Федеральное государственное автономное
образовательное учреждение
Высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра вычислительной техники
кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В. Непомнящий

«____» _____2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника
код – наименование направления

Игра в жанре «Tower defense»
тема

Руководитель _____ В. С. Васильев
подпись, дата

Выпускник _____ С. А. Спиридонов
подпись, дата

Нормоконтролер _____ В. С. Васильев
подпись, дата

Красноярск 2022

Федеральное государственное автономное
образовательное учреждение
Высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра вычислительной техники
кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В. Непомнящий

«____» _____2022 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Спиридонову Сергею Александровичу

Группа КИ18-08Б

Направление 09.03.01 Информатика и вычислительная техника

Тема выпускной работы: Игра в жанре «Tower Defense»

Утверждена приказом по университету №7914/С от 26.05.22

Руководитель ВКР: В.С. Васильев, старший преподаватель.

Перечень разделов ВКР:

1. Разработка спецификации требований.
2. Проектирование.
3. Реализация и документация.

Перечень графического материала: нет

Руководитель _____ Васильев В.С.
подпись, дата

Задание принял к исполнению _____ Спиридонов С.А.
подпись, дата

«____» _____2021 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «игра в жанре «Tower Defense»» содержит 44 страницы текстового документа, 8 таблиц, 24 рисунка, 25 использованных источников.

ИГРА, UNITY, ANDROID, TOWER DEFENSE, СТРАТЕГИЯ, КЛИЕНТ-СЕРВЕР.

Объект выпускной работы: процесс создания программного продукта.

Предмет выпускной квалификационной работы: разработка приложения для устройств на ОС Android.

Целью данной выпускной квалификационной работы является разработка игры в жанре «Tower Defense» с использованием клиент-серверной архитектуры для мобильных устройств на ОС Android.

Выпускная квалификационная работа выполнена в соответствии с индивидуальным заданием. В процессе выполнения решены задачи:

1. Выполнен анализ предметной области проекта
2. Сформировано техническое задание на проект
3. Проанализирован выбор программных средств разработки для мобильных устройств на ОС Android.
4. Создан прототип мобильного приложения.
5. Выполнено тестирование приложения

СОДЕРЖАНИЕ

Введение

В 2021 году 40 процентов всей выручки цифровых игр пришлось на мобильный сегмент [1], разработка мобильных игр является актуальной задачей. Одним из популярных жанров является «Tower defense», суть которого заключается в расстановке башен по определенным правилам и уничтожении с их помощью объектов противника. Ведется исследование различных аспектов этой игры, например анализ поведения игрока [2], разработка адаптивного искусственного интеллекта [3] и алгоритмов автоматического создания игровых карт [4]. Таким образом, работа является актуальной.

Существует множество различных игр в этом жанре, отличающихся развитостью искусственного интеллекта, мультиплеера, типами доступных башен, разновидностью объектов противника и так далее.

Цель работы – создание клиент-серверной версии игры жанра «Tower defense».

Структура работы отражает **решаемые задачи:**

- в первой главе приведены:
  1) анализ игрового процесса;
  2) анализ сильных и слабых сторон широко известных реализаций;
  3) анализ реализаций с открытым исходным кодом на предмет возможности его заимствования;
  4) спецификация требований к разрабатываемой системе, учитывающая результаты проведенных анализов.
- в рамках выполнения второй главы выполнено проектирование системы;
- в третьей главе описаны особенности игры, а также приведены инструкции по сборке и развертыванию системы.

1 Разработка спецификации требований

Для выявления сильных сторон, проведен анализ аналогичных мобильных приложений, а также открытого исходного кода. С учетом этого, разработана спецификация требований к создаваемой системе.

1.1 Анализ сильных и слабых сторон широко известных реализаций

В таблице 1 приведены примеры популярных игр в жанре «Tower defense».

Таблица 1 – Обзор сильных и слабых сторон

Характеристика	Clash Royale	Grow Castle – Tower Defense	Kingdom Rush Origins - TD
Анимации, звуки	есть	есть	есть
Возможность игры онлайн	есть	нет	нет
Возможность добавление друзей	есть	есть	нет
Доступ к игре на платной основе	нет	нет	да
Монетизация (покупки в приложении)	есть	есть	нет
Реклама	есть	есть	нет

1.2 Анализ реализаций с открытым исходным кодом на предмет возможности его заимствования

По состоянию на 24.12.2021 на сервисе Github [8] по запросу «Tower Defense» найдено 209 проектов, написанных на разных языках программирования, из которых 53 – на C#, 37 на Java. Популярность языка C# объясняется движоком Unity, который использует этот язык. Unity крайне удобен в проектах с большим количеством интерактива. Результаты анализа наиболее популярных из них приведены в таблице 2.

Таблица 2 – Основные показатели приложений на Github

Название	castle-game	ArchoMage HD	The Last Stand	towerDefense
Язык	Rust	TypeScript	Java	JavaScript
Наличие серверной части	Отсутствует	Отсутствует	Отсутствует	Отсутствует
Лицензия	PPL-based	MIT	GPL-3.0	MIT
Удалось запустить	Да	Да	Да	Да
Документация	Да	Да	Нет	Да
Время последнего обновления	5 месяцев назад	2 месяца назад	3 года назад	17 месяцев назад
Количество разработчиков	2	1	1	1
Количество звезд	82	39	3	8

Из таблицы видно, что ни одно из приложений с открытым исходным кодом не реализует игру с клиент-серверной архитектурой.

По результатам просмотра аналогов, принято решение реализовать в разрабатываемой системе поддержку классического режима добавив свои идеи и реализовав клиент-серверной архитектуру.

1.3 Анализ игрового процесса

Игрок может покупать юнитов кликая на их иконки, расположенные в левой и правой части экрана. Купленные юниты имеют возможность ближнего и дальнего боя. За уничтожение юнита соперника, игрок получает игровую валюту, на которое может купить дополнительных юнитов. В игровой сессии имеется две стадии игры: стадия боя и стадия планирования. Во время первой стадии разрешается покупка юнитов. Во время второй стадии разрешается покупка усилений характеристик. Стадии заканчивается при достижении таймером отметки 0. На рисунках 1 и 2 подписаны элементы игровых стадий.

Рисунок 1 – Стадия боя

Рисунок 2 – Стадия планирования

1.4 Спецификация требований к разрабатываемой системе, учитывающая результаты проведенных анализов

Функциональные требования выражены в виде диаграмм вариантов использования на рисунках 3 и 4. [9]

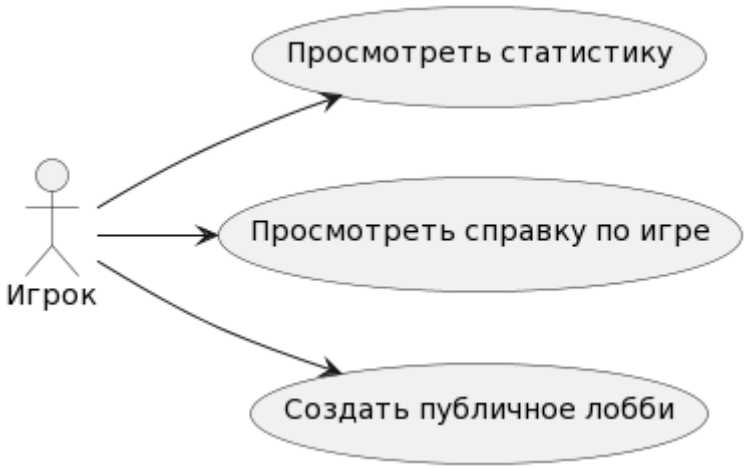

Рисунок 3 – Диаграмма вариантов использования (часть 1)

Рисунок 4 – Диаграмма вариантов использования (часть 2)

1.4.1 Список прецедентов программы

Название прецедента: просмотреть статистику.

Предусловия: пользователь находится в окне “Главное меню”, нажал на кнопку “Статистика”.

Основной сценарий:

1. Из подкаченного с сервера файла JSON парсятся параметры побед и поражений

2. Данные выводятся на экран.

Условие ввода в действие альтернативных сценариев

Альтернативный сценарий (не удается поулчить данные с сервер о пользователе)

Всплывает окно с содержанием: “Невозможно загрузить статистику, так как сервер не доступен. Попробуйте перезапустить игру”.

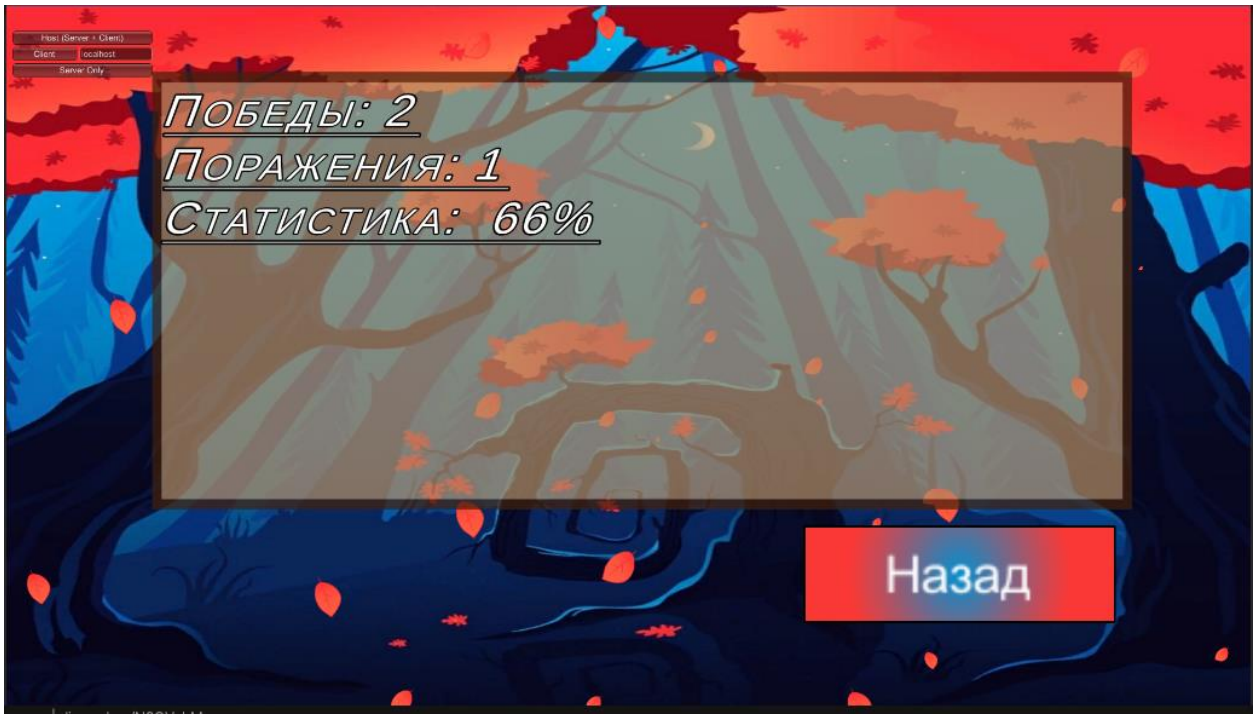

Рисунок 5 – Экран просмотра статистики

Название прецедента: просмотр справки по игре.

Предусловия: пользователь находится в окне “Главное меню”, нажал на кнопку “Галерея”.

Основной сценарий:

Из корневого файла парсятся данные о юнитах.

Условие ввода в действие альтернативных сценариев

Альтернативный сценарий 1 (Отсутствует файл данных)

Всплывает оповещение с надписью: “Произошла ошибка. Отсутствует загружаемый файл”.

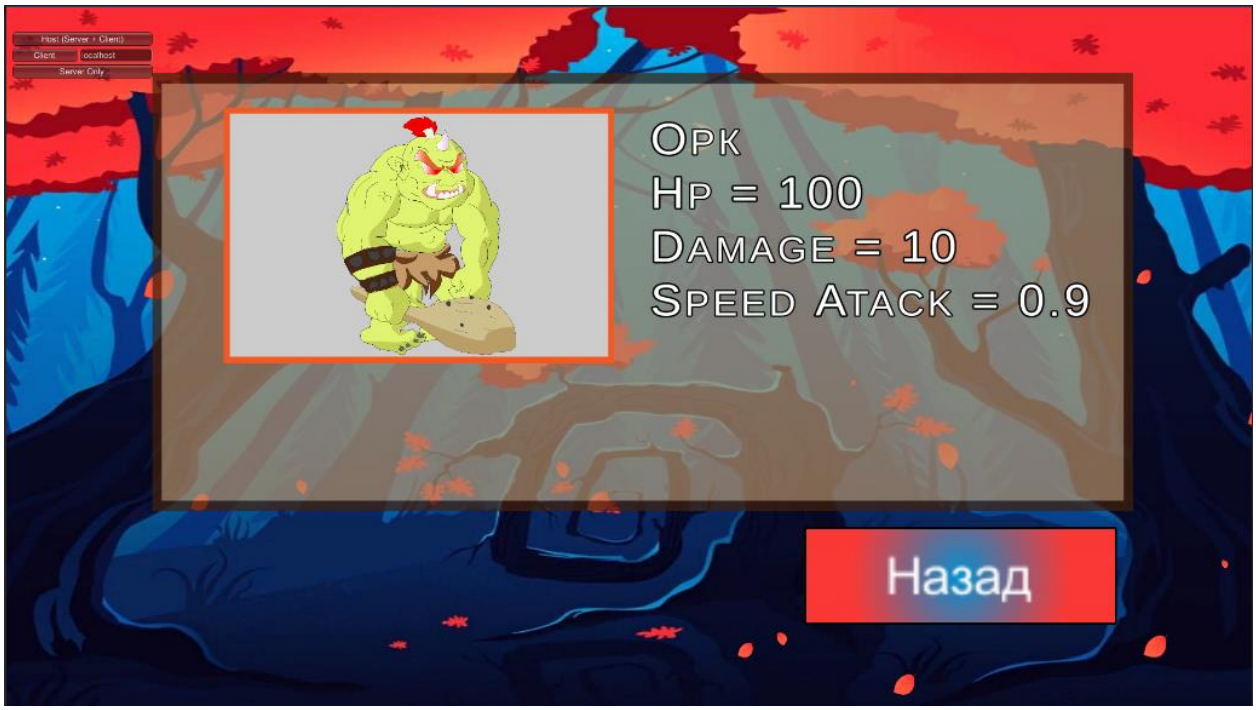

Рисунок 6 – Экран просмотра информации об юнитах

Название прецедента: создать публичное лобби.

Предусловия: пользователь находится в окне “Главное меню”, нажал на кнопку “Создать публичную игру”.

Основной сценарий:

На сервере создается лобби, которое имеет уникальным идентификатором состоящий из 5 символов (латинских букв и цифр) (рисунок 7).

Условие ввода в действие альтернативных сценариев

Альтернативный сценарий (не удалось создать лобби)

1. Менеджер экранов возвращает в окно «Главное меню».

2. Появляется окошко с надписью “Ошибка создания игры. Проверьте подключение к интернету”

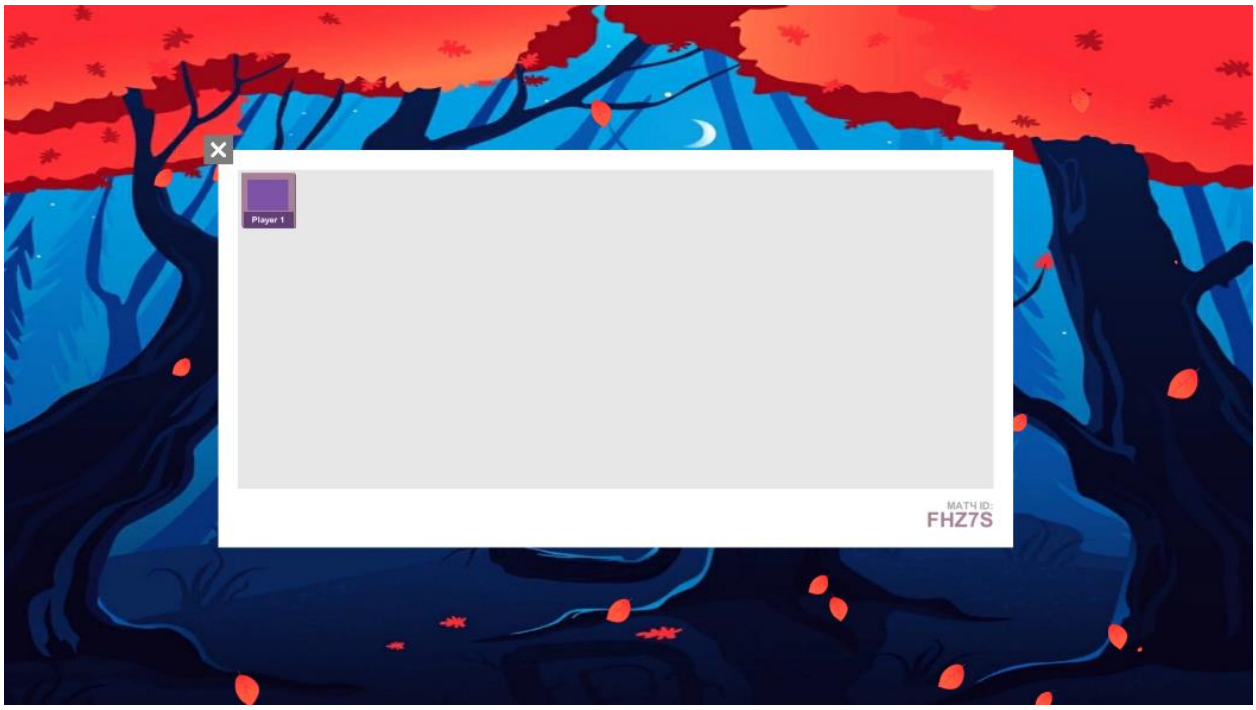

Рисунок 7 – Окно лобби

Название прецедента: создать приватное лобби.

Предусловия: пользователь находится в окне “Главное меню”, нажал на кнопку “Создать приватную игру”.

Основной сценарий:

На сервере создается лобби, которое имеет уникальным идентификатором состоящий из 5 символов (латинских букв и цифр).

Условие ввода в действие альтернативных сценариев

Альтернативный сценарий (не удалось создать лобби)

Менеджер экранов возвращает в стартовое меню. Появляется окошко с надписью “Ошибка создания игры. Проверьте подключение к интернету”

Название прецедента: найти случайное лобби.

Предусловия: пользователь находится в окне “Главное меню”, нажал на кнопку “Поиск игры”.

Основной сценарий:

1. Пользователь переходит в окно “Поиск игры” (рисунок 8).
2. На сервере проверяется список доступных лобби для присоединения.
3. Поиск продолжается, пока не будет нажата кнопка “Отмена”
4. Открывается окно “Лобби” с присутствующим хостом лобби (рисунок 9).

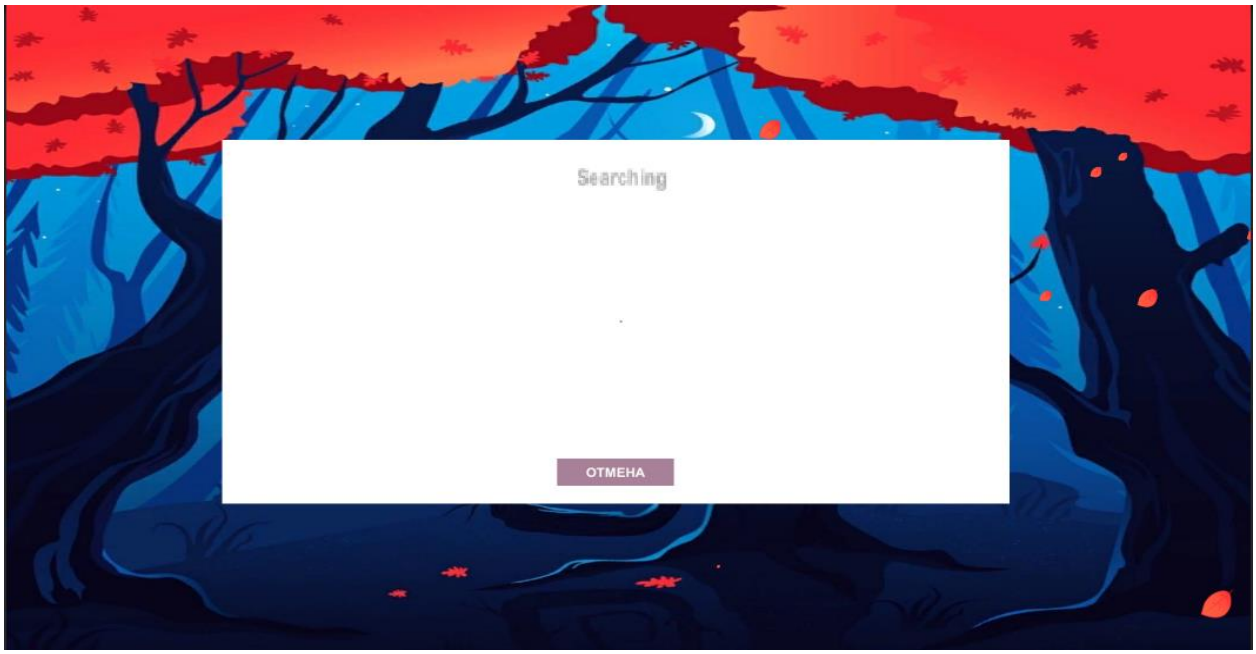

Рисунок 8 – Окно поиска игры

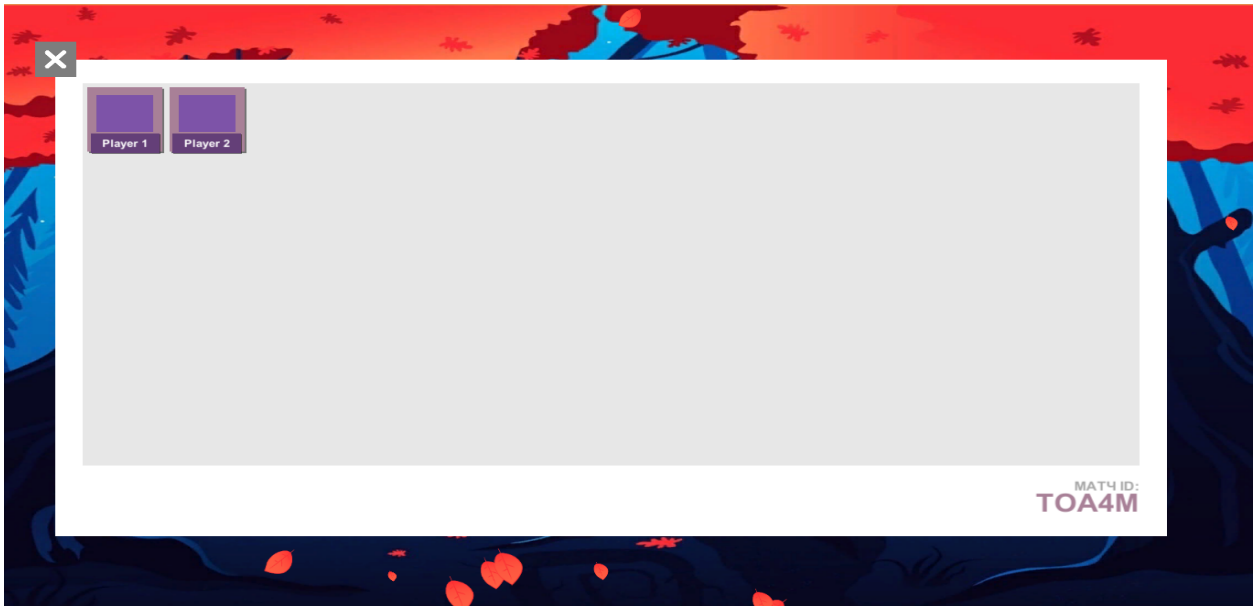

Рисунок 9 – Окно лобби (вид клиента)

Название прецедента: найти существующее лобби.

Предусловия:

1. Пользователь находится в окне “Главное меню”.

Основной сценарий:

1. Открывается окно “Лобби” с присутствующим хостом лобби (рисунок 9).

2. Пользователь вводит ID лобби в окно “Введите ID лобби”.

3. Пользователь нажал на кнопку “Присоединиться к лобби”.

Условие ввода в действие альтернативных сценариев

Альтернативный сценарий (не удалось найти лобби)

Появляется окошко с надписью “Лобби с таким ID не существует”

Название прецедента: начать игру.

Предусловия:

1. Хост лобби находится в окне “Лобби” (рисунок 10);

2. К лобби присоединился клиент.

Основной сценарий:

1. Хост лобби нажимает на кнопку “Поиск лобби”.

2. Открывается сцена с игровой сессией.

Условие ввода в действие альтернативных сценариев

Альтернативный сценарий (недостаточно игроков в лобби)

Кнопка “Поиск лобби” является неактивной.

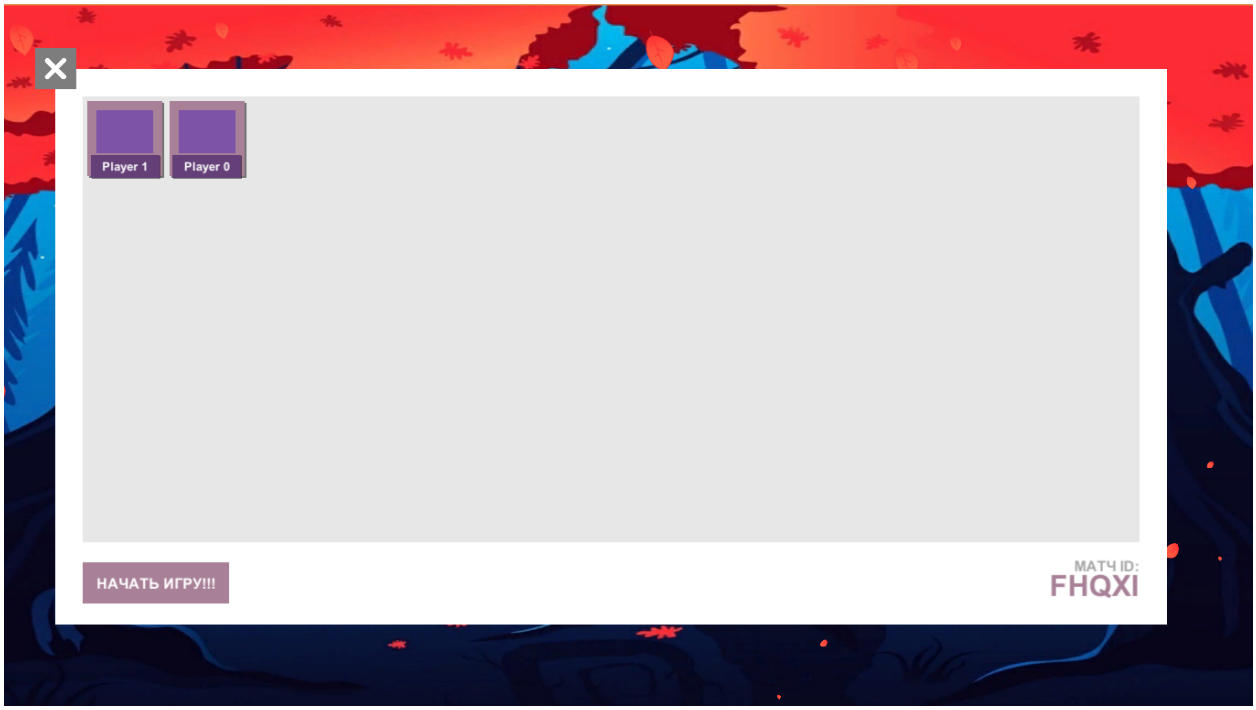

Рисунок 10 – Окно лобби (вид хоста лобби)

Название прецедента: покупка внутриигрового объекта.

Предусловия:

1. Игрок находится на игровом экране.

2. Стадия игры перешла в стадию боя (рисунок 11).

Основной сценарий: пользователь нажал на одну из кнопок игровых объектов, находящихся по краям экрана.

Постусловие:

1. Пользователь тратит валюту

2. Сервер создает юнитов.

Условие ввода в действие альтернативных сценариев

Альтернативный сценарий 1 (пользователь не может призвать юнит)

Область, где расположена информация о валюте выделяется красным цветом, сигнализируя о недостатке валюты.

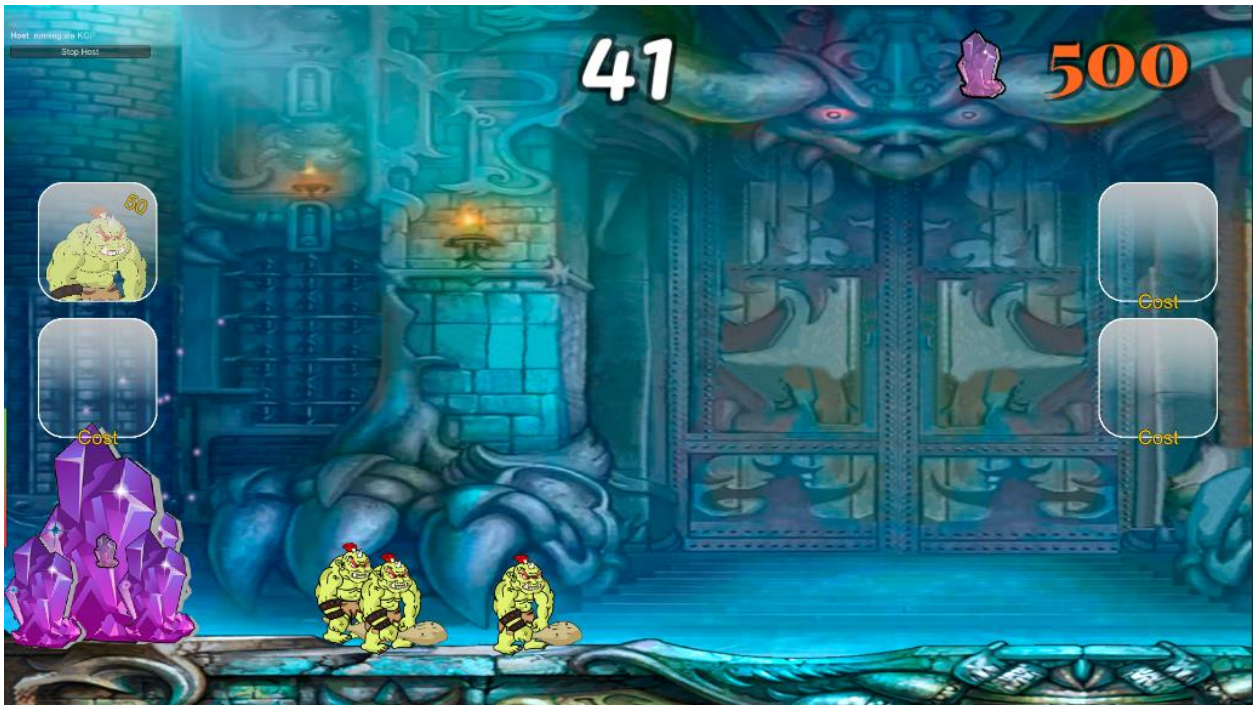

Рисунок 11 – Появление игровых юнитов после покупки

Название прецедента: улучшение внутриигровых объектов

Предусловия:

1. Игрок находится на игровом экране.

2. Стадия игры перешла в стадию планирования (рисунок 12).

Основной сценарий: появляются 3 кнопки, щелкнув на одну, из которых улучшаются выбранные параметры.

Постусловия:

1. Выбранная характеристика увеличилась незначительно.

2. Пользователю дается 400 валюты.

3. Выбранная характеристика увеличилась в 1.5 раз больше, чем в

первом случае.

Условие ввода в действие альтернативных сценариев

Альтернативный сценарий 1 (пользователь нажал на кнопку 1 или 3)

Область, где расположена информация о валюте выделяется красным цветом, сигнализируя о недостатке валюты.

Альтернативный сценарий 2 (пользователь пытается нажать несколько раз на кнопки)

Визуально ничего не происходит, на программном уровне стоит ограничение на одно нажатие.

Рисунок 12 – Меню улучшений

1.4.2 Спецификация требований к серверу

Сервер должен выполнять ряд требований приведенные в функциональных и нефунциональных требованиях.

1.4.2.1 Функциональные требования

Данные, отправляемые с клиента на сервер, и наоборот представляют собой байтовый массив. Далее с помощью плагина массив данных интерпретируется в игровые данные.

Для серверной части разработано API. API позволяет реализовать новых юнитов с базовым функционалом, а также определить состояние клиента в момент подключение к игровой сцене.

Метод OnServerAddPlayer

Метод отслеживает подключение к серверу клиентов, а также при подключении клиента устанавливает его строение в одной из двух позиций SpawnPosition, которые находятся непосредственно на сцене в момент ее инициализации.

Таблица 3 – Описание параметров метода OnServerAddPlayer

Параметр	Тип	Описание
Обязательные		
crystallBase	GameObject	Имитация охраняемого объекта. Подгружается в момент добавления игрока на сервер
conn	NetworkConnection	Уникальный идентификатор объекта, закрепленный за клиентом.

Метод UnitCheckDistanseToEnemy

Каждый момент времени юнит просчитывает расстояние между собой и вражескими юнитами на сервере. При сближении на расстояние меньшее чем дальность атаки юнита вызывает метод UnitDealDamageToUnit, который принимает ссылку на вражеского юнита.

Таблица 4 – Описание параметров метода UnitCheckDistanseToEnemy

Параметр	Тип	Описание
Обязательные		
UnitEmeny	GameObject	Ссылка на самого ближнего вражеского юнита
rangeUnit	float	Дальность атаки юнита

Метод UnitDealDamageToUnit

Юнит получает ссылку на вражеский объект, получая его параметр здоровья синхронизированного с сервером. Срабатывает анимация атаки. На определенном кадре анимации срабатывает событие нанесения урона, после чего здоровье вражеского юнита отнимается на значение равное урону юнита. Если юнит имеет дальнобойную атаку, событие нанесения урона создает снаряд.

Таблица 5 – Описание параметров класса UnitDealDamageToUnit

Параметр	Тип	Описание
Обязательные		
damage	float	Урон юнита
unitEmeny	GameObject	Ссылка на самого ближнего вражеского юнита

Продолжение таблицы 5

unitEnemy.GetComponent<>.health	float	Ссылка на здоровье вражеского юнита
attackSpeed	float	Скорость атаки юнита
Необязательные		
projectile	GameObject	Снаряд, создаваемый юнитами с дальнобойной атакой

Метод ProjectilesDamage

Снаряд принимает ссылку на объект вражеского юнита и параметр урона юнита. Метод запускает функцию MoveToward, которая обрабатывается на сервере, таким образом объект будет двигаться от начальной позиции, до позиции вражеского юнита. Снаряд имеет коллайдер. При столкновении коллайдеров наносится урон вражескому юниту.

Таблица 6 – Описание параметров метод ProjectilesDamage

Параметр	Тип	Описание
Обязательные		
speed	float	Скорость полета снаряда
ref unitEmeny	GameObject	Ссылка на самого ближнего вражеского юнита

Продолжение таблицы 6

ref enemyHealth	float	Ссылка на здоровье вражеского юнита

Метод CmdCreateUnitSmall

При нажатии на кнопку покупки юнитов, клиент отправляет запрос серверу на обработку транзакции. Сервер проверяет сумму доступной валюты и дает соглашение на создание юнита.

Таблица 7 – Описание параметров метода CmdCreateUnitSmall

Параметр	Тип	Описание
Обязательные		
unitSmall	GameObject	Ссылка на префаб юнита
cost	float	Стоимость юнита

Метод DataBaseRegister

При подключении пользователем к онлайн сцене или при разрушении строения противника производится подключение к базе данных по протоколу TCP, затем полученные данные интерпретируется в JSON формат (рисунок 13). Во втором случае срабатывает событие OnDestroyCrystall, которое проверяет разрушенное строение на права у клиента, если права были у клиента, то поле count_lose итерируется, иначе итерируется count_win.

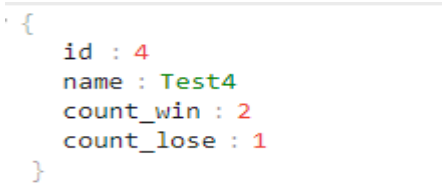

Рисунок 13 – Появление игровых юнитов после покупки

1.4.2.2 Нефункциональные требования

Приложение работает на платформе Android, начиная с минимальной 5.0 и уровнем API 21.

Сервер должен выполнять следующие требования:

- работать на операционной системе Linux Ubuntu 20.4 LTS;
- сервер должен отвечать за одну обязанность.

1.5 Вывод по главе

На основе анализа аналогов с открытым исходным кодом, сделан вывод, что на рынке нет приложений, удовлетворяющих требованиям задания. Рассмотренные игры имеют общий недостаток: отсутствует реализация клиент-серверной архитектуры.

Сформулирована спецификация требований для клиентского приложения в виде диаграмм прецедентов и их текстового описания. Подробно рассмотрены все функции, необходимые для обеспечения конкурентоспособности приложения.

Сформулирована спецификация требований для серверного приложения в виде текстового описания поддерживаемых методов. Методы разработаны в соответствии с требованиями клиенткой части.

2 Проектирование

На основе требования о том, что сервер должен отвечать за одну обязанность, принято решение о создании двух серверов. На рисунке 13 приведена архитектура разработанной системы. Система состоит из трех подсистем: клиентское, серверное двухранговое и трехранговое приложения.

Для получения данных о пользователе клиентское приложение общается с трехранговым серверным приложением по протоколу TCP. Серверное трехранговое приложение возвращает клиентскому приложению данные. Для их хранения на сервере используется база данных, в ней хранятся данные о имени пользователя, его победы и поражения. Во время игровой сессии клиентское приложение на постоянной основе общается с серверным двухранговым приложением по протоколу KCP. В свою очередь сервер хранит всю информацию об подключенных к нему клиентов, обрабатывает исходящую от них информацию и транслирует эту информацию остальным клиентам.

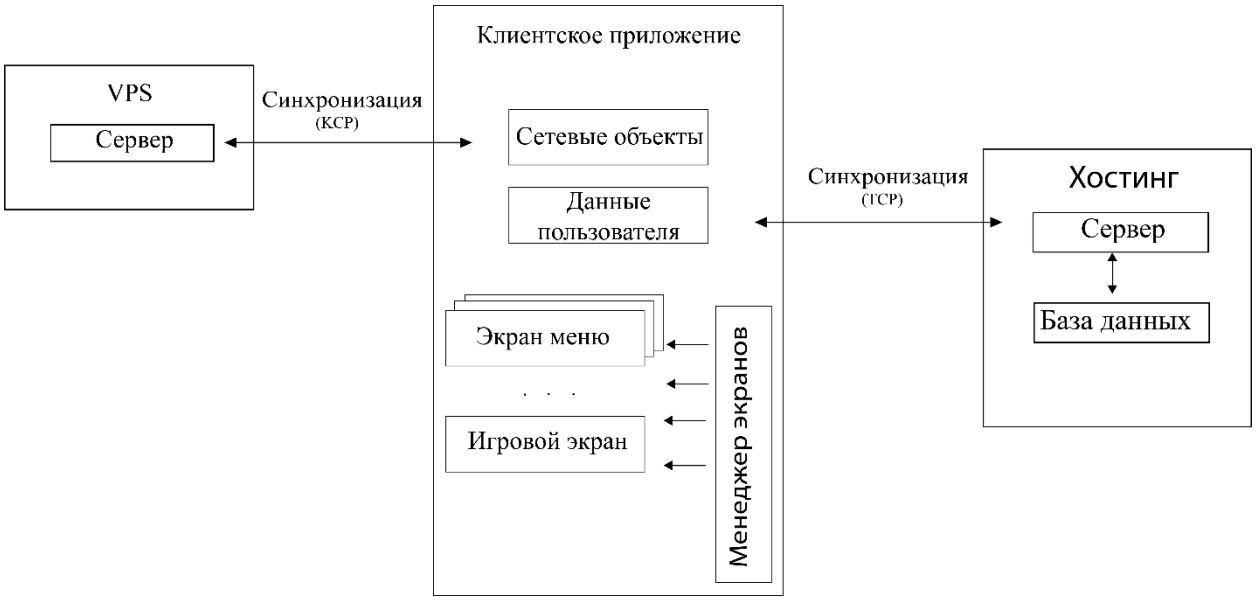

Рисунок 14 – Архитектура системы

2.1 Диаграммы последовательностей

Основная логика базируется на авторитарных объектах пользователя. В разделе приведены диаграммы последовательностей, для наиболее значимых прецедентов, связанных со взаимодействием различных компонентов системы.

На рисунке 15 изображена диаграмма последовательностей для вариантов использования “Найти случайное лобби” и “создать публичное лобби” представляющая процесс взаимодействия программы с сервером для достижения цели – подключение к лобби второго пользователя.

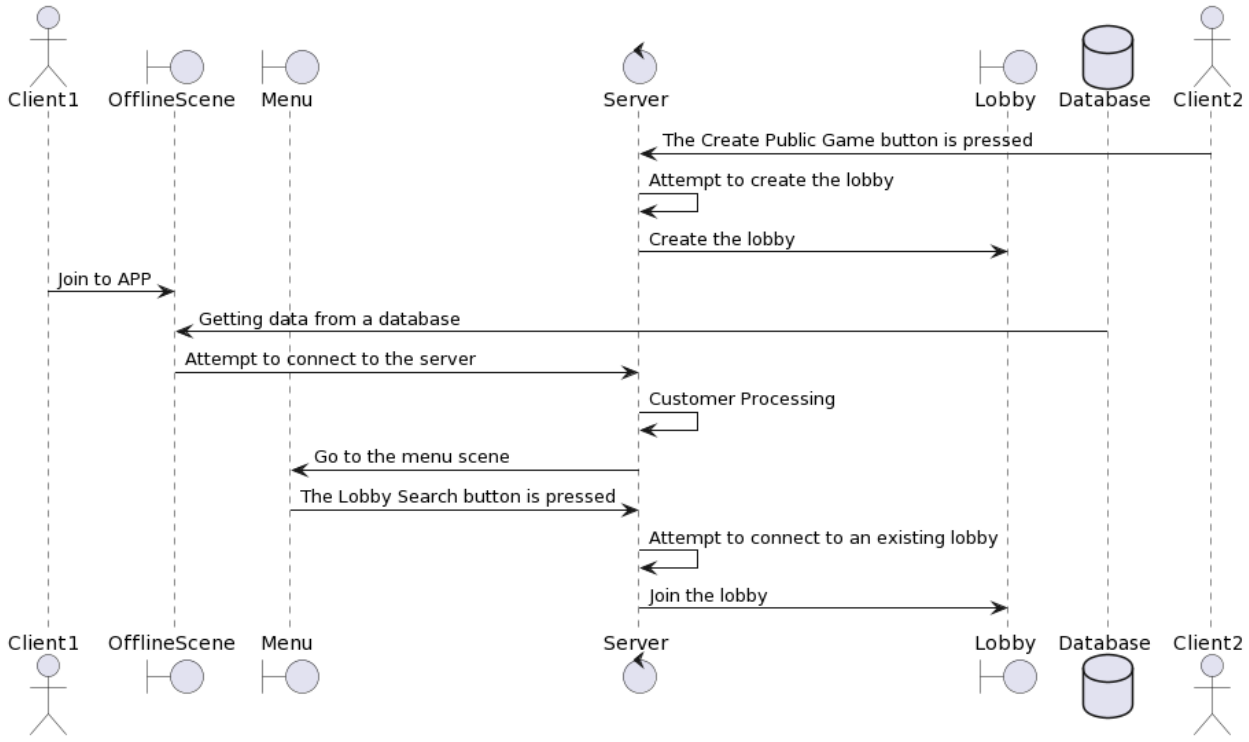

Рисунок 15 – Диаграмма последовательности для вариантов “Найти случайное лобби” и “создать публичное лобби”

На рисунке 16 изображена диаграмма последовательностей для варианта использования “Покупка внутриигрового объекта”, представляющая процесс

взаимодействия программы с сервером для достижения цели – покупка юнита за внутриигровую валюту. При нажатии на кнопку сервер проверяет о наличии доступной валюты у клиента. При положительном результате сервер сообщает клиентам о создании нового объекта, его координат и авторитарном праве. Затем создает его на игровом поле.

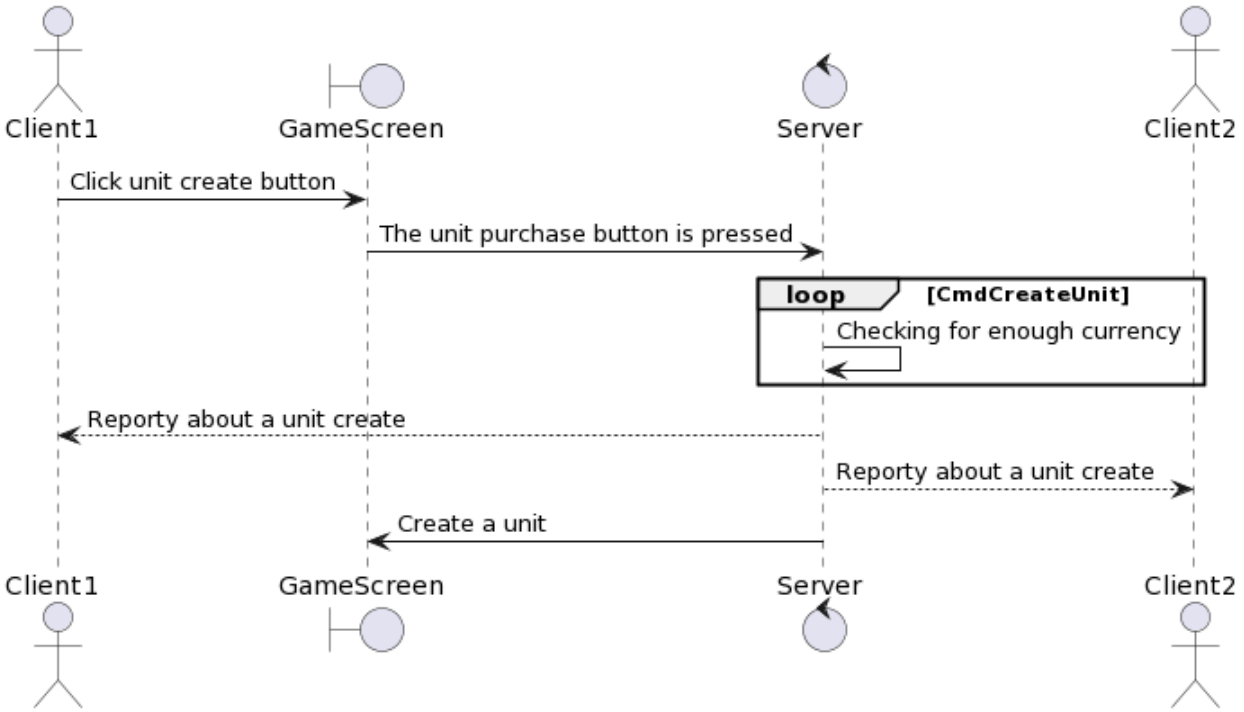

Рисунок 16 – Диаграмма последовательности для варианта “Покупка объекта”

На рисунке 17 изображена диаграмма последовательностей для варианта использования “Улучшения внутриигровых объектов”, представляющая процесс взаимодействия программы с сервером для достижения цели – покупка улучшения. При нажатии на кнопку сервер проверяет наличие валюты у пользователя. При положительном результате в зависимости от нажатой кнопки клиент получит то или иное улучшение обработанного на сервере. Второй клиент не будет знать результата о транзакции первого клиента, как и первый второго.

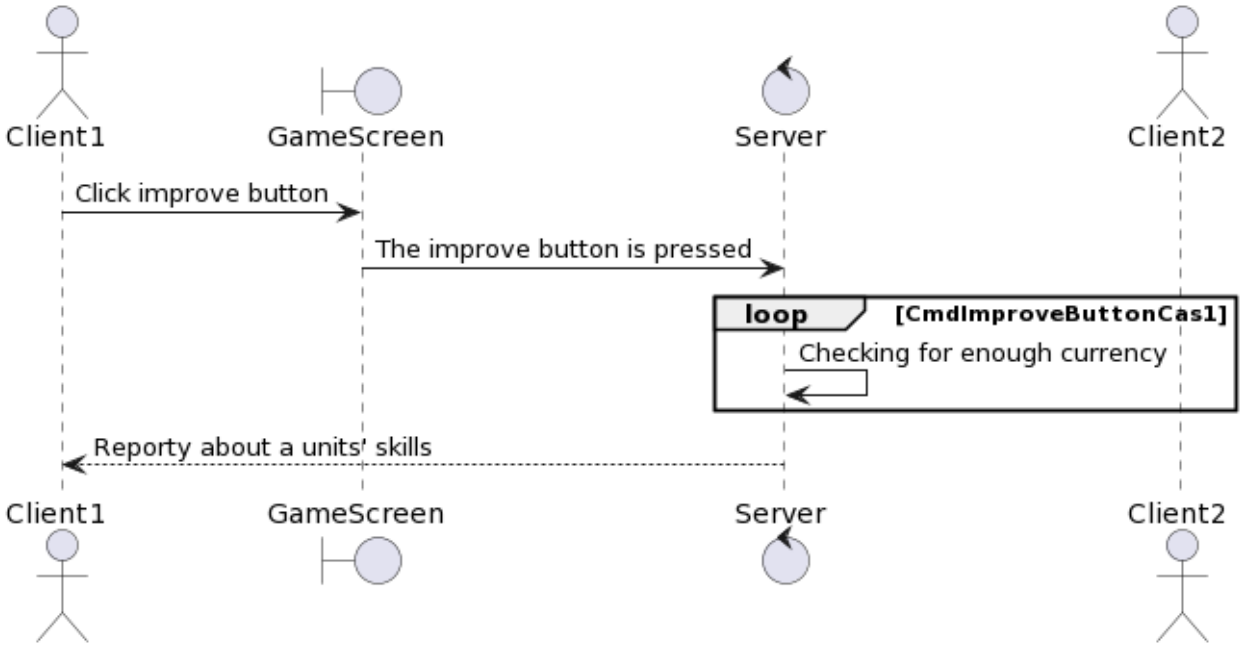

Рисунок 17 – Диаграмма последовательности для варианта “Улучшение внутриигровых объектов”

2.2 Серверная двухранговая архитектура

Для реализации серверной части используется плагин Mirror[10]. Он предлагает ряд классов, которые обрабатывают информацию о клиенте, поведении объектов, создание комнат.

2.2.1 Настройка сетевого менеджера

В первую очередь необходимо настроить объект NetworkManager (рисунок 18), который отвечает за данные о клиенте и их обработке. Плагин предлагает уже готовый объект, но он подойдет только для тривиальных систем. Поэтому создается новый класс, который наследуется от класса NetworkManager, что дает право обрабатывать данные о клиенте.

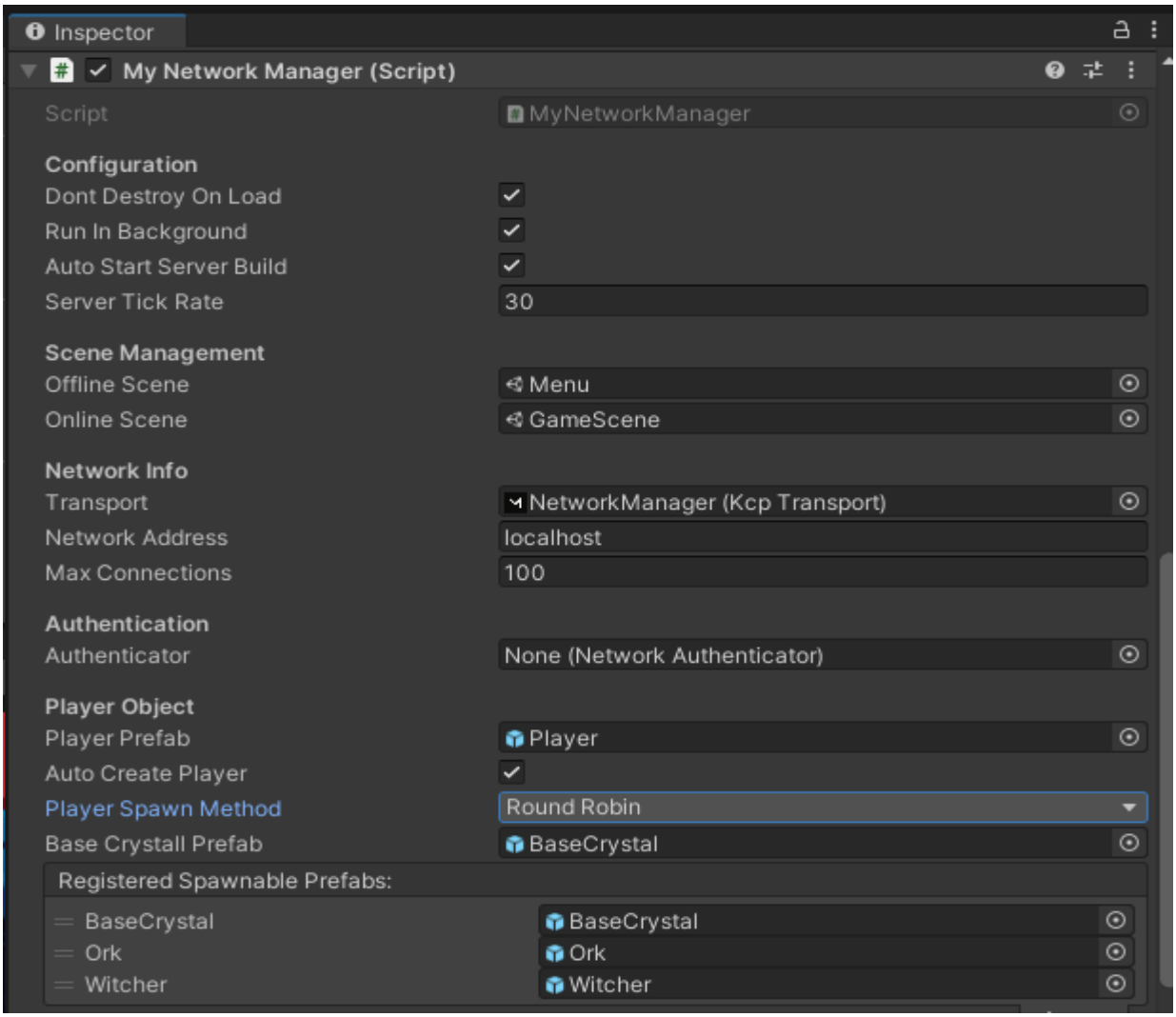

Рисунок 18 – Настройка сетевого менеджера

Теперь сетевой менеджер имеет ряд возможностей, которыми он не обладал до процесса настроек.

1. Сетевой менеджер не будет уничтожаться при загрузке онлайн сцены.

2. При сворачивании экрана сетевой менеджер будет продолжать работать дальше.

3. Сетевой менеджер автоматически включается при запуске приложения.

4. Сетевой менеджер работает с 3 сценами. 1 сцена «OfflineScene» позволяет запустить сервер. 2 сцена «Menu». К ней автоматически

подключаются все игроки при запущенном сервере. 3 сцена «GameScene». После закрытия лобби, клиентов переносит на эту сцену.

5. Клиент с сервером обращаются по протоколу KCP основанный на протоколе UDP.

6. Метод появление клиентов на сцене «GameScene» является кольцевым, что позволяет распределить клиентов по краям игровой сцены без коллизий.

7. Определены сетевые объекты, которые будут взаимодействовать с сервером во время игровой сессии.

2.2.2 Архитектура лобби

Архитектура лобби построена с использованием паттерна MVC. В классе MatchMaker обрабатываются все входные данные. Представлением является UI-элементы лобби. В классе UILobby обрабатываются запросы от клиента, а также настраивается представление. На рисунке 19 изображена структура поведения лобби исходя из своего состояния.

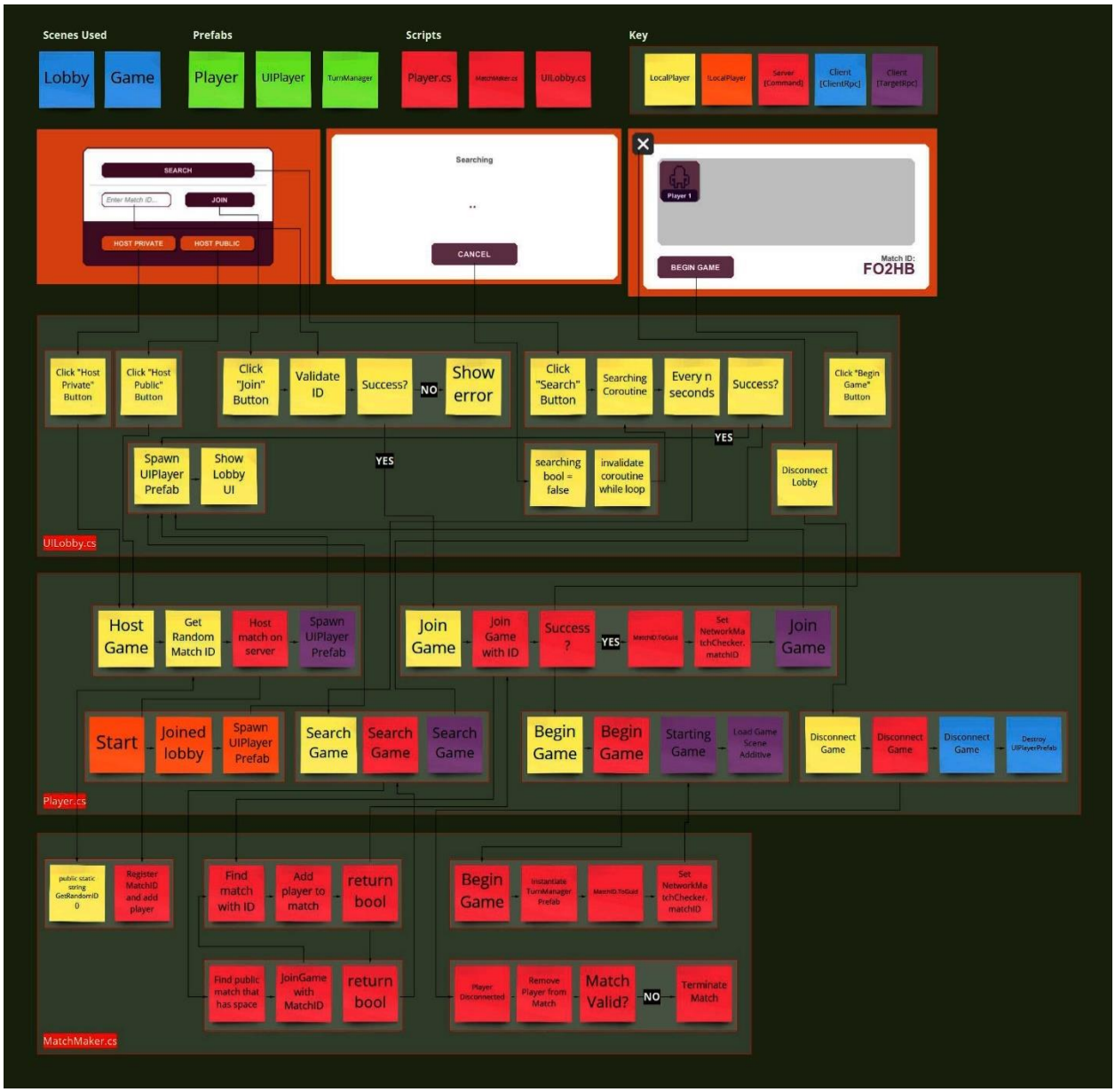

Рисунок 19 – Структура поведения лобби

2.3 База данных

Для хранения информации данных о пользователе на сервере используется база данных (рисунок 20).

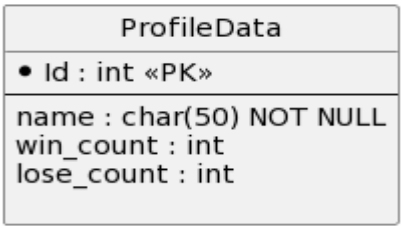

Рисунок 20 – ER-диаграмма базы данных

База данных состоит из одной таблицы (Profile). Таблица содержит в себе информацию о идентификаторе пользователя, числе побед и поражений.

2.4 Диаграмма классов

Юниты доступные в игре имеют много общих черт: здоровье, урон, скорость атаки, дальность атаки и другие. Поэтому целесообразно создать абстрактный класс, от которого будут наследоваться юниты. В абстрактном классе происходит валидация данных. Таким образом, если кто-то попытается из вне подменить значение здоровья или другой переменной, то настройки методов, обращающиеся с сервером, не позволят изменить данные (рисунок 21).

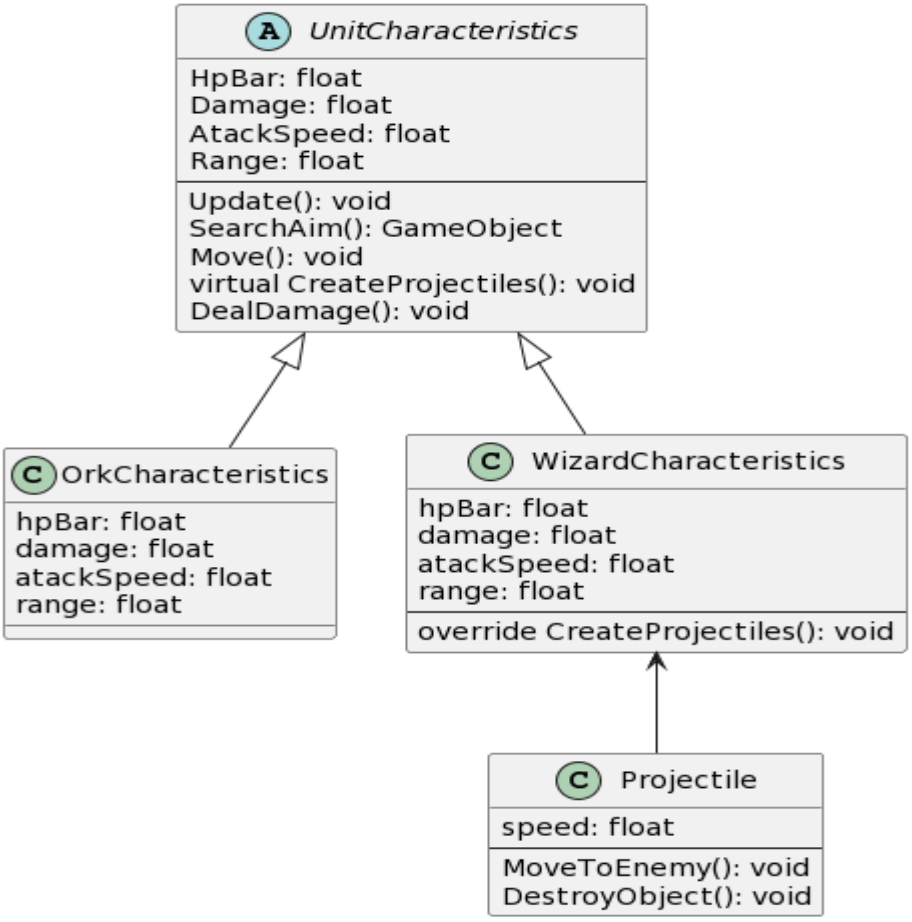

Рисунок 21 – Диаграмма классов архитектуры юнитов

Для того чтобы сервер смог создать юнитов, объект Player должен иметь класс UnitsControl. При нажатии на кнопку подписанный на нее класс ButtonCreate создаст юнита. Юнит содержит в себе класс OrkCharacteristic с двумя переопределенными методами OnStartServer и OnStopServer.

При срабатывании первого метода, ссылка на данный объект передается классу UnitsControl, через событие и добавляет объект в список. Если юнит погибает, то срабатывает метод OnStopServer, ссылка на данный объект передается классу UnitsControl, через событие и удаляет данный объект из списка (рисунок 22).

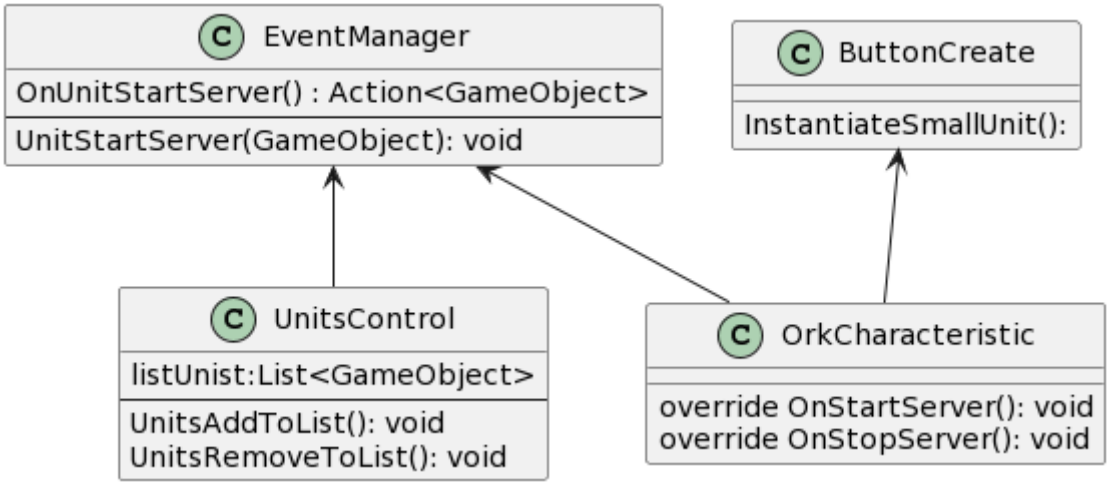

Рисунок 22 – Диаграмма классов появления юнита на игровой сцене

Класс ButtonCas хранит ссылки на все префабы пользователя. Класс содержит список методов, которые хранят улучшения для юнитов и стоимость покупки этих улучшений. Метод из списка выбирается случайным образом в момент вхождения игры в стадию планирования. При нажатии на кнопку, данные увеличиваются на n-ное количество единиц, в зависимости от метода (рисунок 23).

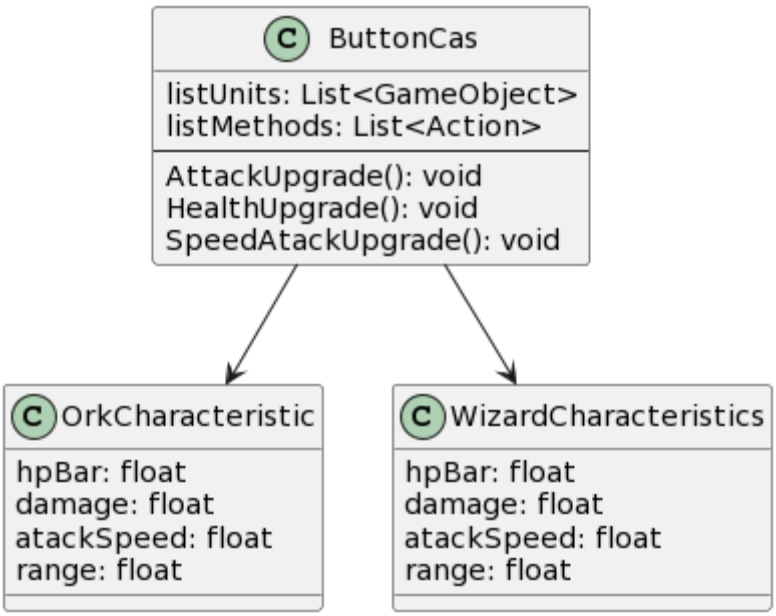

Рисунок 23 – Диаграмма классов покупки улучшений

2.5 Вывод по главе

В соответствии с техническим заданием:

- предложена архитектура системы и структура базы данных;
- с помощью нотации диаграмм последовательностей провизуализированы наиболее сложные отношения в системе, более детально проработано взаимодействие объектов;
- на основе диаграмм последовательностей разработаны диаграммы классов, описывающие статическую модель системы;
- разработана архитектура лобби и серверная часть системы.

3 Реализация и документация

3.1 Реализация

3.1.1 Выбор инструментов

Для реализации игры был выбран движок Unity3D. Движок обладает мощным набором инструментов, для работы как с объектами, так и с UI составляющим. Для движка существует ряд технологий для создания клиент-серверной архитектуры.

Таблица 8 – Продукты для создания серверной части

Характеристики	SignalR (библиотека) [12]	Mirror (плагин)	Photon PUN(плагин) [13]	UNet (плагин) [14]
Написание сервера	Самостоятельно	Базовые настройки	Сервер настроен	Базовые настройки
Сложность написания логики	Очень сложно	Сложно	Средне	Сложно
Русское комьюнити	Почти нет	Мало	Средне	Почти нет
Поддержка разработчиков	Да	Да	Да	Закроется в 2022 году

Продолжение таблицы 8

Цена	Бесплатно	Бесплатно	0.3$ за место на сервере	Бесплатно

Для реализации игры был выбран плагин Mirror. Во-первых, плагин является усовершенствованным плагином от разработчиков Unity – Unet, который закроется в 2022 году, и на место которого придет новое решение. Во-вторых, плагин является абсолютно бесплатным, а также есть хорошая документация. В-третьих, большое англоязычное комьюнити, которое исчисляется десятками тысяч человек.

Для рисовки спрайтов были использованы Adobe Photoshop 2022[15] и Adobe Illustrator 2019[16]. Первый был необходим для рисования растрового изображения в пиксельном формате. Таким способом был нарисован фон для игровой сессии. Второй необходим для рисования спрайтов объектов и UI.

Для анимирования спрайтов лучшим решением является Dragon Bones[17]. Он легок в освоении и является бесплатным.

3.1.2 Игровой процесс

Игрок появляется на одной из двух точек, которые расположены по краям карты. Задача пользователя - разрушить строение соперника.

Для этой задачи в игре существуют юниты, которых можно призвать за внутриигровую валюту. Внутриигровая валюта достается за уничтожение юнитов соперника или за выбор валюты на стадии планирования.

Каждый вид юнитов индивидуален, имеет разное количество здоровья, урона, скорости атаки, анимацию, звуки, видом боя, способности к созданию

снарядов, стоимости призыва и получения валюты соперником при уничтожении юнита.

В игровой сессии существует две стадии. Стадия боя длится полторы минуты. На этой стадии возможна покупка юнитов. По окончанию этой стадии все юниты, находящиеся на игровом поле, проигрывают анимацию смерти и уничтожаются, не принося никакой валюты обеим сторонам.

Затем идет стадия планирования, которая длится 30 секунд. На этой стадии появляется меню покупки, где на выбор предстаёт три варианта улучшения. Первый вариант стоит 400 валюты и совсем незначительно увеличивает выбранную характеристику. Второй вариант дает 400 валюты. Третий вариант стоит 800 валюты и увеличивает в 1.5 раз больше характеристику, чем первый вариант. Затем вновь возобновляется стадия боя. Игровая сессия заканчивается при разрушении любого строения.

3.1.3 Меню

Приложение содержит несколько окон, поэтому необходимо осуществлять переход между ними. Для этого был создан класс EventManagerUI, который хранит события и методы для вызова этих событий. Классы, работающие с окнами, подписываются на события и при необходимости вызывают методы с этими событиями, для осуществления переключений.

Класс NetworkManager уже реализует переключение сцен. При инициализации игровой сессии, NetworkManager переключается из OfflineScene в OnlineScene, куда подгружает все сетевые объекты и их настройки.

3.1.4 Реализация базы данных

Данные приложения (информация о пользователях) хранятся в базе данных MySQL. Чтобы не работать с SQL напрямую, в проекте используется Entity Framework Core - решение для работы с базами данных (ORM), которое используется в программировании на языках семейства .NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей *(entity)*, а не таблиц. Работа с Entity Framework Core осуществляется по принципу “Code First” - ORM строит базу данных на основе миграционных файлов, которые генерируются из переданных моделей

3.1.5 Интерфейс

Интерфейс приложения реализован с помощью многочисленных UI-компонентов – canvas. Canvas является разметочным UI-элементом. UI-элементы поддерживают иерархию. Самые первый в иерархии находится canvas-элемент, растянутый на весь экран. Рендер экрана у основного canvas выбран Screen Space – Overlay. Это свойство выводит все UI-элементы в иерархии поверх игровых объектов на сцене. Внутри основного canvas есть ряд мелких canvas с настроенным pivot, для адаптирования UI-элементов под разного разрешения экраны. Для текста используется экспериментальная разработка Unity – TextMeshPro[18]. Это замена текстовому интерфейсу Unity и устаревшей текстовой сетке.

3.1.6 Оптимизация системы

В Unity3D все объекты представляют собой контейнеры (GameObject) для различных компонентов (Component), которые могут быть встроены как в сам движок (Transform, Rendering и т.д.), так и пользоваться скриптами

(MonoBehaviour). Компонент может быть назначен как напрямую в инспекторе редактора, так и получить прямиком из контейнера, для этого используется функция GetComponent<T>().

GetComponent очень затратная функция, забирающая львиную долю ресурсов системы. При необдуманном использовании функции приложение начинает заметно грузить систему, а как следствие и само начинает виснуть. На рисунке 24 показано различие производительности между оптимизированным способом вызова 20 функции GetComponent() и не оптимизированного

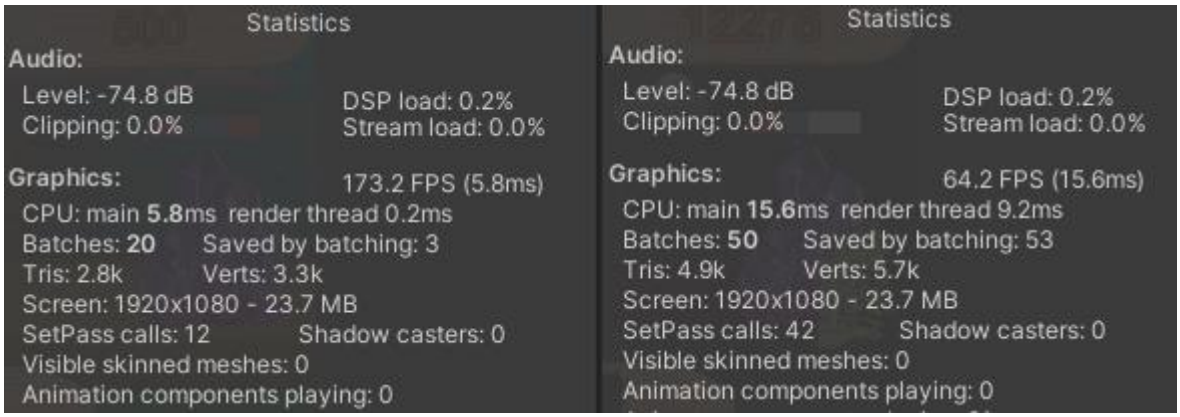

Рисунок 24 – Сравнение производительности двух способов реализации

Кэширование – способ сохранить компонент на стадии инициализации, не обращаясь к нему повторно. Существует множество способов реализации кэширование в Unity3D, но самым производительно верным является способ сохранения ссылки на компонент в функциях Start() или Awake() класса MonoBehaviour.

3.2 Тестирование

Первый этап тестирования приложения осуществлялось с помощью

загрузки на тематический сервер Discord apk фаил и на тематический Telegram канал, посвященный Unity разработке. В тестировании приняло 11 человек и был выявлен ряд ошибок и багов.

Второй этап тестирования приложения осуществлялось на платформе Yandex игры [19]. После прохождения модерации стала доступна возможность бета-тестирования игры, в которой может принять каждый желающий за очки, которые дарит Yandex. Так называемые «Яны» можно тратить на разные внутриигровые предметы предоставленные на Yandex играх.

3.3 Инструкция разработчика

Настройка окружения программиста для разработки приложений с использованием Unity под платформу Android осуществляется нетривиально.

Однако, доставку окружения можно обеспечить средствами виртуализации. В рамках работы предпринималась попытка использования трех готовых докер контейнеров [20, 21, 22]. Ни один из них корректно развернуть не удалось. Поэтому, была настроена виртуальна машина на базе VMware, содержащая необходимый комплект разработки на базе операционной системы Ubuntu 20.4 LTS.

Полученный образ [23, 24] позволяет выполнить сборку проектов, а также их отладку с использованием эмулятора и реального устройства. Для получения доступа к ссылкам необходимо иметь ключи дешифровки: 8yjje8mC9JHwgvvBWhFP8SVJ3xB8MPGwYhQBB860v5W16OJSzo.

3.4 Вывод по главе

1. Реализован прототип игры «Tower-Defense».

2. Составлена инструкция по настройке окружения разработчика под платформу Android с использованием движка Unity.

3. Создан образ для виртуальной машины ОС Ubuntu 20.4 LTS.

4. Проведено тестирование игры с помощью возможностей Yandex игры и людьми из других социальных сетей.

Заключение

В результате проделанной работы:

1. Спроектирован, реализован и протестирован прототип игры “Tower defense”.

2. Создан образ с установленным и настроенным комплектом разработчика под платформу Android с использованием движка Unity.

В разработанном приложении присутствуют недостатки, исправить их можно путем пополнения функциональных возможностей:

- добавить новые карты;
- добавить новых юнитов;
- добавить индивидуальные режимы у юнитов;
- улучшить взаимодействие с базой данных, сделать авторизацию через Yandex игры;
- другого рода улучшения.

Исходный код приложения доступен для скачивания с git-репозитория [25].

Список использованных источников

1.IGN. Годовая выручка видеоигр [Электронный ресурс]. URL: https://ru.ign.com/nintendo-switch/91334/news/v-etom-godu-mirovoi-videoigrovoi-rynok-poluchit-152-milliarda-dollarov-vyruchki (дата обращения: 22.12.2021)

2.ResearchGate. Dynamic Difficulty Adjustment in Tower Defense [Электронный ресурс]. URL: https://www.researchgate.net/publication/283161874_Dynamic_Difficulty_Adjustment_in_Tower_Defence (дата обращения 03.03.2022)

3.Springer Link. Procedural Content Generation of Custom Tower Defense Game Using Genetic Algorithms [Электронный ресурс]. URL: https://link.springer.com/chapter/10.1007/978-3-030-75275-0_54 (дата обращения 03.03.2022)

4. Springer Link. Simple Gamer Interaction Analysis through Tower Defence Games [Электронный ресурс] URL: https://link.springer.com/chapter/10.1007/978-3-319-10774-5_18 (дата обращения 03.03.2022)

5.Google play. Clash Royale [Электронный ресурс]. URL: https://play.google.com/store/apps/details?id=com.supercell.clashroyale&hl=ru&gl=US (дата обращения: 22.12.2021)

6.Google play. Kingdom Rush Origins – TD [Электронный ресурс]. URL: https://play.google.com/store/apps/details?id=com.ironhidegames.android.kingdomrushorigins&hl=ru&gl=US (дата обращения 9.03.2022)

7.Google play. Grow Castle – Tower Defense [Электронный ресурс]. URL: https://play.google.com/store/apps/details?id=com.raongames.growcastle&hl=ru&gl=US (дата обращения 9.03.2022)

8. Блог программиста. UML [Электронный ресурс]. URL: https://pro-prof.com/archives/2594 (дата обращения: 22.12.2021)

9. Блог программиста. Нотации модели сущность - связь [Электронный ресурс]. URL: https://pro-prof.com/archives/8126 (дата обращения: 22.12.2021)

10.Mirror. Mirror Networking [Электронный ресурс]. URL: — Режим доступа: https://mirror-networking.com/ (дата обращения 25.12.2021)

11.PlantUml. plantumlwebserver [Электронный ресурс]. URL: — Режим доступа: http://www.plantuml.com (дата обращения 25.5.2022)

12. Microsoft. SignalR [Электронный ресурс]. URL: — Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/signalr/overview/getting-started/introduction-to-signalr (дата обращения 24.12.2021)

13.Photon. PhotonPun [Электронный ресурс]. URL: — Режим доступа: https://www.photonengine.com/pun (дата обращения 5.11.2021)

14.Unity. Unet [Электронный ресурс]. URL: — Режим доступа: https://docs.unity3d.com/Manual/UNet.html (дата обращения 5.01.2022)

15.Adobe. Photoshop [Электронный ресурс]. URL: — Режим доступа: https://www.adobe.com/ru/products/photoshop.html (дата обращения 10.10.2021)

16.Adobe. Illustrator [Электронный ресурс]. URL: — Режим доступа: https://www.adobe.com/ru/products/illustrator.html (дата обращения 15.10.2021)

17.Dragon Bones [Электронный ресурс]. URL: — Режим доступа: https://docs.egret.com/dragonbones/en (дата обращения 5.02.2022)

18.Unity. TextMeshPro [Электронный ресурс]. URL: — Режим доступа: https://docs.unity3d.com/Manual/com.unity.textmeshpro.html (дата обращения 9.03.2022)

19.YandexGame. Я Игры [Электронный ресурс]. URL: — Режим доступа: https://yandex.ru/games/ (дата обращения 10.05.2022)

20.Docker. DockerHub [Электронный ресурс]. URL: — Режим доступа: https://hub.docker.com/r/unityci/editor (дата обращения 20.05.2022)

21.Docker. DockerHub [Электронный ресурс]. URL: — Режим доступа: https://hub.docker.com/r/cimg/unity (дата обращения 20.05.2022)

22.Docker. DockerHub [Электронный ресурс]. URL: — Режим доступа: https://hub.docker.com/r/unitytechnologies/accelerator (дата обращения 20.05.2022)

23.Mega. meganz [Электронный ресурс]. URL: — Режим доступа: https://mega.nz/folder/mMFA0KIC (дата обращения 20.05.2022)

24.Mega. meganz [Электронный ресурс]. URL: — Режим доступа: https://mega.nz/#F!u2R3CCoL (дата обращения 20.05.2022)

25.GitHub. BrawlTD [Электронный ресурс]. URL: — Режим доступа: https://github.com/SergeySpiridon/BrawlTD1.2 (дата обращения 20.05.2022)

Федеральное государственное автономное
образовательное учреждение
Высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра вычислительной техники
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
«20» 06 2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника
код – наименование направления

Игра в жанре «Tower defense»
тема

Руководитель Васильев, 24.06.22 В. С. Васильев
подпись, дата

Выпускник 21.06.22 С. А. Спиридонов
подпись, дата

Нормоконтролер Васильев, 24.06.22 В. С. Васильев
подпись, дата

Красноярск 2022