках задачи была реализована возможность удаленного включения и выключения выпадения нового типа наград в игре.

## Имплементация индикаторов отслеживания объектов нового типа

Для индикаторов отслеживания объектов нового типа, а именно чемоданов, необходимо было сделать не только логику отслеживания, но и также конфигурацию, которую могли бы изменять не только программисты через код, но геймдизайнеры и UI художники, которым нужна удобная визуализация настроек индикатора.

Конфигурация, реализованная в результате этого условия задачи, представлена на рисунке 3.12.

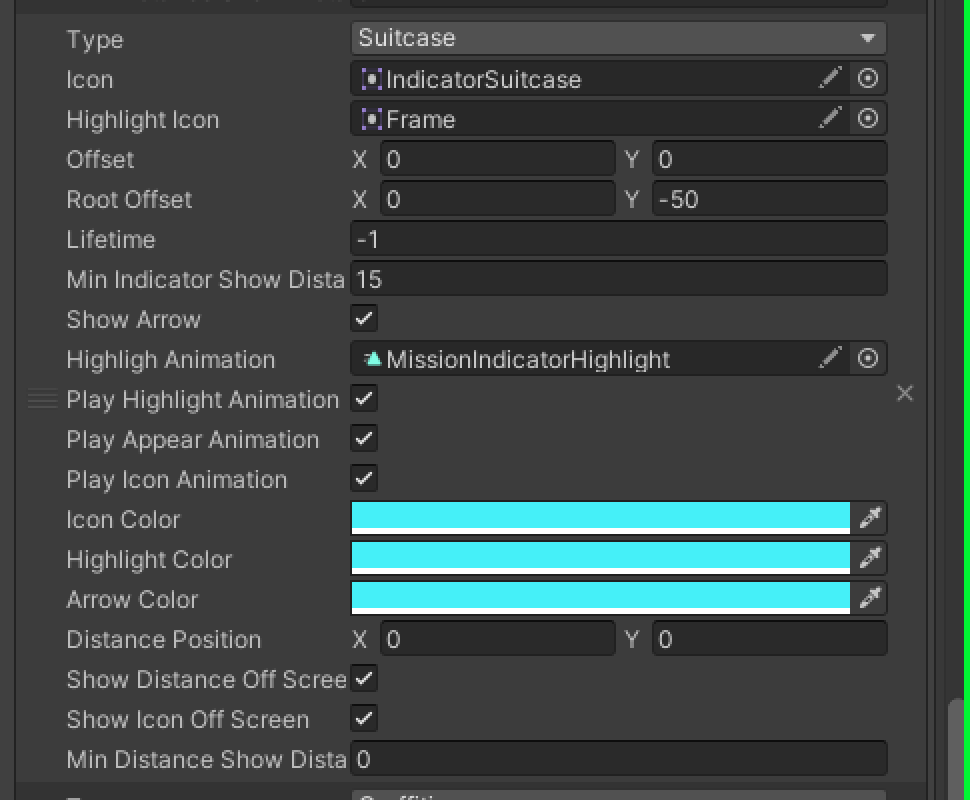


Рисунок 3.12 – Конфигурация настроек индикатора чемодана

Код, из которого собирается эта конфигурация, представлен на рисунке 3.13.

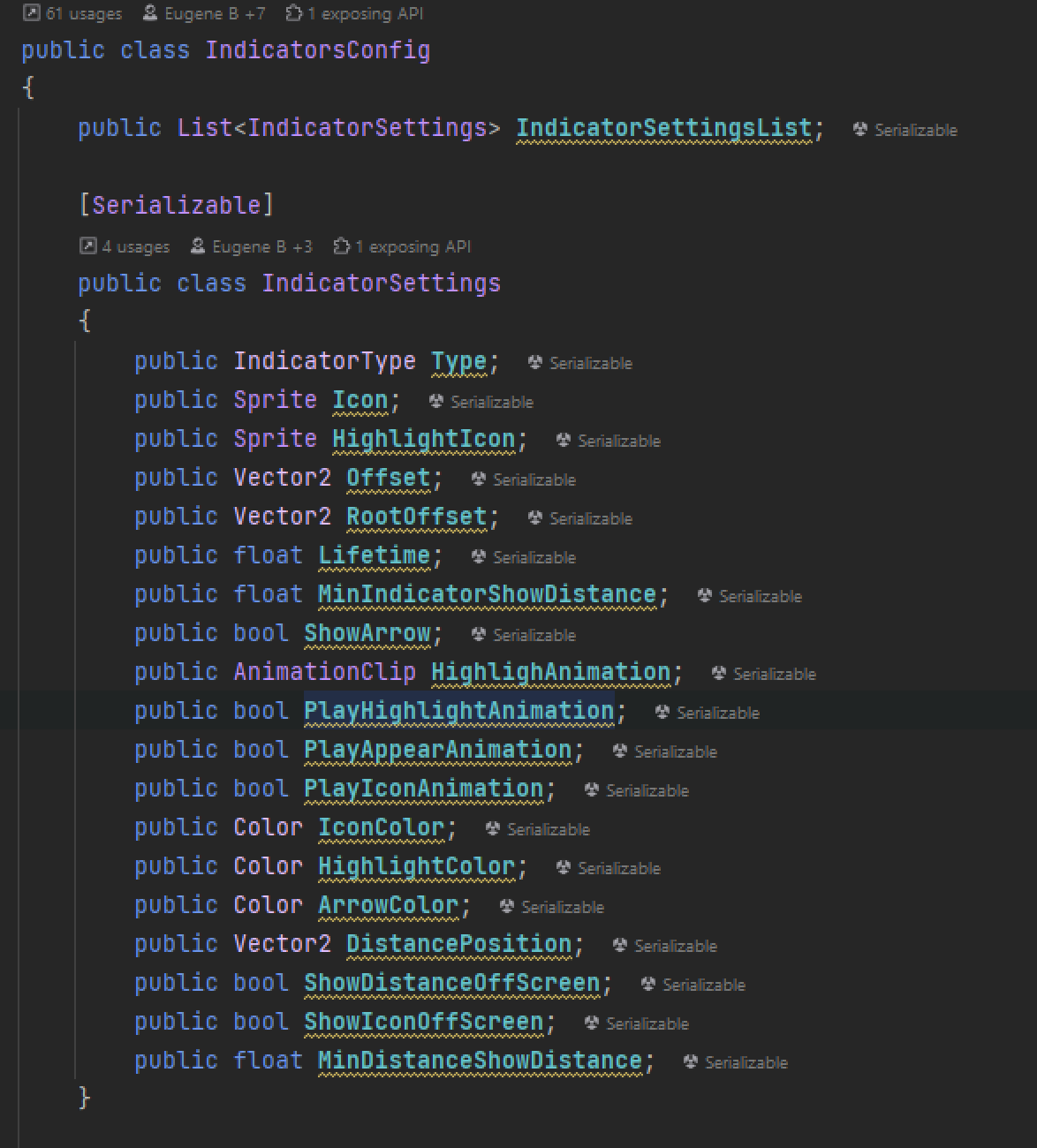


Рисунок 3.13 – Код для конфигурации настроек индикатора чемодана

Для визуального представления необходимо было выставить иконку чемодана и таймер с соответствующим значением под ним. Код представлен на рисунке 3.14.

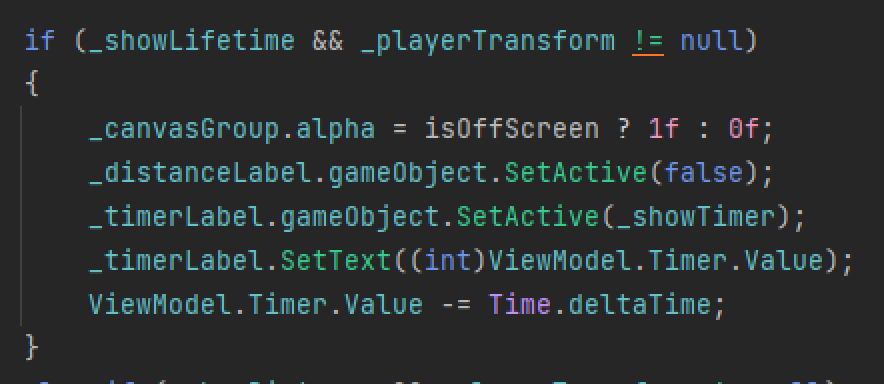


Рисунок 3.14 – Код для конфигурации настроек индикатора чемодана

Код создания объекта индикатора представлен на рисунке 3.15.

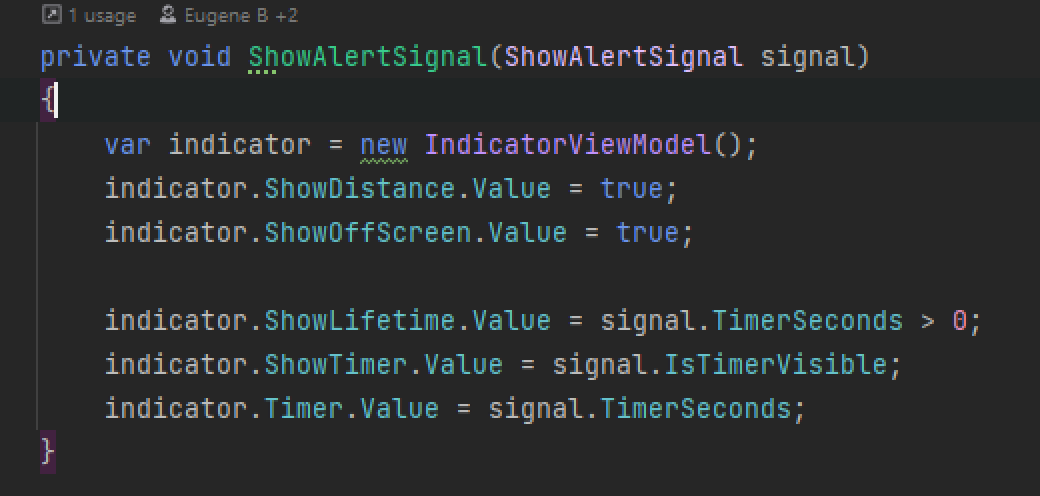


Рисунок 3.15 – Код создания объекта индикатора

Также все индикаторы в игре перемещаются по UI в зависимости от того, в каком направлении от игрока находится объект, на который они указывают. Эта логика для индикатора чемодана представлена на рисунке 3.16.

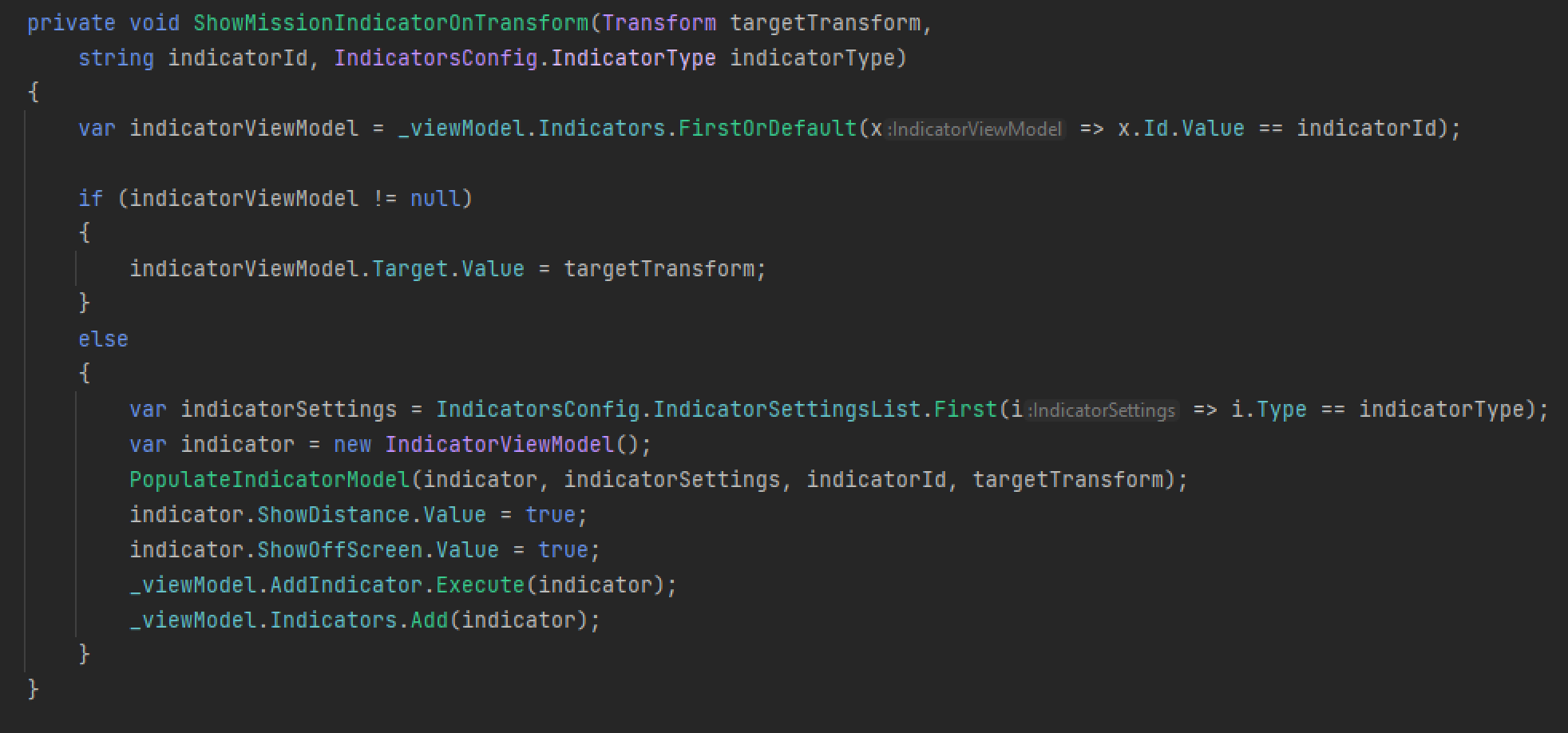


Рисунок 3.16 – Код изменения положения индикатора на экране

В результате на экране начал отображаться голубой индикатор, указывающий на положение чемодана относительно игрока и показывающий таймер, через который этот чемодан исчезнет. Результат представлен на рисунке 3.17.



Рисунок 3.17 – Индикатор отслеживания чемодана

По итогу разработки в рамках задачи были реализованы индикатор для отслеживания объекта нового типа и конфигурация для его настройки.

Заключение

Производственная практика в ООО «Суперлевел» предоставила уникальную возможность погрузиться в процесс разработки мобильных игр и познакомиться с современной IT-индустрией изнутри. В ходе практики была изучена структура компании, технологии и методики, используемые в разработке мобильных приложений, а также особенности функционирования проектов в условиях динамично развивающейся среды.

В рамках индивидуального задания была проведена работа над улучшением существующего проекта компании, а именно удалось получить ценные практические навыки, связанные с разработкой и интеграцией новых функциональных возможностей в существующую систему. В частности, были выполнены задачи по имплементации навыка персонажа, созданию экрана для его покупки и добавлению возможностей удаленного управления функционалом. Также была реализована система индикаторов для отслеживания объектов нового типа. Эти задачи позволили углубить знания в области программирования, научиться эффективно работать с проектной документацией, а также лучше понять процессы командной разработки.

Производственная практика дала возможность применить в реальных условиях теоретические знания, полученные в ходе обучения, вследствие чего значительно укрепить свои профессиональные компетенции. Работа в команде с опытными разработчиками позволила изучить лучшие практики разработки, тестирования и внедрения новых функций, а также освоить навыки взаимодействия с коллегами для успешного выполнения поставленных задач и достижения общих целей.

В итоге, практика в ООО «Суперлевел» стала важным этапом в профессиональном развитии, предоставив ценные знания и опыт, которые будут полезны в дальнейшей карьере в сфере разработки мобильных игр и информационных технологий. Освоенные навыки и приобретенные компетенции создают прочную основу для успешного роста и дальнейшего продвижения в профессиональной области.

Список литературных источников

1. СТП 01-2024 – Стандарт предприятия. Дипломные проекты (работы) общие требования. – Минск: БГУИР, 2024. – 175с.
2. Основные функции и объекты СУБД [Электронный ресурс] – Режим   
   доступа : https://sky.pro/wiki/sql/osnovnye-funkcii-i-obuekty-subd/. – Дата доступа : 15.09.2024
3. Классификация СУБД [Электронный ресурс] – Режим доступа : https://lms.kgeu.ru/mod/book/view.php?id=77952&chapterid=3854. – Дата доступа : 16.09.2024
4. Unity UI [Электронный ресурс] – Режим доступа : https://docs.unity3d.com/Manual/UIToolkits.html. – Дата доступа : 01.07.2024
5. Unity Animator [Электронный ресурс] – Режим доступа : https://docs.unity3d.com/ScriptReference/Animator. – Дата доступа : 02.07.2024
6. Git [Электронный ресурс] – Режим доступа : https://www.atlassian.com/ru/git/tutorials/what-is-git. – Дата доступа : 02.07.2024
7. Firebase [Электронный ресурс] – Режим доступа : https://kolmogorov.pro/what-is-firebase-chto-takoe. – Дата доступа : 02.07.2024