

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ В ЖАНРЕ
КВЕСТ НА UNITY

Студент


(подпись, дата)

Михальцова Полина Валерьевна
(ФИО полностью)

Руководитель


(подпись, дата)

Сорокин А. А.
(фамилия, инициалы)

Нормоконтролёр


(подпись, дата)

Бычков А. С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025 г. № 16)

Заведующий кафедрой

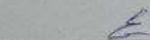

(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

16 июня 2025 № 36)

Секретарь ГЭК


(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«06» 05 2025 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Михальцова Полина Валерьевна гр. 21ВВП2 факультета вычислительной техники
2. Тема работы: Мобильное приложение в жанре квест на Unity

Тема утверждена приказом ПГУ № 601/о от "06" 05 2025 г.

3. Руководитель бакалаврской работы: доцент кафедры ВТ ПГУ Сорокин А. А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Назначение разработки: разработка стабильно работающего, оптимизированного под мобильные устройства игрового продукта с интуитивно понятным управлением и визуально привлекательным интерфейсом с помощью языка программирования C# и платформы Unity.
Исходные данные: техническое задание, включающее описание механик и пользовательского интерфейса

5. Перечень подлежащих разработке вопросов

1. Анализ предметной области
2. Анализ аналогичных квест-игр
3. Разработка структуры сцен
4. Создание логики уникальных головоломок и системы взаимодействия с объектами

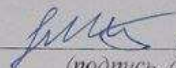
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Выбор темы и разработка	20%	06.05.2025 –	

	индивидуального плана проведения работ		07.05.2025	
2	Выбор технологий разработки	15%	08.05.2025 – 14.05.2025	
3	Анализ аналогичных приложений	15%	15.05.2025 – 18.05.2025	
4	Разработка структуры, создание логики и системы взаимодействия	35%	19.05.2025 – 26.05.2025	
5	Оформление выпускной квалификационной работы	15%	27.05.2025 – 30.05.2025	

Дата выдачи задания " 06 " 05 2025 г.

Руководитель бакалаврской работы  06.05.2025 Сорокин А. А.
(подпись, дата)

Задание к исполнению принял студент  06.05.25 Михальцова П. В.
(подпись, дата)

Работу к защите допустить
Декан факультета  16.06.2025 Фионова Л. Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 82 страницы, 20 рисунков, 4 таблицы, 13 источников, 3 приложения.

КВЕСТ, СКРИПТЫ, UNITY, БАЗА ДАННЫХ, ГОЛОВОЛОМКИ, ANDROID

Цель работы – разработать игровое приложение для Android, предоставляющее квестовый опыт с элементами головоломок, для развития мышления и навыков.

В результате проделанной работы было разработано мобильное игровое приложение, которое предоставляет качественный квестовый опыт с упором на решение головоломок и удобное управление.

В ходе разработки мобильного приложения использовались следующие технологии: C#, Unity, Microsoft Visual Studio 2022, BandLab, Adobe Photoshop.

					ПГУ 1.090301.37.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Михальцова П.В.		10.06.25	«Мобильное приложение в жанре квест на Unity» Пояснительная записка	Лит.	Лист	Листов
Провер.		Сорокин А.А.		10.06.25			2	82
Н. Контр.		Бычков А.С.		10.06.25		ФВТ, каф. ВТ		
Утверд.		Митрохин М.А.		10.06.25		гр. 21ВВП2		

Содержание

Перечень сокращений и обозначений.....	4
Введение.....	5
1 Анализ предметной области	6
1.1 Описание предметной области	6
1.2 Анализ актуальности темы.....	7
1.3 Анализ аналогов игровых приложений	7
1.3.1 Серия игр «Cube Escape Collection».....	8
1.3.2 Хоррор – квест игра «The Secret Elevator».....	9
1.3.1 Хоррор – квест игра «Fran Bow».....	11
1.3.4 Вывод.....	11
1.4 Постановка задачи.....	12
2 Выбор технологий разработки приложения.....	14
2.1 Игровые платформы	14
2.2 Графический редактор.....	18
2.3 Музыкальный редактор	20
2.4 Язык программирования.....	22
3 Архитектура мобильной квест - игры	24
4 Реализация приложения	27
Заключение	42
Список использованных источников	43
Приложение А – UML-диаграммы.....	44
Приложение Б – Листинг программы	47
Приложение В – Презентация.....	80

Перечень сокращений и обозначений

В настоящей работе применяются следующие сокращения и обозначения

ЯП – язык программирования

Android – операционная система для мобильных устройств

C# – язык программирования

PNG— растровый формат хранения графической информации

UI – User Interface (пользовательский интерфейс)

UML (Unified Modeling Language) – унифицированный язык моделирования

Unity – игровая платформа Unity

2D/3D – двухмерная/трехмерная графика

					ПГУ 1.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		4

Введение

В последние годы мобильные игры стали неотъемлемой частью цифровой культуры, привлекая внимание миллионов пользователей по всему миру. Существует множество видов и категорий приложений, как развлекательных, так и образовательных. Одним из наиболее популярных жанров в этой области являются квесты, которые предлагают игрокам опыт взаимодействия в виртуальном мире. Квесты не только развлекают, но и развивают креативность и навыки решения проблем. Также они помогают выработать в человеке внимательность, память и логическое мышление.

Именно приложение на мобильном устройстве позволит пользователям наслаждаться квест-игрой и совершенствовать свои умения в любое время и в любом месте. В связи с этим, разработка мобильных квест-игр становится актуальной задачей для многих разработчиков, стремящихся создать инновационные продукты, способные удовлетворить потребности современных игроков.

Для реализации таких игр требуется использование мощных инструментов и технологий. В дипломной работе будет рассмотрен процесс реализации мобильной игры в жанре квест с использованием платформы Unity. Результаты данной работы могут быть полезны как для начинающих разработчиков, так и для более опытных специалистов, стремящихся расширить свои знания и навыки в области мобильной игровой разработки.

1 Анализ предметной области

Разработка программного продукта всегда начинается с анализа. Требуется определить каким пользователям будет интересен данный продукт, выбрать современные технологии для разработки, которые предоставят в будущем поддерживать и совершенствовать приложение. В данном разделе рассматриваются общие теоретические вопросы реализации квестовой игры для мобильных устройств. Изучение этих вопросов поможет выявить основные проблемы и задачи, которые необходимо решить в процессе разработки программного продукта.

Цель приложения — предоставить пользователям увлекательный и интересный опыт, который сочетает в себе элементы игры и обучения.

1.1 Описание предметной области

Предметная область охватывает создание мобильного игрового приложения, разработанного для платформы Android с использованием Unity. Квест-приложение представляет собой интерактивную игру, в которой пользователи выполняют различные задачи, решают загадки и взаимодействуют с предметами. Основная цель игры — развлечение и интеллектуальное развитие игрока через решение логических задач.

Квесты в играх часто включают в себя элементы исследования, поиска предметов, решение логических задач и принятие решений. Такие приложения просты тем, что в них используются основные механики, что позволяет пользователю более быстро освоить управление.

Головоломки в видеоиграх могут принимать различные формы и механики. Некоторые из них требуют от игрока правильного размещения объектов или решения шифров, в то время как другие связаны с управлением определёнными элементами игрового мира. В современных играх данного жанра часто используется цепочка квестов — это серия заданий, которые

выполняются в определённой последовательности. Завершение предыдущего задания в цепочке, является обязательным условием для того, чтобы получить следующее.

В данном продукте пользователю нужно перемещаться по разным локациям, рационально сопоставлять подсказки и найденные вещи с задачами, где требуется разгадка.

1.2 Анализ актуальности темы

В современном мире существует множество игр, чтобы люди разных возрастов могли пользоваться ими. Квестовая игра будет направлена на широкий круг пользователей. Тема разработки мобильных приложений в условиях растущей популярности мобильных игр и увеличения числа пользователей смартфонов очень актуальна. Квесты способствуют развитию логического мышления, креативности и навыков решения проблем.

Младшее и среднее поколения используют игры в развлекательных или образовательных целях. Квесты помогают развивать логическое мышление и креативность. Также игры могут проходить дети вместе со своими родителями, это поможет укреплению семейных связей и развитию навыков командной работы.

Более взрослое поколение чаще всего прибегает к играм, чтобы расслабиться и отвлечься от проблем. Пользователи, которые любят разгадывать загадки и решать задачи, найдут в приложении интересный контент, который будет стимулировать их умственные способности.

1.3 Анализ аналогов игровых приложений

Каждый год на рынке появляются новые игровые приложения, предлагающие новые механики, захватывающие сюжеты и новые загадки для пользователя. Квест-игра — один из основных жанров компьютерных игр,

представляющий собой интерактивную историю с главным героем, управляемым игроком.

Для создания такого приложения важно провести анализ аналогичных решений. Это поможет выявить сильные и слабые стороны уже существующих игр, а также определить требования к разрабатываемому продукту. Исследование будет проводиться с точки зрения игровых механик, визуального стиля, удобства управления и других ключевых аспектов, влияющих на пользовательский опыт. Сюжетная составляющая не рассматривалась в качестве преимущества или недостатка, так как основное внимание будет уделено техническим характеристикам. Будут изучены три мобильных приложения:

- серия игр «Cube Escape Collection»;
- хоррор – квест игра «The Secret Elevator»;
- хоррор – квест игра «Fran Bow».

1.3.1 Серия игр «Cube Escape Collection»

Cube Escape Collection является сборником игр-головоломок. Создание игры происходило в игровом движке Unity на языках программирования C# или JavaScript. Для реализации графики и анимации использовались такие программы, как Adobe Photoshop и Illustrator.

Интерфейс игры отличается простотой и понятностью. Это дает возможность сосредоточиться игрокам на исследовании мира и решении головоломок. Все интерактивные элементы четко выделены, поэтому проще взаимодействовать с элементами. Интерфейс инвентаря позволяет легко управлять собранными объектами, это можно увидеть на рисунке 1.

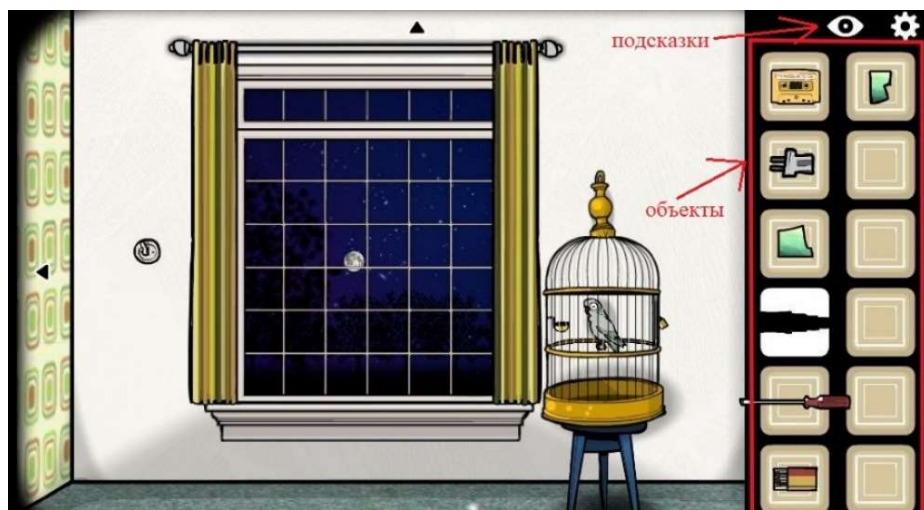


Рисунок 1 – Интерфейс инвентаря и подсказок\

Данное приложение имеет положительные параметры, которые делают ее до сих пор популярной. Такие как:

- минималистичный визуальный стиль. Использована 2D – графика, это снижает требования к устройству;
- оптимизация под мобильные устройства. Игра работает даже на слабых устройствах.

Несмотря на свою популярность и количество особенностей, Cube Escape Collection имеет свои недостатки. Такие как:

- однообразная игровая механика у головоломок. Присутствуют повторяющиеся загадки, что делает игровой процесс предсказуемым и однотипным;
- отсутствие интерактивных элементов. Имеет ограниченные взаимодействия с игровым окружением.

1.3.2 Хоррор – квест игра «The Secret Elevator»

The Secret Elevator – это 3D игра, которая сочетает в себе элементы головоломки и приключения. Данная игра была разработана на игровом движке Unity с использованием ЯП C# и программ для создания 3D графики.

Интерфейс в этом приложении с простыми элементами управления, которые позволяют легко взаимодействовать с объектами. Элементы

включают в себя инвентарь, кнопки взаимодействия и подсказки. Это показано на рисунке 2.



Рисунок 2 – Интерфейс игры «The Secret Elevator»

Положительные аспекты игры:

- формирование уникальных игровых элементов при каждом запуске. Некоторые объекты в игровом мире формируются случайным образом, это обеспечивает вариативность прохождения;
- логическая связность уровней. Задачи построены так, что решение одной загадки ведет к следующей.

Отрицательные аспекты игры:

- сложность головоломок. Большинство подсказок и ключевых элементов спрятаны в окружении;
- высокие требования к устройству. Из-за детализированной графики, игра некорректно работает на слабых устройствах;
- неоднозначность выражений. Формулировка задач и подсказок неясны и многозначны. Это затрудняет понимание, что требуется от пользователя.

1.3.3 Хоррор – квест игра «Fran Bow»

Fran Bow — это уникальная квест игра с элементами хоррора и психологической драмы. Игра была разработана на игровом движке Unity на языке C# с использованием программы Photoshop для создания графики.

Интерфейс игры показан на рисунке 3.



Рисунок 3 – Интерфейс игры «Fran Bow»

Преимущества приложения:

- богатый интерактив. Множество предметов и взаимодействий с окружением;
- четкая структура. Игра разделена на отдельные сцены, это помогает отслеживать прогресс.

Недостатки приложения:

- проблемы с оптимизацией. Возникают проблемы с производительностью или графикой на некоторых устройствах;
- неудобное управление. Мелкие объекты затрудняют взаимодействие при управлении с экрана.

1.3.4 Вывод

Разрабатываемая игра должна сочетать в себе лучшие черты рассмотренных аналогов, избегая недостатков, чтобы предложить

пользователям качественный игровой опыт. В таблице 1 представлено сравнение существующих аналогов.

Таблица 1 – Сравнение аналоговых игровых приложений

Критерий	Cube Escape	The Secret Elevator	Fran Bow
Графика	2D	3D	2D
Головоломки	Простые	Средние	Сложные
Оптимизация	Высокая	Средняя	Низкая
Управление	Удобное	Удобное	Неудобное
Интерактивность	Ограниченная	Средняя	Высокая

Исходя из полученных результатов, было принято решение создать приложение в жанре квест «Lunar Echoes», которое будет предоставлять удобный интерфейс, наличие разнообразных головоломок, оптимизированную 2D графику, сбалансированную сложность.

1.4 Постановка задачи

Проект представляет собой разработку обучающего и развлекательного приложения в виде 2D мобильной игры в жанре квест, которая будет соответствовать современным требованиям игроков. Основная задача заключается в создании увлекательной и стабильной игры с элементами поиска вещей и головоломками.

Функциональные требования к приложению:

- четкое разделение на уровни;
- мноуровневая структура сложности;
- механика решения головоломок различного типа;
- динамическое изменение уровня в зависимости от действия игрока;
- реализация интерактивных объектов, с которыми игрок может взаимодействовать, и обратную связь с ними.

Нефункциональные требования:

- минималистичный, но выразительный визуальный стиль;
- атмосферный звуковой дизайн.

Приложение должно предоставить пользователю качественный квестовый опыт с упором на решение головоломок и удобное управление, оставаясь стабильным на большинстве мобильных устройств.

					ПГУ 1.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		13

2 Выбор технологий разработки приложения

Разработка приложения требует тщательного подхода к выбору инструментов. Важно найти сочетание между функциональностью выбранных технологий и их эффективностью при работе на Android с различными техническими характеристиками. В этом разделе будут рассмотрены графические и звуковые редакторы, игровые платформы и ЯП.

2.1 Игровые платформы

Игровой движок — это программное обеспечение, предназначенное для разработки компьютерных игр и прочих интерактивных программ, обрабатывающих графику в реальном времени.

От выбора зависит эффективность создания, качество продукта и его производительность на разных устройствах. В настоящий момент представлено несколько популярных платформ, каждое из них обладает уникальными характеристиками, преимуществами и ограничениями. Проведя анализ ключевых особенностей, инструментов и возможностей будет отобран наиболее подходящий инструмент для разработки. Это позволит выявить оптимальный вариант для достижения поставленных задач и высоких результатов.

Наиболее известные три платформы:

- Unity;
- Unreal Engine;
- Godot Engine.

Unity – это мощная игровая платформа, которая поддерживает кросс-платформенную разработку и позволяет создавать 2D и 3D игры [1]. Основным языком программирования является C# [2]. Движок показан на рисунке 4.

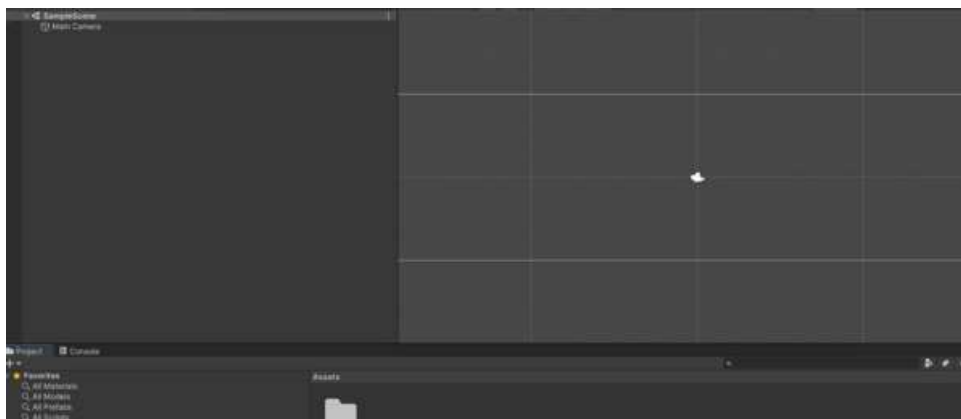


Рисунок 4 – Интерфейс Unity

Преимущества:

- оптимизация под Android. Отличная производительность даже на слабых устройствах [3];
- обширная документация. Большая база знаний.
- множество инструментов и библиотек [4];
- поддержка различных разрешений экранов;
- доступна без необходимости приобретения лицензии.

Недостатки:

- сложность интерфейса [5];
- возможные проблемы с производительностью.

Unreal Engine – инструмент, предназначенный для создания игр 2D и 3D с реалистичной графикой и сложной физикой, представлен на рисунке 5. Поддерживает программирование на C++ и визуальное скриптование с помощью Blueprints.

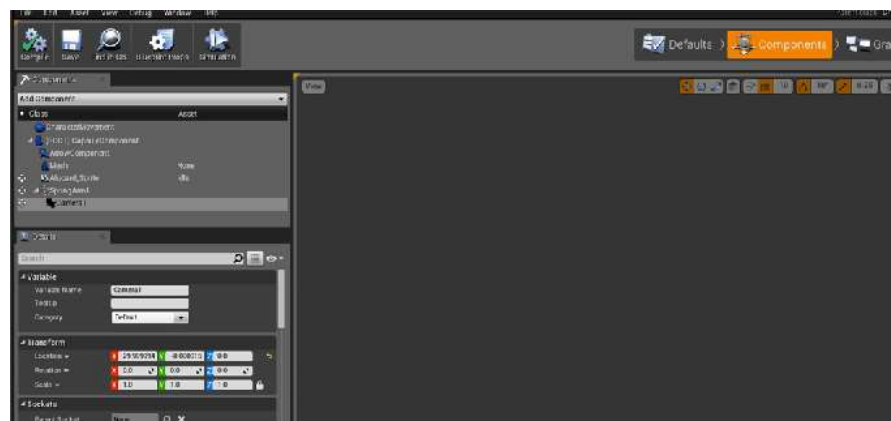


Рисунок 5 – Интерфейс Unreal Engine

Преимущества:

- высококачественная графика и продвинутые эффекты;
- подходит для сложных проектов;
- не требуется приобретение лицензии.

Недостатки:

- высокие системные требования;
- сложность оптимизации для слабых устройств;
- ограниченная документация по мобильной разработке.

Godot Engine – игровой движок с открытым исходным кодом, подходит для 2D и 3D игр, изображен на рисунке 6.

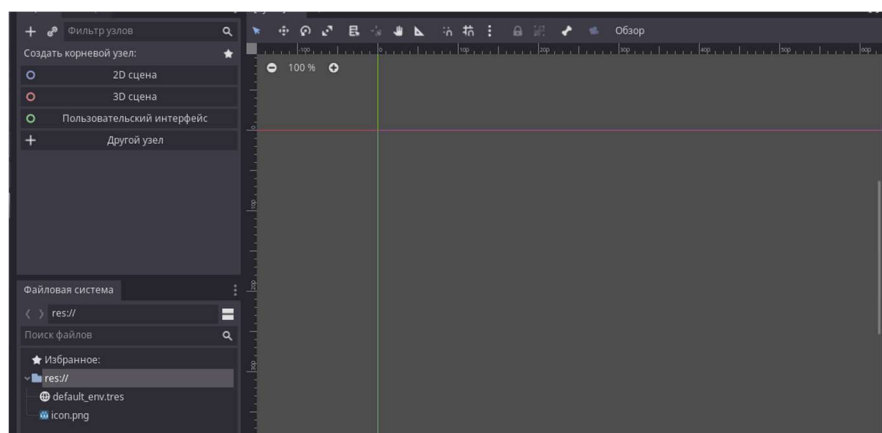


Рисунок 6 – Интерфейс Unreal Engine

Преимущества:

- быстрый запуск и сборка;
- низкие системные требования;
- удобные встроенные инструменты.

Недостатки:

- менее развитая документация;
- недостаток инструментов.

После описания основных характеристик трех платформ была составлена сравнительная таблица, позволяющая принять обоснованное решение при выборе.

Таблица 2 – Сравнительная таблица игровых платформ

Критерий	Unity	Unreal Engine	Godot Engine
Графика	Хорошая	Отличная	Средняя
Производительность	Оптимальная	Требовательная	Легкая
Язык программирования	C#	C++/Blueprints	C#/GDScript
Мобильная оптимизация	Лучшая	Средняя	Ограниченная
Документация	Огромная	Большая	Средняя
Стоимость	Бесплатно	Бесплатно	Бесплатно

На основе сравнительной таблицы 2 сделан вывод, что Unity является оптимальным выбором, сочетая в себе простоту освоения, хорошую производительность на Android и все необходимые инструменты для реализации.

2.2 Графический редактор

В создании мобильного приложения важную роль играет создание визуальных элементов, то есть фонов, интерфейсов, объектов и других графических ресурсов. В данной области широко применяются профессиональные редакторы. Наиболее популярные:

- Adobe Photoshop;
- Adobe Illustrator.

Adobe Photoshop — это профессиональное программное обеспечение для редактирования изображений, самый известный и широко используемый графический редактор в мире [6]. Данное приложение изображено на рисунке 7.

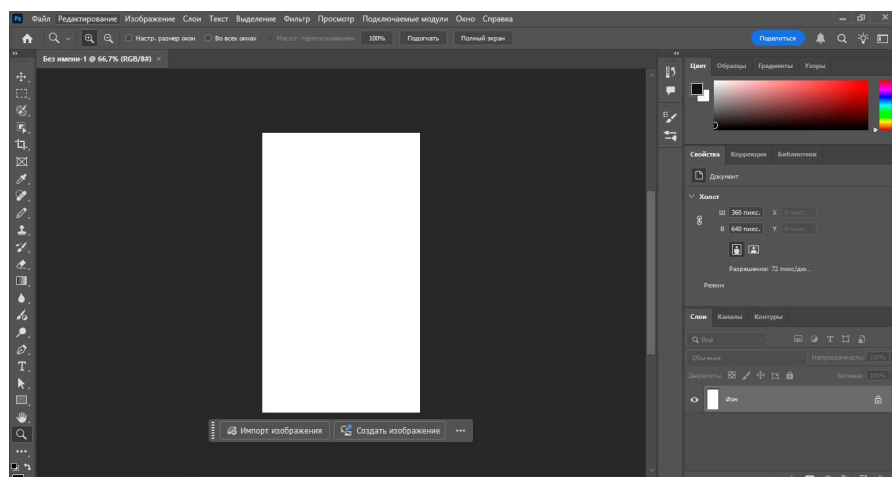


Рисунок 7 – Приложение Adobe Photoshop

Преимущества:

- поддержка растровой графики, слоев и масок. Подходит для спрайтов и текстур;
- широкий набор инструментов для рисования и редактирования;
- совместимость с игровыми движками. Формат PNG легко импортируется в Unity;
- большое количество обучающих материалов.

Недостатки:

- большой размер файлов;
- высокие системные требования;
- имеет платные инструменты.

Adobe Illustrator – векторный графический редактор, предназначенный для создания масштабируемой графики, логотипов, иконок и иллюстраций. Интерфейс приложения представлен на рисунке 8.

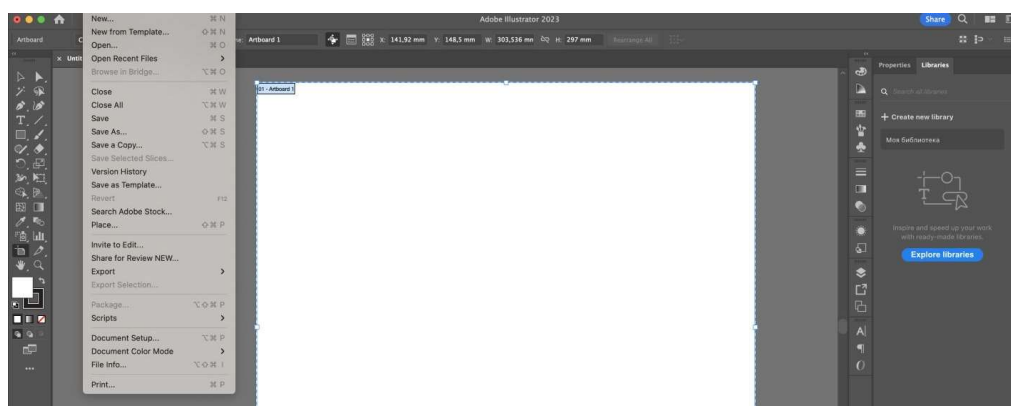


Рисунок 8 – Приложение Adobe Illustrator

Преимущества:

- векторная графика. Изображения масштабируются без потери качества;
- точные инструменты рисования. Подходит для создания интерфейсных элементов;
- меньше размер файлов. Векторные изображения весят меньше растровых.

Недостатки:

- не подходит для создания растровой графики. Неудобен для создания спрайтов и текстур;
- сложность создания сложных текстур. Ограниченный набор инструментов для работы с пикселями.

На основе основных критерий была составлена таблица 3 для выбора

графического редактора.

Таблица 3 – Сравнение графических редакторов

Критерий	Photoshop	Illustrator
Тип графики	Растровая	Векторная
Масштабирование	Теряет качество	Без потерь
Интерфейс игры	Подходит для текстур	Не подходит для текстур
Экспорт	PNG, JPG	SVG, EPS
Сложность освоения	Средняя	Выше средней

Более универсальным и удобным инструментом является Adobe Photoshop, так как данный редактор лучше подходит для создания и обработки текстур и визуальных элементов. Имеет удобный экспорт и гибкий набор инструментов, которые можно комбинировать.

2.3 Музыкальный редактор

Создание звукового оформления имеет большую важность, так как влияет на вовлеченность пользователя. Для создания аудиоконтента разработчики используют цифровые аудиостанции. Будут рассмотрены два популярных редактора:

- BandLab;
- FL Studio.

BandLab — это платформа для создания музыки. Она позволяет записывать и редактировать аудио, добавлять эффекты, микшировать треки и экспортировать аудио в различных форматах. Платформа изображена на рисунке 9.

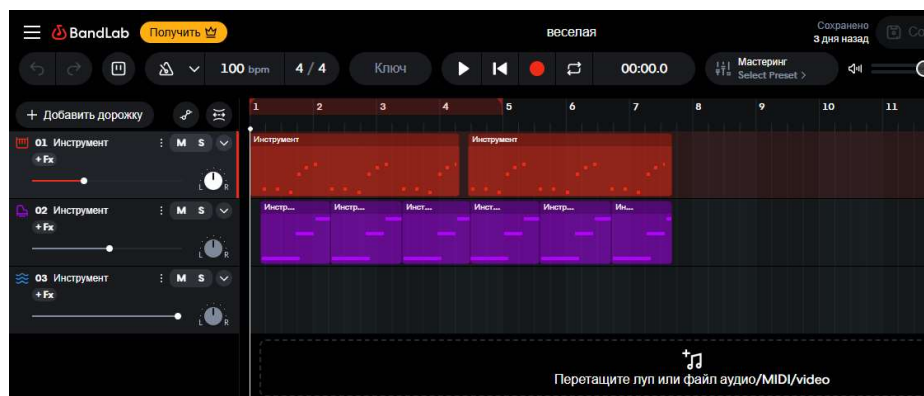


Рисунок 9 – Платформа BandLab

Преимущества:

- не требуется приобретение лицензии;
- простота интерфейса;
- облачное хранение.

Недостатки:

- ограниченные возможности тонкой настройки звука;
- зависимость от интернет-соединения.

FL Studio – профессиональная цифровая аудиостанция, широко используемая в музыкальной индустрии. Интерфейс и инструменты редактора представлены на рисунке 10.



Рисунок 10 – Платформа FL Studio

Преимущества:

- профессиональный набор инструментов;
- возможность создания сложных композиций;
- точное редактирование нот и ритма.

Недостатки:

- сложный интерфейс;
- платная лицензия;
- высокие системные требования.

Таблица 4 – Сравнительная таблица музыкальных редакторов

Критерий	BandLab	FL Studio
Стоимость	Бесплатный	Платный
Сложность освоения	Низкая	Высокая
Требования к устройству	Низкие	Высокие
Функциональность	Базовая	Профессиональная

На основе таблицы для создания игры на Android более предпочтительным выбором является редактор BandLab. Он не требует специальных навыков работы с профессиональными станциями, при этом предоставляет необходимые инструменты для создания качественной музыкальной составляющей.

2.4 Язык программирования

Так как Unity была выбрана как игровая платформа, то для написания кода будет использован язык C# в Microsoft Visual Studio 2022 [1]. На рисунке 11 показан интерфейс данного приложения.

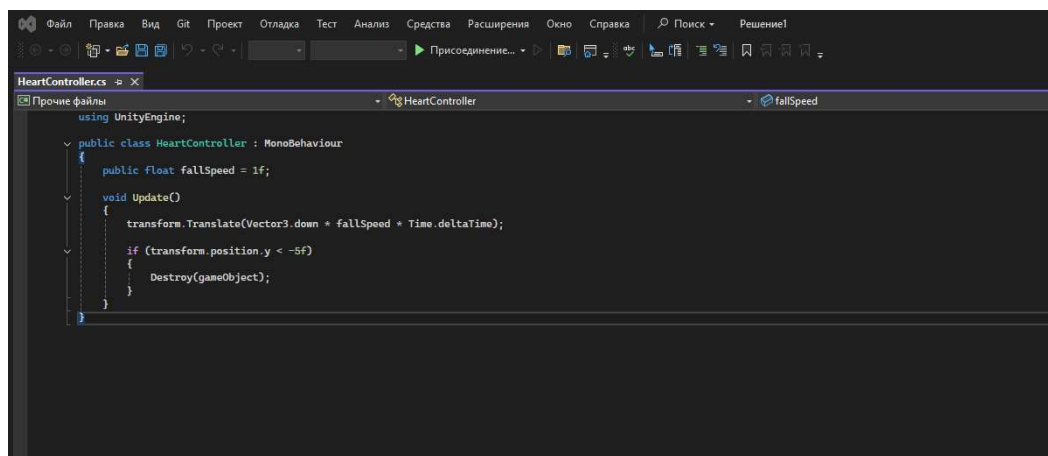


Рисунок 11 – Приложение Microsoft Visual Studio 2022

C# — это объектно-ориентированный ЯП, он широко используется для создания различных типов приложений. Данный язык является основным языком для разработки в Unity, что обеспечивает максимальную совместимость и поддержку всех функций движка [7].

Преимущества:

- компилируемый язык, что обеспечивает высокую скорость выполнения кода по сравнению с другими языками [8];
- поддержка многопоточности позволяет эффективно распределять вычислительные нагрузки;
- четкий и структурированный синтаксис, облегчает написание и поддержку кода;
- встроенные механизмы обработки исключений повышают стабильность приложения;
- обилие учебных материалов, форумов и готовых решений для Unity на C#.

ЯП C# демонстрирует полную функциональную достаточность для решения всех поставленных задач разработки.

3 Архитектура мобильной квест - игры

Архитектура строится с учетом принципов модульности, масштабируемости и оптимальной производительности на мобильных устройствах. Основное внимание уделяется ключевым компонентам системы, их взаимодействию и структурной организации [9]. Архитектура игры делится на 2 основных части.

Клиентская сторона в себя включает несколько параметров:

- игровой движок. Основная платформа для разработки, содержащая инструменты для создания 2D графики, анимации, физики и управления;
- сцены. Каждая локация игры представлена отдельной сценой, которая загружается по мере необходимости;
- игровая логика. Скрипты на C# для управления игровым процессом, включая взаимодействие с объектами, управление персонажем и обработку событий [10];
- пользовательский интерфейс. Элементы интерфейса, такие как меню, инвентарь и диалоги;
- аудио. Звуковые эффекты и музыка, используемые для создания атмосферы и улучшения игрового опыта.

Контент включает в себя:

- графика. 2D модели, текстуры и анимации, используемые для визуализации игровых объектов и персонажей;
- сценарий. Логика квестов, включая задачи и диалоги.

В разрабатываемой игровой системе предусмотрен единственный тип пользователя – «Игрок». Его основные критерии:

- управление игровым сеансом. Инициализация нового игрового процесса. Завершение текущего процесса.

– навигация по пространству. Последовательное перемещение между локациями (комнатами). Взаимодействие с интерактивными элементами окружения.

– решение игровых задач. Выполнение логических головоломок. Взаимодействие с анимированными объектами. Активация интерактивных элементов управления.

– завершение игрового цикла. Рестарт игрового процесса. Окончательное завершение игровой сессии.

Данный функционал реализует базовый игровой цикл, обеспечивая последовательное прохождение контента с возможностью повторного прохождения или завершения игры по желанию пользователя. На основе анализа основных критериев пользователя была создана диаграмма вариантов использования, она представлена на рисунке 12.



Рисунок 12 – UML – диаграмма вариантов использования

Описание основных взаимодействий:

- Пользователь нажимает на «Начало игры».
- Система последовательно загружает три комнаты с квестами.

- После каждой головоломки передается часть кода.
- Финальная проверка кода с двумя вариантами исхода.

На основе этого была сделана UML – диаграмма последовательности, она изображена на рисунке 13.

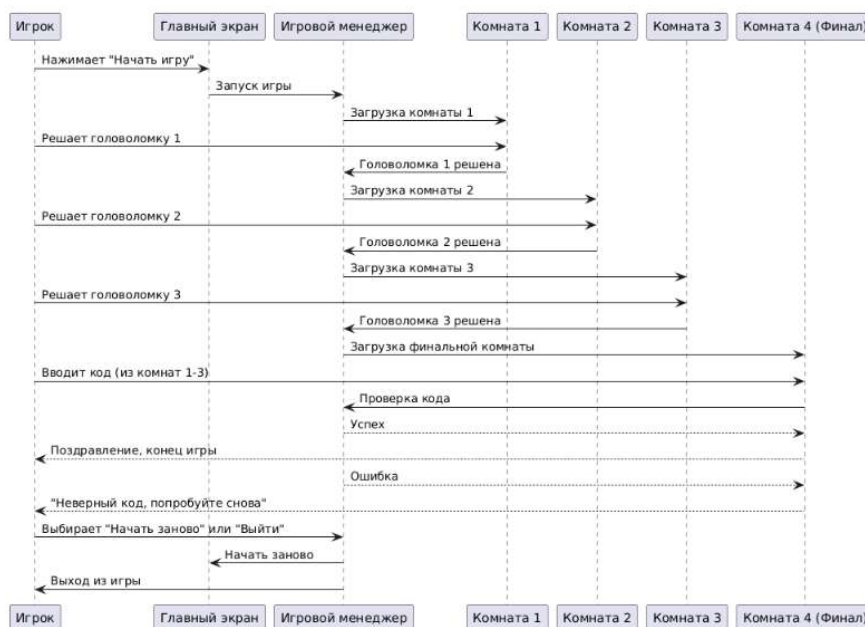


Рисунок 13 – UML – диаграмма последовательности

Разрабатываемая игра должна демонстрировать стабильную производительность и функционирование всех систем [1]. Это требование обусловлено необходимостью обеспечения комфортного игрового процесса для пользователя. Поэтому ниже представлены минимальные требования к устройству игрока для корректного использования приложения:

- Операционная система: Android 8.0 и выше
- Процессор: 2.0 ГГц, минимум 4 ядра
- Оперативная память (RAM): 3 ГБ и выше
- Свободное место на устройстве: от 150 МБ
- Разрешение экрана: поддержка современных экранов с соотношением сторон 16:9 и выше
- Интернет-соединение: не требуется

4 Реализация и тестирование приложения

Первый этап в разработке квест - приложения — это создание сцен, то есть начального, конечного экрана, комнат и экранов с логическими загадками. Для каждой комнаты была отрисована уникальная основа [11]. Чтобы в игровом процессе пользователь мог взаимодействовать с предметами, другими слоями были нарисованы объекты для интерактива и основных квестовых заданий [12]. В 1 комнате – «Крестики- нолики», во 2 – «Стрельба по уткам», в 3 – «Отгадай загадку», в 4 – «Введите код». Все сцены представлены на рисунках 14 - 16.



Рисунок 14 – Отрисовка начального и конечного экрана



Рисунок 15 – Все комнаты с добавлением объектов для взаимодействия



Рисунок 16 – Экраны квестов

Для каждой сцены были написаны скрипты. В файле Starter.cs представлен скрипт на C#, который выполняет две основные функции, «Запуск игры» и «Выход» из нее на начальном экране. Скрипт служит контроллером для кнопок меню [13].

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Starter : MonoBehaviour
{
    public void StartGame()
    {
        SceneManager.LoadScene("Room1");
    }

    public void QuitGame()
    {
        #if UNITY_EDITOR
            UnityEditor.EditorApplication.isPlaying = false;
        #else
            Application.Quit();
        #endif
    }
}
```

Также был нарисован инвентарь для объектов и написан код. Код в файле InventorySystem.cs реализует простую систему управления с фиксированным количеством слотов.

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class InventorySystem : MonoBehaviour
```

```

{
    public InventoryItem[] inventorySlots = new InventoryItem[3];
    public LockController lockController;

    void Awake()
    {
        InitializeInventoryItems();
    }

    private void InitializeInventoryItems()
    {
        for (int i = 0; i < inventorySlots.Length; i++)
        {
            if (inventorySlots[i] != null)
            {
                inventorySlots[i].Initialize(i, this);
            }
        }
    }

    public void OnItemClicked(int itemId)
    {
        lockController?.OnInventoryItemClicked(itemId);
    }

    public Sprite GetItemSprite(int itemId)
    {
        return IsValidItemId(itemId) ? inventorySlots[itemId].ItemSprite : null;
    }

    public void SetItemVisible(int itemId, bool visible)
    {
        if (IsValidItemId(itemId))
        {
            inventorySlots[itemId].gameObject.SetActive(visible);
        }
    }

    public void SetItemsInteractable(bool interactable)
    {
        foreach (var slot in inventorySlots)
        {
            if (slot != null) slot.SetInteractable(interactable);
        }
    }

    private bool IsValidItemId(int itemId) => itemId >= 0 && itemId < inventorySlots.Length;
}

```

Данный скрипт взаимодействует с элементами инвентаря InventoryItem.cs и контроллером блокировок LockController.cs.

В первой комнате головоломка заключается в игре «Крестики-нолики» с роботом, который ставит знак случайно. Код в файле RobotDialog.cs реализует беседу с роботом, который последовательно выводит фразы с эффектом печати и запускает игру. После победы он показывает другой диалог с наградой. Это показано на рисунке 17.

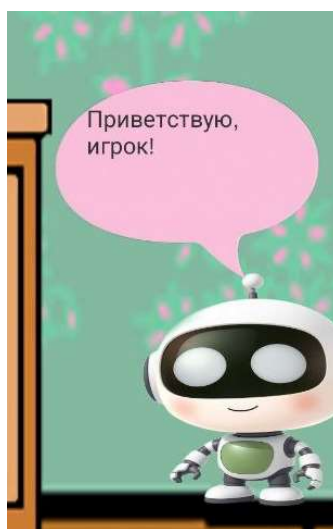


Рисунок 17 – Диалог с роботом

Реализация первой загадки с выводом статуса находится в файле TicTacToeGame.cs. За переход на другую локацию отвечает скрипт в NextSceneArrow.cs. Он использует интерфейс 'IPointerClickHandler' для обработки касаний.

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using UnityEngine.EventSystems;
using System.Collections;

public class NextSceneArrow : MonoBehaviour, IPointerClickHandler
{
    [Header("Settings")]
    public string nextSceneName;

    private bool isTransitioning = false;
```

```

public void OnPointerClick(PointerEventData eventData)
{
    if (!isTransitioning && !string.IsNullOrEmpty(nextScene-
Name))
    {
        TransitionToNextScene();
    }
}

private void TransitionToNextScene()
{
    isTransitioning = true;

    SceneManager.LoadScene(nextSceneName);
}
}

```

Требуется кликнуть на монитор во второй комнате, чтобы начать прохождение другого квеста. Для него были написаны код в файле DuckController.cs, который управляет поведением «уток» в игре, реализуя движение к заданной точке, обработку выстрелов по ним и визуальную отладку, и в DuckHuntGame.cs, который отвечает за появление элементов, подсчет очков, интерфейс и завершение игры. На рисунке 18 показано, как это выглядит в приложении.



Рисунок 18 – Интерфейс игры с утками

С предпоследней локации нужно нажать на растение, чтобы открылась последняя загадка. Она реализована в файле RiddleSystem.cs, он содержит

систему вопросов с тремя вариантами ответов, отображением панели, плавным появлением затемнённого фона, обработкой выбора и показом стрелки после правильного ответа.

В последней комнате необходимо ввести в правильном порядке полученные цифры ранее, чтобы доиграть до конца. Скрипт «LockController» реализует игровую логику замка с кодовым вводом через предметы инвентаря. Он управляет UI-панелью замка, отслеживает ввод кода с помощью предметов из инвентаря и проверяет правильность введённого кода.

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

public class LockController : MonoBehaviour, IPointerClickHandler
{
    [Header("Settings")]
    public int correctCode = 138;
    public float resetDelay = 2f;

    [Header("References")]
    public GameObject lockPanel;
    public Image[] codeSlots = new Image[3];
    public InventorySystem inventorySystem;
    public GameObject замок;
    public GameObject arrowObject;

    private int[] enteredDigits = new int[3] { -1, -1, -1 };
    private int currentSlotIndex = 0;
    private bool isLockActive = false;

    private Dictionary<int, int> itemIdToDigitMap = new Dictionary<int, int>()
    {
        { 0, 1 },
        { 1, 3 },
        { 2, 8 }
    };

    private void Awake()
    {
        if (arrowObject != null) arrowObject.SetActive(false);
        lockPanel.SetActive(false);
        InitializeCodeSlots();
    }
}
```

```

private void InitializeCodeSlots()
{
    foreach (var slot in codeSlots)
    {
        if (slot != null)
        {
            slot.gameObject.SetActive(true);
            slot.sprite = null;
            slot.color = new Color(1, 1, 1, 0.5f);
        }
    }
}

public void OnPointerClick(PointerEventData eventData)
{
    if (!isLockActive) OpenLock();
}

private void OpenLock()
{
    isLockActive = true;
    lockPanel.SetActive(true);
    inventorySystem.SetItemsInteractable(true);
    currentSlotIndex = 0;
}

private void CloseLock()
{
    isLockActive = false;
    lockPanel.SetActive(false);
    StartCoroutine(CloseGame());
    zamok.SetActive(false);
    inventorySystem.SetItemsInteractable(false);
}

public IEnumerator CloseGame()
{
    if (arrowObject != null)
    {
        arrowObject.SetActive(true);

        CanvasGroup arrowCanvas = arrowObject.GetComponent<CanvasGroup>();
        if (arrowCanvas != null)
        {
            arrowCanvas.alpha = 0;
            float fadeTime = 1f;
            float timer = 0;

            while (timer < fadeTime)
            {

```

```

        timer += Time.deltaTime;
        arrowCanvas.alpha = Mathf.Lerp(0, 1, timer /
fadeTime);
        yield return null;
    }
}

}

}

}

public void OnInventoryItemClicked(int itemId)
{
    if (!isLockActive || currentSlotIndex >= codeSlots.Length)
return;

    var sprite = inventorySystem.GetItemSprite(itemId);
    if (sprite == null) return;

    codeSlots[currentSlotIndex].gameObject.SetActive(true);
    codeSlots[currentSlotIndex].sprite = sprite;
    codeSlots[currentSlotIndex].color = Color.white;

    enteredDigits[currentSlotIndex] = item-
IdToDigitMap[itemId];
    inventorySystem.SetItemVisible(itemId, false);

    currentSlotIndex++;

    if (currentSlotIndex >= codeSlots.Length)
    {
        CheckCode();
    }
}

private void CheckCode()
{
    int enteredCode = enteredDigits[0] * 100 + enteredDig-
its[1] * 10 + enteredDigits[2];

    if (enteredCode == correctCode)
    {
        Debug.Log("Code correct!");
        CloseLock();
    }
    else
    {
        Debug.Log("Wrong code!");
        StartCoroutine(ResetLock());
    }
}

private IEnumerator ResetLock()
{

```

```

        yield return new WaitForSeconds(resetDelay);

        for (int i = 0; i < enteredDigits.Length; i++)
        {
            if (enteredDigits[i] != -1)
            {
                foreach (var pair in itemIdToDigitMap)
                {
                    if (pair.Value == enteredDigits[i])
                    {
                        inventorySystem.SetItemVisible(pair.Key,
true);
                        break;
                    }
                }
                enteredDigits[i] = -1;
            }
        }

        InitializeCodeSlots();
        currentSlotIndex = 0;
    }
}

```

В мобильное приложение были добавлены не только головоломки, но и интерактивные элементы [7]:

- Переключатель (toggle) для UI-элемента с изображением фонаря, который меняет спрайт при клике в файле LanternToggle.cs.

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

[RequireComponent(typeof(Image))]
public class LanternToggle : MonoBehaviour, IPointerClickHandler
{
    public Sprite lanternOnSprite;
    public Sprite lanternOffSprite;

    private Image lanternImage;
    private bool isOn = false;

    void Awake()
    {
        lanternImage = GetComponent<Image>();
        lanternImage.sprite = lanternOffSprite;
    }

    public void OnPointerClick(PointerEventData eventData)
    {

```

```

        ToggleLantern();
    }

    private void ToggleLantern()
    {
        isOn = !isOn;
        lanternImage.sprite = isOn ? lanternOnSprite : lanternOff-
Sprite;
    }
}

```

– При нажатии на часы, происходит переключение между нормальным и "сломленным" состоянием, а также добавляет эффект тряски при повторном клике, за это отвечает скрипт «Clock».

– Интерактивное поведение собаки, реагирующей на клики пользователя, представлено в файле InteractiveDog.cs. При нажатии собака либо издаёт лай, либо показывает анимированное сердце.

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections;

[RequireComponent(typeof(Image))]
public class InteractiveDog : MonoBehaviour, IPointerClickHandler
{
    [Header("Dog Settings")]
    public AudioClip[] barkSounds;
    public Sprite heartSprite;
    public Transform heartSpawnPoint;

    [Header("Heart Settings")]
    public Vector2 heartSize = new Vector2(100, 100);
    public float heartScale = 1f;

    [Header("Timing Settings")]
    public float actionDelay = 3f;
    public float heartDisplayTime = 1f;
    public float fadeDuration = 0.3f;

    [Header("Probabilities")]
    [Range(0, 100)] public int heartChance = 70;

    private AudioSource audioSource;
    private bool canInteract = true;
    private Image dogImage;
}

```

```

void Awake()
{
    dogImage = GetComponent<Image>();
    audioSource = GetComponent<AudioSource>();
    if (audioSource == null)
    {
        audioSource = gameObject.AddComponent<AudioSource>();
    }
}

public void OnPointerClick(PointerEventData eventData)
{
    if (canInteract)
    {
        StartCoroutine(PerformAction());
    }
}

private IEnumerator PerformAction()
{
    canInteract = false;

    bool showHeart = Random.Range(0, 100) < heartChance;

    if (showHeart)
    {
        ShowHeart();
    }
    else
    {
        Bark();
    }

    yield return new WaitForSeconds(actionDelay);
    canInteract = true;
}

private void Bark()
{
    if (barkSounds.Length > 0)
    {
        AudioClip randomBark = barkSounds[Random.Range(0,
barkSounds.Length)];
        audioSource.PlayOneShot(randomBark);
    }
}

private void ShowHeart()
{
    if (heartSprite != null && heartSpawnPoint != null)
    {

```

```

        GameObject heartObj = new GameObject("Heart");
        heartObj.transform.SetParent(transform.parent);
        heartObj.transform.position = heartSpawnPoint.position;
        heartObj.transform.localScale = Vector3.one * heartScale;

        Image heartImage = heartObj.AddComponent<Image>();
        heartImage.sprite = heartSprite;

        RectTransform rt = heartObj.GetComponent<RectTransform>();
        rt.sizeDelta = heartSize;

        CanvasGroup canvasGroup = heartObj.AddComponent<CanvasGroup>();
        StartCoroutine(FadeAndDestroyHeart(heartObj));
    }

    private IEnumerator FadeAndDestroyHeart(GameObject heartObj)
    {
        CanvasGroup canvasGroup = heartObj.GetComponent<CanvasGroup>();
        float timer = 0f;

        yield return new WaitForSeconds(heartDisplayTime - fadeDuration);

        while (timer < fadeDuration)
        {
            timer += Time.deltaTime;
            if (canvasGroup != null)
            {
                canvasGroup.alpha = 1f - (timer / fadeDuration);
            }
            yield return null;
        }

        if (heartObj != null)
        {
            Destroy(heartObj);
        }
    }
}

```

Все дополнительные визуальные эффекты представлены на рисунке 19.

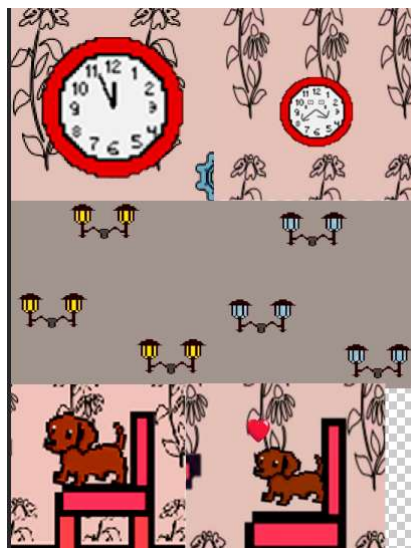


Рисунок 19 – Визуальные интерактивы

Скрипт «MusicManager» — это класс для управления музыкой в игре. Он автоматически воспроизводит музыкальные треки при загрузке различных сцен и сохраняется между сценами. Созданные звуковые эффекты представлены на рисунке 20.

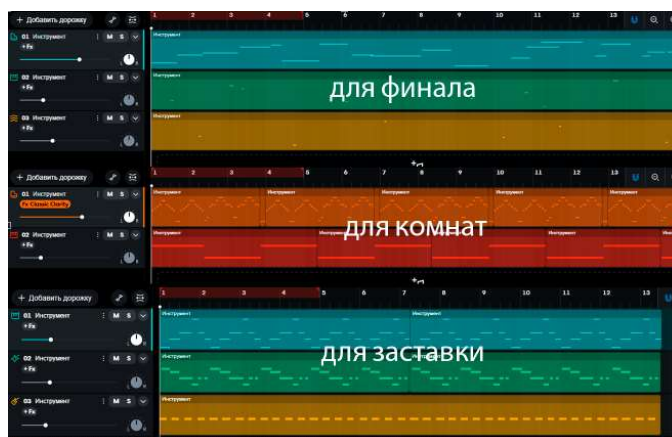


Рисунок 20 – Создание аудиосопровождения

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class MusicManager : MonoBehaviour
{
    public static MusicManager Instance { get; private set; }

    [Header("Music Tracks")]
```

```

public AudioClip mainMenuMusic;
public AudioClip roomMusic;
public AudioClip LastRoomMusic;

[Header("Settings")]
[Range(0f, 1f)] public float volume = 0.5f;
public bool loopMusic = true;

private AudioSource audioSource;

void Awake()
{
    if (Instance == null)
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
        audioSource = gameObject.AddComponent<AudioSource>();
        audioSource.volume = volume;
        audioSource.loop = loopMusic;
    }
    else
    {
        Destroy(gameObject);
    }
}

void OnEnable()
{
    SceneManager.sceneLoaded += OnSceneLoaded;
}

void OnDisable()
{
    SceneManager.sceneLoaded -= OnSceneLoaded;
}

private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
{
    switch (scene.name)
    {
        case "StartScene":
            PlayMusic(mainMenuMusic);
            break;
        case "Room1":
            PlayMusic(roomMusic);
            break;
        case "Room2":
            PlayMusic(roomMusic);
            break;
        case "Room3":
            PlayMusic(roomMusic);
            break;
    }
}

```

```

        case "LastRoom":
            PlayMusic(LastRoomMusic);
            break;
        default:
            PlayMusic(mainMenuMusic);
            break;
    }
}

public void PlayMusic(AudioClip clip)
{
    if (clip == null || audioSource.clip == clip && audioSource.isPlaying)
        return;

    audioSource.clip = clip;
    audioSource.Play();
}

public void StopMusic()
{
    audioSource.Stop();
}

public void SetVolume(float newVolume)
{
    volume = Mathf.Clamp01(newVolume);
    audioSource.volume = volume;
}

public void ToggleMute()
{
    audioSource.mute = !audioSource.mute;
}
}

```

Заключение

В ходе выполнения дипломной работы было разработано мобильное игровое приложение для Android в жанре квест с использованием платформы Unity. В процессе создания были успешно решены все поставленные задачи, включая проектирование архитектуры, реализацию игровых механик, создание визуального и звукового оформления, а также тестирование и оптимизацию приложения. Был применен подход для разделения нагрузки между компонентами и оптимизирована работа с ресурсами для мобильных устройств.

В результате реализована система последовательных головоломок, создан интерактивный инвентарь и система взаимодействия с предметами. Также была внедрена проверка финального кода, собранного из частей, полученных в каждой локации.

Разработанное приложение имеет перспективы развития, например, могут быть добавлены новые уровни и головоломки, внедрена система достижений и сохранения прогресса, разработка режима для совместного прохождения.

В данной работе применены современные подходы к разработке мобильных игр, включая оптимизацию ресурсов и адаптацию под различные устройства. Созданный продукт демонстрирует эффективное использование движка Unity и языка программирования C#.

Все цели дипломного проекта были достигнуты. Разработанное приложение соответствует поставленным задачам и предоставляет пользователям увлекательный и образовательный опыт. Полученные в ходе работы знания и навыки могут быть применены в дальнейшей профессиональной деятельности в области игровой разработки.

Список использованных источников

1. Корнилов А.В. / Unity. Полное руководство / ISBN 978-5-94387-721-6
2. Создание игр на Unity: с чего начать? / <https://habr.com/ru/articles/828302/> / (дата обращения: 18.05.2025).
3. Руководство: Начало разработки под Android / <https://docs.unity3d.com/ru/530/Manual/android-GettingStarted.html> / (дата обращения 06.05.2025).
4. Разработка 2D-игр / <https://docs.unity3d.com/Manual/Unity2D.html> / (дата обращения 11.05.2025).
5. Изучение интерфейса / <https://docs.unity3d.com/ru/530/Manual/LearningtheInterface.html> / (дата обращения: 19.05.2025).
6. Дунаев В. / Photoshop CS6: Понятный самоучитель / ISBN 978-5-459-01691-8
7. Бонд Д.Г. / Unity и C#. Геймдев от идеи до реализации / ISBN 978-5-4461-0715-5
8. Подбельский В.В. / Программирование базовый курс C# / ISBN 978-5-534-10616-9
9. Язык описания квестов или как сделать квестовую систему на Unity / <https://habr.com/ru/articles/676010/> / (дата обращения: 18.05.2025).
10. Ферроне Х. / Изучаем C# через разработку игр на Unity / ISBN 978-5-4461-2932-4
11. Пиксель-арт для начинающих: инструкция по применению / <https://habr.com/ru/companies/playgendary/articles/482536/> / (дата обращения 13.05.2025).
12. Основные сведения о рабочей среде Adobe Photoshop / <https://helpx.adobe.com/ru/photoshop/using/workspace-basics.html> / (дата обращения: 07.05.2025).
13. Денисов Д. / Разработка игры в Unity. С нуля и до реализации / ISBN 978-5-532-94186-1

Приложение А

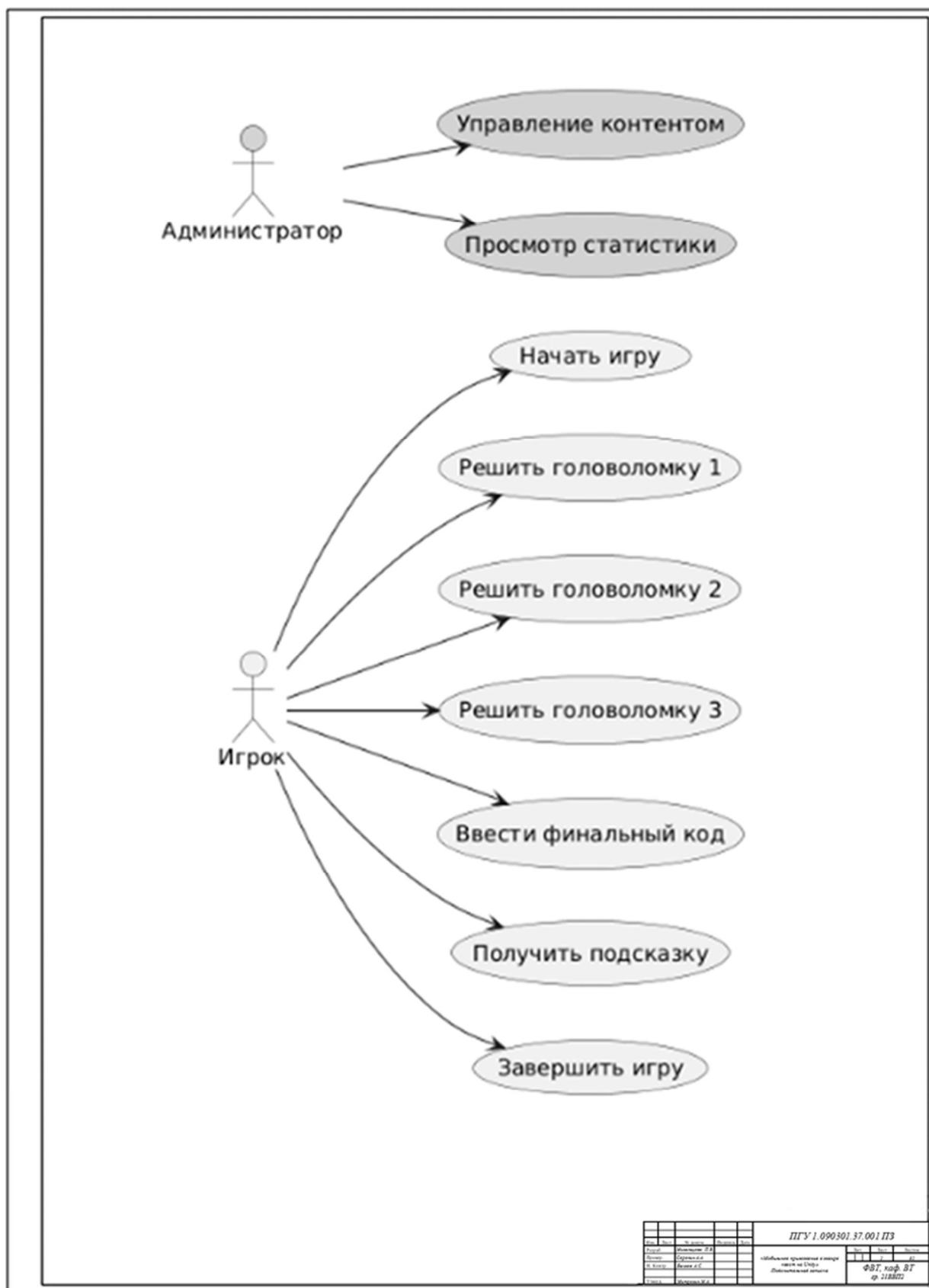
(обязательное)

Мобильное приложение в жанре квест на Unity

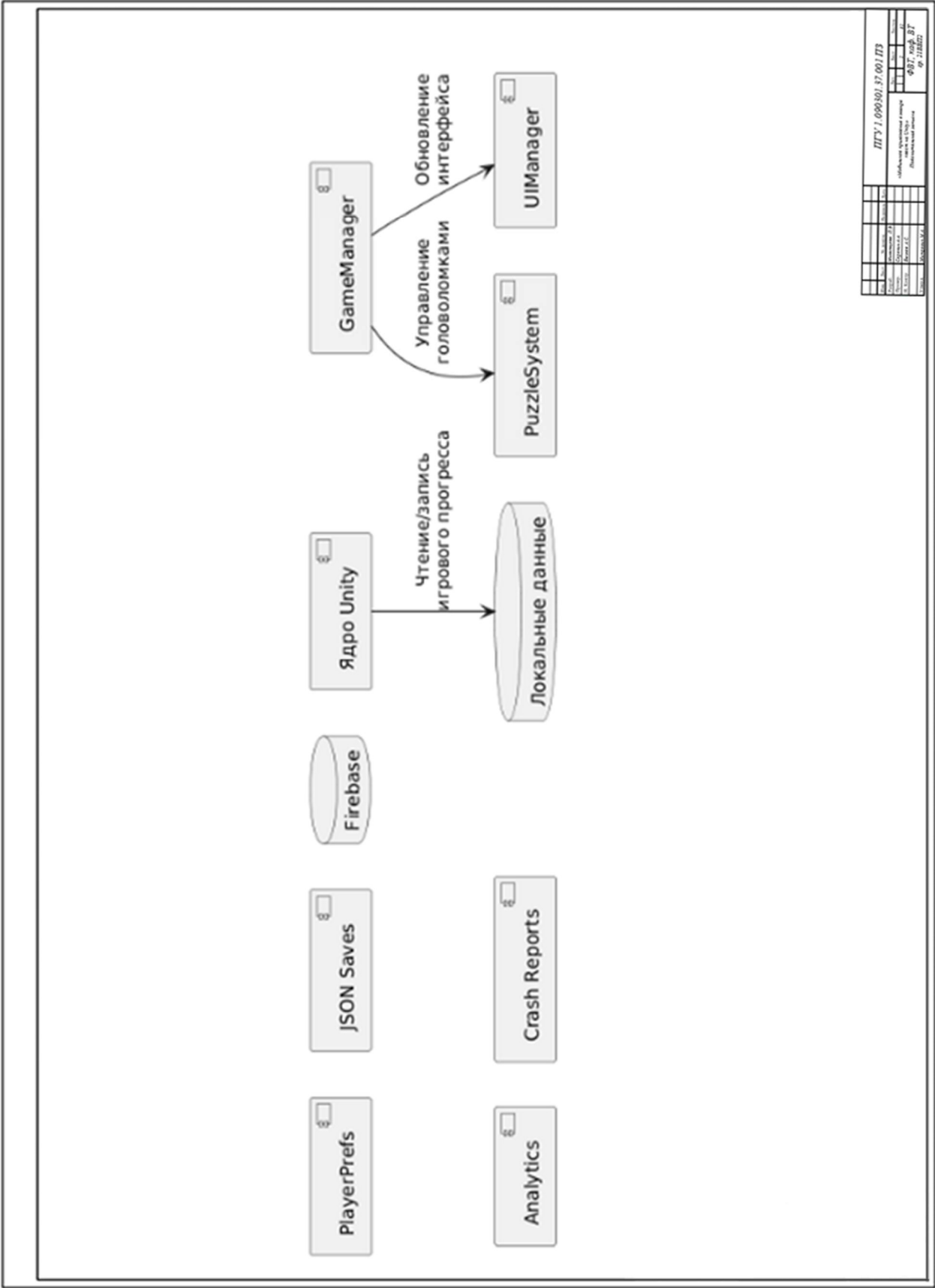
UML-диаграммы

					ПГУ 1.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		44

Приложение А.1 - Диаграмма прецедентов



Приложение А.2 - Диаграмма развертывания



ПГУ 1.090301.37.001 ПЗ			
Изм.	Лист	№ докум	Подпись
1	1	1.090301.37.001 ПЗ	
2	2	1.090301.37.001 ПЗ	
3	3	1.090301.37.001 ПЗ	
4	4	1.090301.37.001 ПЗ	
5	5	1.090301.37.001 ПЗ	
6	6	1.090301.37.001 ПЗ	
7	7	1.090301.37.001 ПЗ	
8	8	1.090301.37.001 ПЗ	
9	9	1.090301.37.001 ПЗ	
10	10	1.090301.37.001 ПЗ	
11	11	1.090301.37.001 ПЗ	
12	12	1.090301.37.001 ПЗ	
13	13	1.090301.37.001 ПЗ	
14	14	1.090301.37.001 ПЗ	
15	15	1.090301.37.001 ПЗ	
16	16	1.090301.37.001 ПЗ	
17	17	1.090301.37.001 ПЗ	
18	18	1.090301.37.001 ПЗ	
19	19	1.090301.37.001 ПЗ	
20	20	1.090301.37.001 ПЗ	
21	21	1.090301.37.001 ПЗ	
22	22	1.090301.37.001 ПЗ	
23	23	1.090301.37.001 ПЗ	
24	24	1.090301.37.001 ПЗ	
25	25	1.090301.37.001 ПЗ	
26	26	1.090301.37.001 ПЗ	
27	27	1.090301.37.001 ПЗ	
28	28	1.090301.37.001 ПЗ	
29	29	1.090301.37.001 ПЗ	
30	30	1.090301.37.001 ПЗ	
31	31	1.090301.37.001 ПЗ	
32	32	1.090301.37.001 ПЗ	
33	33	1.090301.37.001 ПЗ	
34	34	1.090301.37.001 ПЗ	
35	35	1.090301.37.001 ПЗ	
36	36	1.090301.37.001 ПЗ	
37	37	1.090301.37.001 ПЗ	
38	38	1.090301.37.001 ПЗ	
39	39	1.090301.37.001 ПЗ	
40	40	1.090301.37.001 ПЗ	
41	41	1.090301.37.001 ПЗ	
42	42	1.090301.37.001 ПЗ	
43	43	1.090301.37.001 ПЗ	
44	44	1.090301.37.001 ПЗ	
45	45	1.090301.37.001 ПЗ	
46	46	1.090301.37.001 ПЗ	
47	47	1.090301.37.001 ПЗ	
48	48	1.090301.37.001 ПЗ	
49	49	1.090301.37.001 ПЗ	
50	50	1.090301.37.001 ПЗ	
51	51	1.090301.37.001 ПЗ	
52	52	1.090301.37.001 ПЗ	
53	53	1.090301.37.001 ПЗ	
54	54	1.090301.37.001 ПЗ	
55	55	1.090301.37.001 ПЗ	
56	56	1.090301.37.001 ПЗ	
57	57	1.090301.37.001 ПЗ	
58	58	1.090301.37.001 ПЗ	
59	59	1.090301.37.001 ПЗ	
60	60	1.090301.37.001 ПЗ	
61	61	1.090301.37.001 ПЗ	
62	62	1.090301.37.001 ПЗ	
63	63	1.090301.37.001 ПЗ	
64	64	1.090301.37.001 ПЗ	
65	65	1.090301.37.001 ПЗ	
66	66	1.090301.37.001 ПЗ	
67	67	1.090301.37.001 ПЗ	
68	68	1.090301.37.001 ПЗ	
69	69	1.090301.37.001 ПЗ	
70	70	1.090301.37.001 ПЗ	
71	71	1.090301.37.001 ПЗ	
72	72	1.090301.37.001 ПЗ	
73	73	1.090301.37.001 ПЗ	
74	74	1.090301.37.001 ПЗ	
75	75	1.090301.37.001 ПЗ	
76	76	1.090301.37.001 ПЗ	
77	77	1.090301.37.001 ПЗ	
78	78	1.090301.37.001 ПЗ	
79	79	1.090301.37.001 ПЗ	
80	80	1.090301.37.001 ПЗ	
81	81	1.090301.37.001 ПЗ	
82	82	1.090301.37.001 ПЗ	
83	83	1.090301.37.001 ПЗ	
84	84	1.090301.37.001 ПЗ	
85	85	1.090301.37.001 ПЗ	
86	86	1.090301.37.001 ПЗ	
87	87	1.090301.37.001 ПЗ	
88	88	1.090301.37.001 ПЗ	
89	89	1.090301.37.001 ПЗ	
90	90	1.090301.37.001 ПЗ	
91	91	1.090301.37.001 ПЗ	
92	92	1.090301.37.001 ПЗ	
93	93	1.090301.37.001 ПЗ	
94	94	1.090301.37.001 ПЗ	
95	95	1.090301.37.001 ПЗ	
96	96	1.090301.37.001 ПЗ	
97	97	1.090301.37.001 ПЗ	
98	98	1.090301.37.001 ПЗ	
99	99	1.090301.37.001 ПЗ	
100	100	1.090301.37.001 ПЗ	

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.37.001 ПЗ

Приложение Б

(Обязательное)

Мобильное приложение в жанре квест на Unity

Листинг программы

					ПГУ 1.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		47

Приложение Б.1

Файл Clock.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections;

[RequireComponent(typeof(Image))]
public class Clock : MonoBehaviour, IPointerClickHandler
{
    [Header("Clock Settings")]
    public Sprite normalClock;
    public Sprite brokenClock;
    public float shakeDuration = 0.3f;
    public float shakeIntensity = 10f;

    private Image clockImage;
    private bool isBroken = false;
    private RectTransform rectTransform;
    private Vector3 originalPosition;

    void Awake()
    {
        clockImage = GetComponent<Image>();
        rectTransform = GetComponent<RectTransform>();
        originalPosition = rectTransform.localPosition;
        clockImage.sprite = normalClock;
    }

    public void OnPointerClick(PointerEventData eventData)
    {
        if (!isBroken)
        {
            clockImage.sprite = brokenClock;
            isBroken = true;
        }
        else
        {
            StartCoroutine(ShakeAndBreakEffect());
        }
    }

    private IEnumerator ShakeAndBreakEffect()
    {
        float elapsed = 0f;
        while (elapsed < shakeDuration)
        {
            Vector2 shakeOffset = Random.insideUnitCircle *
            shakeIntensity;
        }
    }
}
```

```

        rectTransform.localPosition = originalPosition +
(Vector3)shakeOffset;
        elapsed += Time.deltaTime;
        yield return null;
    }
    rectTransform.localPosition = originalPosition;

}
}

```

Приложение Б.2

Файл DuckController.cs

```

using UnityEngine;
using UnityEngine.EventSystems;

[RequireComponent(typeof(SpriteRenderer))]
public class DuckController : MonoBehaviour, IPointerClickHandler
{
    [Header("Settings")]
    public float speed = 500f;
    public bool showDebugLines = true;
    public Color debugLineColor = Color.cyan;

    private DuckHuntGame gameManager;
    private Vector3 targetPosition;
    private bool isDead = false;

    public void Initialize(DuckHuntGame manager, Vector3 target)
    {
        if (manager == null)
        {
            Debug.LogError("GameManager не задан!");
            Destroy(gameObject);
            return;
        }

        gameManager = manager;
        targetPosition = target;

        UpdateRotation();
    }

    void Update()
    {
        if (isDead || gameManager == null) return;

        transform.position = Vector3.MoveTowards(
            transform.position,
            targetPosition,

```

```

        speed * Time.deltaTime
    );

    UpdateRotation();

    if (showDebugLines)
    {
        Debug.DrawLine(transform.position, targetPosition,
debugLineColor);
    }

    if (Vector3.Distance(transform.position, targetPosition)
< 0.1f)
    {
        DestroyDuck();
    }
}

void UpdateRotation()
{
    Vector3 moveDirection = targetPosition - transform.posi-
tion;
    if (moveDirection.x != 0)
    {
        transform.localScale = new Vector3(
            moveDirection.x < 0 ? -1 : 1,
            1,
            1
        );
    }
}

void DestroyDuck()
{
    if (isDead) return;

    isDead = true;
    if (gameManager != null)
        gameManager.DuckKilled(gameObject);
    Destroy(gameObject);
}

public void OnPointerClick(PointerEventData eventData)
{
    if (isDead || gameManager == null) return;

    isDead = true;
    SpriteRenderer renderer = GetComponent<SpriteRenderer>();
    if (renderer != null)
        renderer.color = Color.red;

    if (gameManager != null)

```

```

        gameManager.DuckKilled(gameObject);
        Destroy(gameObject, 0.1f);
    }

    void OnDrawGizmosSelected()
    {
        Gizmos.color = Color.yellow;
        Gizmos.DrawLine(transform.position, targetPosition);
    }
}

```

Приложение Б.3

Файл DuckHuntGame.cs

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections;
using System.Collections.Generic;

public class DuckHuntGame : MonoBehaviour, IPointerClickHandler
{
    [Header("Game Settings")]
    public int ducksToKill = 30;
    public float duckSpawnInterval = 0.5f;

    [Header("UI References")]
    public Text ducksCounterText;
    public GameObject gameOverPanel;
    public Text gameOverText;
    public GameObject arrowObject;
    public InventorySystem inventorySystem;

    [Header("Game Objects")]
    public GameObject gameBackground;
    public GameObject duckPrefab;
    public Transform[] spawnPoints;

    private int ducksKilled = 0;
    private bool gameActive = false;
    private List<GameObject> activeDucks = new List<GameObject>();

    void Start()
    {
        gameBackground.SetActive(false);
        gameOverPanel.SetActive(false);

        if (arrowObject != null) arrowObject.SetActive(false);

        if (spawnPoints.Length != 6)

```

```

        {
            Debug.LogError("Нужно ровно 6 спавн-поинтов!");
        }
    }

    public void OnPointerClick(PointerEventData eventData)
    {
        if (!gameActive)
        {
            StartGame();
        }
    }

    void StartGame()
    {
        gameActive = true;
        ducksKilled = 0;
        activeDucks.Clear();

        gameBackground.SetActive(true);
        gameOverPanel.SetActive(false);

        UpdateUI();
        StartCoroutine(SpawnDucks());
    }

    IEnumerator SpawnDucks()
    {
        while (gameActive && ducksKilled < ducksToKill)
        {
            SpawnDuck();
            yield return new WaitForSeconds(duckSpawnInterval);
        }
    }

    void SpawnDuck()
    {
        if (spawnPoints.Length != 6 || duckPrefab == null) return;

        int startPoint = Random.Range(0, spawnPoints.Length);
        int targetPoint = (startPoint + 3) % 6;

        if (spawnPoints[startPoint] == null || spawnPoints[targetPoint] == null)
        {
            Debug.LogError("Один из спавн-поинтов не задан!");
            return;
        }

        GameObject duck = Instantiate(
            duckPrefab,
            spawnPoints[startPoint].position,

```

```

        Quaternion.identity,
        gameBackground.transform
    );

    DuckController duckController = duck.GetComponent<DuckController>();
    if (duckController != null)
    {
        duckController.Initialize(
            this,
            spawnPoints[targetPoint].position
        );
        activeDucks.Add(duck);
    }
    else
    {
        Destroy(duck);
    }
}

public void DuckKilled(GameObject duck)
{
    if (duck == null || !activeDucks.Contains(duck)) return;

    ducksKilled++;
    activeDucks.Remove(duck);
    Destroy(duck);

    UpdateUI();

    if (ducksKilled >= ducksToKill)
    {
        EndGame();
    }
}

void UpdateUI()
{
    if (ducksCounterText != null)
        ducksCounterText.text = $"{ducksKilled}/{ducksToKill}";
}

void EndGame()
{
    gameActive = false;
    StopAllCoroutines();

    if (gameOverPanel != null)
        gameOverPanel.SetActive(true);
    if (gameOverText != null)
        gameOverText.text = "Ты победил! Все утки убиты!";
}

```

```

        foreach (var duck in activeDucks)
        {
            if (duck != null)
                Destroy(duck);
        }
        activeDucks.Clear();

        StartCoroutine(CloseGame());
    }

    public IEnumerator CloseGame()
    {
        yield return new WaitForSeconds(3f);
        if (gameBackground != null)
            gameBackground.SetActive(false);
        if (gameOverPanel != null)
            gameOverPanel.SetActive(false);

        if (arrowObject != null)
        {
            arrowObject.SetActive(true);

            CanvasGroup arrowCanvas = arrowObject.GetComponent<CanvasGroup>();
            if (arrowCanvas != null)
            {
                arrowCanvas.alpha = 0;
                float fadeTime = 1f;
                float timer = 0;

                while (timer < fadeTime)
                {
                    timer += Time.deltaTime;
                    arrowCanvas.alpha = Mathf.Lerp(0, 1, timer /
fadeTime);
                    yield return null;
                }
            }

            if (inventorySystem != null)
            {
                inventorySystem.SetItemVisible(1, true);
            }
        }
    }
}

```

Приложение Б.4

Файл HeartController.cs

					ПГУ 1.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		54

```

using UnityEngine;

public class HeartController : MonoBehaviour
{
    public float fallSpeed = 1f;

    void Update()
    {
        transform.Translate(Vector3.down * fallSpeed * Time.deltaTime);

        if (transform.position.y < -5f)
        {
            Destroy(gameObject);
        }
    }
}

```

Приложение Б.5

Файл InteractiveDog.cs

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections;

[RequireComponent(typeof(Image))]
public class InteractiveDog : MonoBehaviour, IPointerClickHandler
{
    [Header("Dog Settings")]
    public AudioClip[] barkSounds;
    public Sprite heartSprite;
    public Transform heartSpawnPoint;

    [Header("Heart Settings")]
    public Vector2 heartSize = new Vector2(100, 100);
    public float heartScale = 1f;

    [Header("Timing Settings")]
    public float actionDelay = 3f;
    public float heartDisplayTime = 1f;
    public float fadeDuration = 0.3f;

    [Header("Probabilities")]
    [Range(0, 100)] public int heartChance = 70;

    private AudioSource audioSource;
    private bool canInteract = true;
    private Image dogImage;
}

```

```

void Awake()
{
    dogImage = GetComponent<Image>();
    audioSource = GetComponent<AudioSource>();
    if (audioSource == null)
    {
        audioSource = gameObject.AddComponent<AudioSource>();
    }
}

public void OnPointerClick(PointerEventData eventData)
{
    if (canInteract)
    {
        StartCoroutine(PerformAction());
    }
}

private IEnumerator PerformAction()
{
    canInteract = false;

    bool showHeart = Random.Range(0, 100) < heartChance;

    if (showHeart)
    {
        ShowHeart();
    }
    else
    {
        Bark();
    }

    yield return new WaitForSeconds(actionDelay);
    canInteract = true;
}

private void Bark()
{
    if (barkSounds.Length > 0)
    {
        AudioClip randomBark = barkSounds[Random.Range(0,
barkSounds.Length)];
        audioSource.PlayOneShot(randomBark);
    }
}

private void ShowHeart()
{
    if (heartSprite != null && heartSpawnPoint != null)
    {
        GameObject heartObj = new GameObject("Heart");

```

```

        heartObj.transform.SetParent(transform.parent);
        heartObj.transform.position = heartSpawnPoint.position;
        heartObj.transform.localScale = Vector3.one * heartScale;

        Image heartImage = heartObj.AddComponent<Image>();
        heartImage.sprite = heartSprite;

        RectTransform rt = heartObj.GetComponent<RectTransform>();
        rt.sizeDelta = heartSize;

        CanvasGroup canvasGroup = heartObj.AddComponent<CanvasGroup>();
        StartCoroutine(FadeAndDestroyHeart(heartObj));
    }

    private IEnumerator FadeAndDestroyHeart(GameObject heartObj)
    {
        CanvasGroup canvasGroup = heartObj.GetComponent<CanvasGroup>();
        float timer = 0f;

        yield return new WaitForSeconds(heartDisplayTime - fadeDuration);

        while (timer < fadeDuration)
        {
            timer += Time.deltaTime;
            if (canvasGroup != null)
            {
                canvasGroup.alpha = 1f - (timer / fadeDuration);
            }
            yield return null;
        }

        if (heartObj != null)
        {
            Destroy(heartObj);
        }
    }
}

```

Приложение Б.6

Файл InventoryItem.cs

```

using UnityEngine;
using UnityEngine.EventSystems;

```

					ПГУ 1.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		57

```

using UnityEngine.UI;

public class InventoryItem : MonoBehaviour, IPointerClickHandler
{
    [SerializeField] private Image itemImage;
    private int itemId;
    private InventorySystem inventorySystem;

    public Sprite ItemSprite => itemImage.sprite;

    public void Initialize(int id, InventorySystem system)
    {
        itemId = id;
        inventorySystem = system;
        itemImage = GetComponent<Image>();
        itemImage.raycastTarget = false;
    }

    public void SetInteractable(bool interactable)
    {
        itemImage.raycastTarget = interactable;
    }

    public void OnPointerClick(PointerEventData eventData)
    {
        inventorySystem.OnItemClicked(itemId);
    }
}

```

Приложение Б.7

Файл InventorySystem.cs

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class InventorySystem : MonoBehaviour
{
    public InventoryItem[] inventorySlots = new InventoryItem[3];
    public LockController lockController;

    void Awake()
    {
        InitializeInventoryItems();
    }

    private void InitializeInventoryItems()
    {
        for (int i = 0; i < inventorySlots.Length; i++)
        {
            if (inventorySlots[i] != null)

```

```

        {
            inventorySlots[i].Initialize(i, this);
        }
    }

    public void OnItemClicked(int itemId)
    {
        lockController?.OnInventoryItemClicked(itemId);
    }

    public Sprite GetItemSprite(int itemId)
    {
        return IsValidItemId(itemId) ? inventorySlots[itemId].ItemSprite : null;
    }

    public void SetItemVisible(int itemId, bool visible)
    {
        if (IsValidItemId(itemId))
        {
            inventorySlots[itemId].gameObject.SetActive(visible);
        }
    }

    public void SetItemsInteractable(bool interactable)
    {
        foreach (var slot in inventorySlots)
        {
            if (slot != null) slot.SetInteractable(interactable);
        }
    }

    private bool IsValidItemId(int itemId) => itemId >= 0 && itemId < inventorySlots.Length;
}

```

Приложение Б.8

Файл LanternToggle.cs

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

[RequireComponent(typeof(Image))]
public class LanternToggle : MonoBehaviour, IPointerClickHandler
{
    public Sprite lanternOnSprite;
    public Sprite lanternOffSprite;
}

```

```

private Image lanternImage;
private bool isOn = false;

void Awake()
{
    lanternImage = GetComponent<Image>();
    lanternImage.sprite = lanternOffSprite;
}

public void OnPointerClick(PointerEventData eventData)
{
    ToggleLantern();
}

private void ToggleLantern()
{
    isOn = !isOn;
    lanternImage.sprite = isOn ? lanternOnSprite : lanternOff-
Sprite;
}
}

```

Приложение Б.9

Файл LockController.cs

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;

public class LockController : MonoBehaviour, IPointerClickHandler
{
    [Header("Settings")]
    public int correctCode = 138;
    public float resetDelay = 2f;

    [Header("References")]
    public GameObject lockPanel;
    public Image[] codeSlots = new Image[3];
    public InventorySystem inventorySystem;
    public GameObject zamok;
    public GameObject arrowObject;

    private int[] enteredDigits = new int[3] { -1, -1, -1 };
    private int currentSlotIndex = 0;
    private bool isLockActive = false;

    private Dictionary<int, int> itemIdToDigitMap = new Diction-
ary<int, int>()

```

```

{
    {0, 1},
    {1, 3},
    {2, 8}
};

private void Awake()
{
    if (arrowObject != null) arrowObject.SetActive(false);
    lockPanel.SetActive(false);
    InitializeCodeSlots();
}

private void InitializeCodeSlots()
{
    foreach (var slot in codeSlots)
    {
        if (slot != null)
        {
            slot.gameObject.SetActive(true);
            slot.sprite = null;
            slot.color = new Color(1, 1, 1, 0.5f);
        }
    }
}

public void OnPointerClick(PointerEventData eventData)
{
    if (!isLockActive) OpenLock();
}

private void OpenLock()
{
    isLockActive = true;
    lockPanel.SetActive(true);
    inventorySystem.SetItemsInteractable(true);
    currentSlotIndex = 0;
}

private void CloseLock()
{
    isLockActive = false;
    lockPanel.SetActive(false);
    StartCoroutine(CloseGame());
    zamok.SetActive(false);
    inventorySystem.SetItemsInteractable(false);
}

public IEnumerator CloseGame()
{
    if (arrowObject != null)
    {

```

```

        arrowObject.SetActive(true);

        CanvasGroup arrowCanvas = arrowObject.GetComponent<CanvasGroup>();
        if (arrowCanvas != null)
        {
            arrowCanvas.alpha = 0;
            float fadeTime = 1f;
            float timer = 0;

            while (timer < fadeTime)
            {
                timer += Time.deltaTime;
                arrowCanvas.alpha = Mathf.Lerp(0, 1, timer /
fadeTime);

                yield return null;
            }
        }
    }

    public void OnInventoryItemClicked(int itemId)
    {
        if (!isLockActive || currentSlotIndex >= codeSlots.Length)
return;

        var sprite = inventorySystem.GetItemSprite(itemId);
        if (sprite == null) return;

        codeSlots[currentSlotIndex].gameObject.SetActive(true);
        codeSlots[currentSlotIndex].sprite = sprite;
        codeSlots[currentSlotIndex].color = Color.white;

        enteredDigits[currentSlotIndex] = item-
IdToDigitMap[itemId];
        inventorySystem.SetItemVisible(itemId, false);

        currentSlotIndex++;

        if (currentSlotIndex >= codeSlots.Length)
        {
            CheckCode();
        }
    }

    private void CheckCode()
    {
        int enteredCode = enteredDigits[0] * 100 + enteredDig-
its[1] * 10 + enteredDigits[2];

        if (enteredCode == correctCode)
        {

```

```

        Debug.Log("Code correct!");
        CloseLock();
    }
    else
    {
        Debug.Log("Wrong code!");
        StartCoroutine(ResetLock());
    }
}

private IEnumerator ResetLock()
{
    yield return new WaitForSeconds(resetDelay);

    for (int i = 0; i < enteredDigits.Length; i++)
    {
        if (enteredDigits[i] != -1)
        {
            foreach (var pair in itemIdToDigitMap)
            {
                if (pair.Value == enteredDigits[i])
                {
                    inventorySystem.SetItemVisible(pair.Key,
true);

                    break;
                }
            }
            enteredDigits[i] = -1;
        }
    }

    InitializeCodeSlots();
    currentSlotIndex = 0;
}
}

```

Приложение Б.10

Файл MusicManager.cs

```

using UnityEngine;
using UnityEngine.SceneManagement;

public class MusicManager : MonoBehaviour
{
    public static MusicManager Instance { get; private set; }

    [Header("Music Tracks")]
    public AudioClip mainMenuMusic;
    public AudioClip roomMusic;
    public AudioClip lastRoomMusic;
}

```

```

[Header("Settings")]
[Range(0f, 1f)] public float volume = 0.5f;
public bool loopMusic = true;

private AudioSource audioSource;

void Awake()
{
    if (Instance == null)
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
        audioSource = gameObject.AddComponent<AudioSource>();
        audioSource.volume = volume;
        audioSource.loop = loopMusic;
    }
    else
    {
        Destroy(gameObject);
    }
}

void OnEnable()
{
    SceneManager.sceneLoaded += OnSceneLoaded;
}

void OnDisable()
{
    SceneManager.sceneLoaded -= OnSceneLoaded;
}

private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
{
    switch (scene.name)
    {
        case "StartScene":
            PlayMusic(mainMenuMusic);
            break;
        case "Room1":
            PlayMusic(roomMusic);
            break;
        case "Room2":
            PlayMusic(roomMusic);
            break;
        case "Room3":
            PlayMusic(roomMusic);
            break;
        case "LastRoom":
            PlayMusic(lastRoomMusic);
            break;
    }
}

```

```

        default:
            PlayMusic(mainMenuMusic);
            break;
    }
}

public void PlayMusic(AudioClip clip)
{
    if (clip == null || audioSource.clip == clip && audioSource.isPlaying)
        return;

    audioSource.clip = clip;
    audioSource.Play();
}

public void StopMusic()
{
    audioSource.Stop();
}

public void SetVolume(float newVolume)
{
    volume = Mathf.Clamp01(newVolume);
    audioSource.volume = volume;
}

public void ToggleMute()
{
    audioSource.mute = !audioSource.mute;
}
}

```

Приложение Б.11

Файл NextSceneArrow.cs

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using UnityEngine.EventSystems;
using System.Collections;

public class NextSceneArrow : MonoBehaviour, IPointerClickHandler
{
    [Header("Settings")]
    public string nextSceneName;

    private bool isTransitioning = false;

    public void OnPointerClick(PointerEventData eventData)

```

					ПГУ 1.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		65

```

        {
            if (!isTransitioning && !string.IsNullOrEmpty(nextScene-
Name))
            {
                TransitionToNextScene();
            }
        }

private void TransitionToNextScene()
{
    isTransitioning = true;

    SceneManager.LoadScene(nextSceneName);
}
}

```

Приложение Б.12

Файл RabbitController.cs

```

using System.Collections;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;

public class RabbitController : MonoBehaviour, IPointerClickHand-
ler
{
    [Header("Settings")]
    public Image heartPrefab;
    public float spawnHeight = 100f;
    public float fallSpeed = 50f;
    public int heartsPerClick = 2;
    public float spawnInterval = 0.3f;
    public float heartLifetime = 2f;
    public float fadeDuration = 0.5f;

    private Canvas parentCanvas;
    private RectTransform rabbitRect;

    void Start()
    {
        parentCanvas = GetComponentInParent<Canvas>();
        if (parentCanvas == null)
            Debug.LogError("No Canvas found in parents!");

        rabbitRect = GetComponent<RectTransform>();
    }

    public void OnPointerClick(PointerEventData eventData)
    {

```

```

        StartCoroutine(SpawnHearts());
    }

IEnumerator SpawnHearts()
{
    for (int i = 0; i < heartsPerClick; i++)
    {
        Image newHeart = Instantiate(heartPrefab, parentCanvas.transform);
        newHeart.color = Color.white;

        RectTransform heartRect = newHeart.GetComponent<RectTransform>();

        heartRect.anchoredPosition = GetHeartStartPosition(rabbitRect);

        heartRect.anchoredPosition += new Vector2(
            Random.Range(-100f, 100f),
            Random.Range(150f, 170f));

        StartCoroutine(AnimateHeart(newHeart));
        yield return new WaitForSeconds(spawnInterval);
    }
}

Vector2 GetHeartStartPosition(RectTransform rabbitRect)
{
    return new Vector2(
        rabbitRect.anchoredPosition.x,
        rabbitRect.anchoredPosition.y + spawnHeight);
}

IEnumerator AnimateHeart(Image heart)
{
    RectTransform rectTransform = heart.GetComponent<RectTransform>();
    CanvasGroup canvasGroup = heart.gameObject.AddComponent<CanvasGroup>();
    float timer = 0f;

    rectTransform.localScale = Vector3.zero;

    while (timer < 0.2f)
    {
        timer += Time.deltaTime;
        rectTransform.localScale = Vector3.one * (timer / 0.2f);

        yield return null;
    }

    timer = 0f;
}

```

```

        Vector2 startPosition = rectTransform.anchoredPosition;

        while (timer < heartLifetime)
        {
            timer += Time.deltaTime;

            rectTransform.anchoredPosition = startPosition +
                Vector2.down * fallSpeed * timer;

            if (timer > heartLifetime - fadeDuration)
            {
                float fadeProgress = (timer - (heartLifetime -
                    fadeDuration)) / fadeDuration;
                canvasGroup.alpha = 1f - fadeProgress;
            }

            yield return null;
        }

        Destroy(heart.gameObject);
    }

    private void Reset()
    {
        if (GetComponent<Collider2D>() == null)
            gameObject.AddComponent<BoxCollider2D>();
    }
}

```

Приложение Б.13

Файл RiddleSystem.cs

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections;

public class RiddleSystem : MonoBehaviour, IPointerClickHandler
{
    [Header("UI References")]
    public Image riddleBackground;
    public Text riddleText;
    public Button[] answerButtons = new Button[3];
    public Text[] answerTexts = new Text[3];
    public GameObject riddlePanel;
    public GameObject arrowObject;
    public float arrowShowDelay = 1f;
    public InventorySystem inventorySystem;

    [Header("Settings")]
}

```

```

public float fadeDuration = 0.5f;
public Color backgroundColor = new Color(0, 0, 0, 0.8f);

private string[,] riddles = new string[,]
{
    {"Что можно увидеть с закрытыми глазами?", "Сон", "Тьму",
    "Мечты", "0"},
    {"Что становится влажным, пока сохнет?", "Полотенце",
    "Зонт", "Мыло", "0"},
    {"Что идет вверх и вниз, но не двигается?", "Лестница",
    "Температура", "Лифт", "0"},
    {"Чем больше берешь, тем больше оставляешь. Что это?",
    "Следы", "Долги", "Фотографии", "0"},
    {"Что принадлежит вам, но другие используют это чаще?",
    "Имя", "Телефон", "Одежда", "0"}
};

private int currentRiddleIndex;
private bool isRiddleActive = false;

void Start()
{
    if (riddlePanel != null)
    {
        riddlePanel.SetActive(false);
    }

    InitializeAnswerButtons();
}

private void InitializeAnswerButtons()
{
    for (int i = 0; i < answerButtons.Length; i++)
    {
        int index = i;
        answerButtons[i].onClick.AddListener(() =>
OnAnswerSelected(index));
    }
}

public void OnPointerClick(PointerEventData eventData)
{
    if (!isRiddleActive)
    {
        ShowRandomRiddle();
    }
}

private void ShowRandomRiddle()
{
    currentRiddleIndex = Random.Range(0, riddles.GetLength(0));
}

```

```

        ShowRiddle(currentRiddleIndex);
    }

    private void ShowRiddle(int riddleIndex)
    {
        isRiddleActive = true;
        riddlePanel.SetActive(true);

        riddleText.text = riddles[riddleIndex, 0];

        int answersCount = Mathf.Min(answerTexts.Length, riddles.GetLength(1) - 2);
        for (int i = 0; i < answersCount; i++)
        {
            answerTexts[i].text = riddles[riddleIndex, i + 1];
            answerButtons[i].gameObject.SetActive(true);
        }

        for (int i = answersCount; i < answerButtons.Length; i++)
        {
            answerButtons[i].gameObject.SetActive(false);
        }

        StartCoroutine(FadeIn());
    }

    private void OnAnswerSelected(int answerIndex)
    {
        if (!isRiddleActive) return;

        int correctAnswer = int.Parse(riddles[currentRiddleIndex, 4]);
        bool isCorrect = (answerIndex == correctAnswer);

        Debug.Log(isCorrect ? "Правильно!" : "Неправильно!");

        if (isCorrect)
        {
            StartCoroutine(HideRiddleWithBackground());
        }
        else
        {
            StartCoroutine(FadeOut());
        }
    }

    private IEnumerator FadeIn()
    {
        if (riddleBackground != null)
        {
            riddleBackground.color = Color.clear;
            float elapsed = 0;

```

```

        while (elapsed < fadeDuration)
        {
            elapsed += Time.deltaTime;
            riddleBackground.color = Color.Lerp(Color.clear,
backgroundcolor, elapsed / fadeDuration);
            yield return null;
        }
    }

private IEnumerator FadeOut()
{
    if (riddleBackground != null)
    {
        Color startColor = riddleBackground.color;
        float elapsed = 0;

        while (elapsed < fadeDuration)
        {
            elapsed += Time.deltaTime;
            riddleBackground.color = Color.Lerp(startColor,
Color.clear, elapsed / fadeDuration);
            yield return null;
        }

        HideRiddle();
    }

private IEnumerator HideRiddleWithBackground()
{
    if (riddleBackground != null)
    {
        riddleBackground.gameObject.SetActive(false);
    }

    yield return new WaitForSeconds(0.1f);

    HideRiddle();
}

private void HideRiddle()
{
    isRiddleActive = false;
    riddlePanel.SetActive(false);
    if (arrowObject != null)
    {
        StartCoroutine(ShowArrowAfterDelay());
    }

    if (inventorySystem != null)

```

```

        {
            inventorySystem.SetItemVisible(2, true);
        }
    }

IEnumerator ShowArrowAfterDelay()
{
    yield return new WaitForSeconds (arrowShowDelay);
    arrowObject.SetActive(true);
}
}

```

Приложение Б.14

Файл RobotDialog.cs

```

using UnityEngine;
using UnityEngine.UI;
using TMPPro;
using UnityEngine.EventSystems;
using System.Collections;

public class RobotDialog : MonoBehaviour, IPointerClickHandler
{
    private string[] dialogPhrases = {
        "Приветствую, игрок!",
        "Я - страж этой комнаты...",
        "Чтобы пройти дальше, тебе нужно...",
        "Победить меня в игре!",
        "Готов начать?"
    };

    private string[] winPhrases = {
        "Не может быть!",
        "Ты победил меня...",
        "Вот твоя награда...",
        "Цифра для кода: ",
        "Проход свободен."
    };

    public float textDisplaySpeed = 0.05f;
    public int codeDigit = 1;

    [Header("UI References")]
    public GameObject dialogPanel;
    public Text dialogText;
    public GameObject arrowObject;
    public float arrowShowDelay = 1f;

    [Header("Game References")]
    public GameObject gameBackground;
}

```

```

public TicTacToeGame ticTacToeGame;
public InventorySystem inventorySystem;

private int currentPhraseIndex = 0;
private bool isDialogActive = false;
private bool isTextAnimating = false;
private Coroutine textAnimation;
private bool gameStarted = false;
private bool gameCompleted = false;

private void Awake()
{
    dialogPanel.SetActive(false);
    gameBackground.SetActive(false);
    gameCompleted = false;
    gameStarted = false;
}

public void OnPointerClick(PointerEventData eventData)
{
    if (gameCompleted) return;

    if (!isDialogActive && !gameStarted)
    {
        StartDialog();
    }
    else if (!isTextAnimating)
    {
        ShowNextPhrase();
    }
    else
    {
        SkipTextAnimation();
    }
}

void StartDialog()
{
    if (dialogPhrases.Length == 0) return;

    currentPhraseIndex = 0;
    isDialogActive = true;
    dialogPanel.SetActive(true);
    DisplayCurrentPhrase();
}

public void ShowWinDialog()
{
    currentPhraseIndex = 0;
    isDialogActive = true;
    dialogPanel.SetActive(true);
}

```

```

        winPhrases[3] = "Цифра для кода: " + codeDigit;

        dialogPhrases = winPhrases;
        DisplayCurrentPhrase();
    }

    void DisplayCurrentPhrase()
    {
        if (textAnimation != null)
            StopCoroutine(textAnimation);

        textAnimation = StartCoroutine(AnimateText(dialogPhrases[currentPhraseIndex]));
    }

    IEnumerator AnimateText(string text)
    {
        isTextAnimating = true;
        dialogText.text = "";

        foreach (char letter in text.ToCharArray())
        {
            dialogText.text += letter;
            yield return new WaitForSeconds(textDisplaySpeed);
        }

        isTextAnimating = false;
    }

    void SkipTextAnimation()
    {
        StopCoroutine(textAnimation);
        dialogText.text = dialogPhrases[currentPhraseIndex];
        isTextAnimating = false;
    }

    void ShowNextPhrase()
    {
        currentPhraseIndex++;

        if (currentPhraseIndex < dialogPhrases.Length)
        {
            DisplayCurrentPhrase();
        }
        else
        {
            EndDialog();
        }
    }

    void EndDialog()
    {

```

```

        dialogPanel.SetActive(false);
        isDialogActive = false;

        if (dialogPhrases == winPhrases)
        {
            gameCompleted = true;
            if (inventorySystem != null)
            {
                inventorySystem.SetItemVisible(0, true);
            }

            if (arrowObject != null)
            {
                StartCoroutine(ShowArrowAfterDelay());
            }
        }
        else
        {
            gameStarted = true;
            StartCoroutine(StartGameAfterDialog());
        }
    }

    IEnumerator StartGameAfterDialog()
    {
        yield return new WaitForSeconds(0.1f);
        gameBackground.SetActive(true);
        ticTacToeGame.StartGame();
    }

    IEnumerator ShowArrowAfterDelay()
    {
        yield return new WaitForSeconds(arrowShowDelay);
        arrowObject.SetActive(true);
    }

    private void Reset()
    {
        if (GetComponent<Collider2D>() == null)
        {
            gameObject.AddComponent<BoxCollider2D>();
        }
    }
}

```

Приложение Б.15

Файл Starter.cs

```

using UnityEngine;
using UnityEngine.SceneManagement;

```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.37.001 ПЗ

Лист

75

```

public class Starter : MonoBehaviour
{
    public void StartGame()
    {
        SceneManager.LoadScene("Room1");
    }

    public void QuitGame()
    {
#ifdef UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
        Application.Quit();
#endif
    }
}

```

Приложение Б.16

Файл TicTacToeGame.cs

```

using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class TicTacToeGame : MonoBehaviour
{
    [Header("Game Elements")]
    public Image[] cellImages;
    public Sprite xSprite;
    public Sprite oSprite;
    public GameObject gameBackground;
    public Text gameStatusText;

    [Header("Game Settings")]
    public float robotDelay = 1f;

    [Header("Game Settings")]
    public RobotDialog robotDialog;

    private int[] board = new int[9];
    private bool isPlayerTurn = true;
    private bool gameActive = false;

    void Start()
    {
        gameBackground.SetActive(false);
        if (gameStatusText != null) gameStatusText.gameObject.SetActive(false);
    }
}

```

```

public void StartGame()
{
    gameBackground.SetActive(true);
    if (gameStatusText != null)
    {
        gameStatusText.gameObject.SetActive(true);
        gameStatusText.text = "Ваш ход (X)";
    }
    gameActive = true;
    ResetBoard();
}

public void OnCellClicked(int cellIndex)
{
    if (!gameActive || board[cellIndex] != 0 || !isPlayerTurn)
        return;

    MakeMove(cellIndex, 1, xSprite);

    if (CheckWin(1))
    {
        EndGame("Вы победили!");
        return;
    }

    if (IsBoardFull())
    {
        EndGame("Ничья!");
        return;
    }

    isPlayerTurn = false;
    if (gameStatusText != null) gameStatusText.text = "Ход
робота...";
    StartCoroutine(RobotMove());
}

IEnumerator RobotMove()
{
    yield return new WaitForSeconds(robotDelay);

    int randomCell;
    do
    {
        randomCell = Random.Range(0, 9);
    } while (board[randomCell] != 0);

    MakeMove(randomCell, 2, oSprite);

    if (CheckWin(2))
    {

```

```

        EndGame("Робот победил!");
        yield break;
    }

    if (IsBoardFull())
    {
        EndGame("Ничья!");
        yield break;
    }

    isPlayerTurn = true;
    if (gameStatusText != null) gameStatusText.text = "Ваш ход
(X) ";
}

void MakeMove(int cellIndex, int player, Sprite sprite)
{
    board[cellIndex] = player;
    cellImages[cellIndex].sprite = sprite;
    cellImages[cellIndex].gameObject.SetActive(true);
    cellImages[cellIndex].color = Color.white;
}

bool CheckWin(int player)
{
    int[,] winConditions = new int[8, 3] {
        {0, 1, 2}, {3, 4, 5}, {6, 7, 8},
        {0, 3, 6}, {1, 4, 7}, {2, 5, 8},
        {0, 4, 8}, {2, 4, 6}
    };

    for (int i = 0; i < 8; i++)
    {
        if (board[winConditions[i, 0]] == player &&
            board[winConditions[i, 1]] == player &&
            board[winConditions[i, 2]] == player)
        {
            return true;
        }
    }
    return false;
}

bool IsBoardFull()
{
    foreach (int cell in board)
    {
        if (cell == 0) return false;
    }
    return true;
}

```

```

void ResetBoard()
{
    board = new int[9];
    foreach (Image img in cellImages)
    {
        img.gameObject.SetActive(false);
    }
    isPlayerTurn = true;
    gameActive = true;
}

void EndGame(string message)
{
    gameActive = false;
    if (gameStatusText != null) gameStatusText.text = message;

    if (message.Contains("Вы победили"))
    {
        Invoke("ShowWinDialog", 1f);
    }
    else
    {
        Invoke("RestartGame", 3f);
    }
}

void ShowWinDialog()
{
    CloseGame();
    robotDialog.ShowWinDialog();
}

void RestartGame()
{
    ResetBoard();
    if (gameStatusText != null) gameStatusText.text = "Ваш ход
(X) ";
}

public void CloseGame()
{
    gameBackground.SetActive(false);
    if (gameStatusText != null) gameStatusText.gameObject.SetActive(false);
    gameActive = false;
}
}

```

Приложение В

(обязательное)

Мобильное приложение в жанре квест на Unity

Презентация

					ПГУ 1.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		80

БАКАЛАВРСКАЯ РАБОТА

на тему

**МОБИЛЬНОЕ ПРИЛОЖЕНИЕ В
ЖАНРЕ КВЕСТ НА UNITY**

Выполнил студент гр. 21ВВП2: Михальцова П.В.

Цели и задачи

Предметная область охватывает создание мобильного игрового приложения, разработанного для платформы Android с использованием Unity. Основная цель игры — развлечение и интеллектуальное развитие игрока через решение логических задач. Приложение должно предоставить пользователю качественный квестовый опыт с упором на решение головоломок и удобное управление, оставаясь стабильным на большинстве мобильных устройств.

Функциональные требования к приложению:

- многоуровневая структура сложности;
- механика решения головоломок различного типа;
- реализация интерактивных объектов, с которыми игрок может взаимодействовать.

Нефункциональные требования:

- минималистичный, но выразительный визуальный стиль;
- атмосферный звуковой дизайн.

Выбранные технологии разработки приложения

Игровая платформа:

Unity сочетает в себе простоту освоения, все необходимые инструменты и хорошую производительность на Android.



Графический редактор:

Adobe Photoshop подходит для создания и обработки текстур и визуальных элементов. Имеет удобный экспорт и гибкий набор инструментов.



Музыкальный редактор:

BandLab предоставляет необходимые инструменты для создания качественной музыкальной составляющей.



Язык программирования:

C# в Microsoft Visual Studio 2022. Он демонстрирует полную функциональную достаточность для решения всех поставленных задач разработки.



Игровая система

Управление игровым сеансом. Инициализация нового игрового процесса. Завершение текущего процесса.

Навигация по пространству. Последовательное перемещение между локациями (комнатами).

Решение игровых задач. Выполнение логических головоломок. Взаимодействие с интерактивными элементами окружения.

Завершение игрового цикла. Рестарт игрового процесса.

Окончательное завершение игровой сессии.

Данный функционал реализует базовый игровой цикл, обеспечивая последовательное прохождение контента с возможностью повторного прохождения или завершения игры по желанию пользователя.



Большое спасибо за внимание!

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.37.001 ПЗ

Лист

82