МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка AR – игры «Phantasmagoria»

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп 20-1  Якименко Илья Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2023

Аннотация

Курсовой проект на тему: Разработка AR – игры «Phantasmagoria».

Курсовой проект состоит из введения, трех глав, заключения, списка использованной литературы и приложений.

Во введении обоснована актуальность темы, цели и задачи.

В первой главе «Описание предметной области» рассмотрены аналоги разрабатываемого приложения, техническое задание для этого приложения на основе этих аналогов и описание структуры разрабатываемого приложения.

Вторая глава «Описание разработки приложения» посвящена самой разработке приложения, состоящей из обоснования средств разработки, разработки интерфейса, а также изображения логики работы приложения с схемой взаимодействия компонентов проекта.

Третья глава «Тестирование и установка приложения» содержит информацию о методе тестирования приложения, его результат и установку приложения.

В заключении приведены основные выводы, полученные в результате работы курсового проекта.

Список использованной литературы насчитывает 15 ссылок на источники.

В приложениях содержатся листинги рабочего кода и руководство пользователя.

Содержит 29 таблиц, 26 рисунков.

Ссылка на GitHub - https://github.com/YakimenkoIlya/-Phantasmagoria

Содержание

[Введение 4](#_Toc134485429)

[1 Описание предметной области 5](#_Toc134485430)

[1.1 Аналоги разрабатываемого приложения 5](#_Toc134485431)

[1.1.1 Pokemon Go 5](#_Toc134485432)

[1.1.2 The Walking Dead: Our World 6](#_Toc134485433)

[1.1.3 Night Terrors: Bloody Mary 8](#_Toc134485434)

[1.2 Техническое задание 10](#_Toc134485435)

[1.2.1 Краткая характеристика области применения 10](#_Toc134485436)

[1.2.2 Основания для разработки программного изделия 10](#_Toc134485437)

[1.2.3 Функциональное и эксплуатационное назначение 10](#_Toc134485438)

[1.2.4 Требования к программе 11](#_Toc134485439)

[1.3 Описание структуры приложения 12](#_Toc134485440)

[2 Описание разработки приложения 13](#_Toc134485441)

[2.1 Обоснование средств разработки 13](#_Toc134485442)

[2.1.1 Android Studio 13](#_Toc134485443)

[2.1.2 Unreal Engine 5 13](#_Toc134485444)

[2.1.3 Unity 14](#_Toc134485445)

[2.2 Разработка интерфейса 15](#_Toc134485446)

[2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта 20](#_Toc134485447)

[2.4 Описание переменных, компонентов, классов и подпрограмм 21](#_Toc134485448)

[3 Тестирование и установка приложения 34](#_Toc134485449)

[Заключение 40](#_Toc134485450)

[Список использованной литературы 41](#_Toc134485451)

[Приложение А. Листинг кода 43](#_Toc134485452)

[Приложение Б. Руководство пользователя 61](#_Toc134485453)

# Введение

Дополненная реальность (AR) может быть эффективно задействована в проектах различной направленности и предназначения. Одним из основных сфер применения данной технологии является производство мобильных игр.

Несмотря на большой потенциал, относительную новизну и сильное производимое впечатление на пользователя, дополненная реальность является относительно редко применимой технологией в индустрии мобильных игр.

Игра «Pokemon Go», вышедшая в 2016, произвела настоящий культурный феномен, однако не смогла стать родоначальником для появления большого количества игр с применением дополненной реальности по известным франшизам от крупных студий, за редким исключением. Причиной тому может являться то, что игровые компании посчитали главным фактором популярности приложения именно взятую за основу вселенную, отлично подходящую для формата игры, а не технологию и предложение нового опыта для игроков.

Данную ситуацию можно исправить, продемонстрировав игрокам и разработчикам пример приложения, которое сможет удовлетворить запросы и потребности первых, а также создание и поддержка которого являются выполнимыми и финансового выгодными для вторых.

Целью курсовой работы является создание мобильного приложения (игры) «Phantasmagoria», предназначенного для развлечения пользователей.

В соответствии с целью курсовой работы были поставлены следующие задачи исследования:

1. Изучение игрового движка Unity, а также средств создания дополненной реальности в данной программе.
2. Создание концепта и структуры будущего приложения
3. Разработка мобильного приложения

# 1 Описание предметной области

## Аналоги разрабатываемого приложения

1.1.1 Pokemon Go



Рисунок 1 – Постер игры “Pokemon Go”



Рисунок 2 – Отображение интерфейса приложения во время игровой сессии (ловли покемонов)



Рисунок 3 – Интерфейс приложения во время игровой сессии (сражения)

Pokemon GO – многопользовательская ролевая мобильная игра по мотивам известной франшизы «Pokemon» (Покемон) на основе определения местоположения дополненной реальности. [4]

Достоинства приложения:

1. Взятие за основу популярной франшизы, помогающее игре приобрести большое количество пользователей
2. Низкий порог вхождения для игроков, несложный геймплей

Недостатки приложения:

1. Относительно однообразный геймплей (игровой процесс), сводящийся к нахождению и ловле существ (покемонов), находящихся в разных точках реального мира и проведения сражений между ними (битвы происходят без использования дополненной реальности).
2. Блокировка приложения в России.

1.1.2 The Walking Dead: Our World



Рисунок 4 – Постер игры “The Walking Dead: Our World”



Рисунок 5 – Интерфейс приложения во время игровой сессии (сражения)

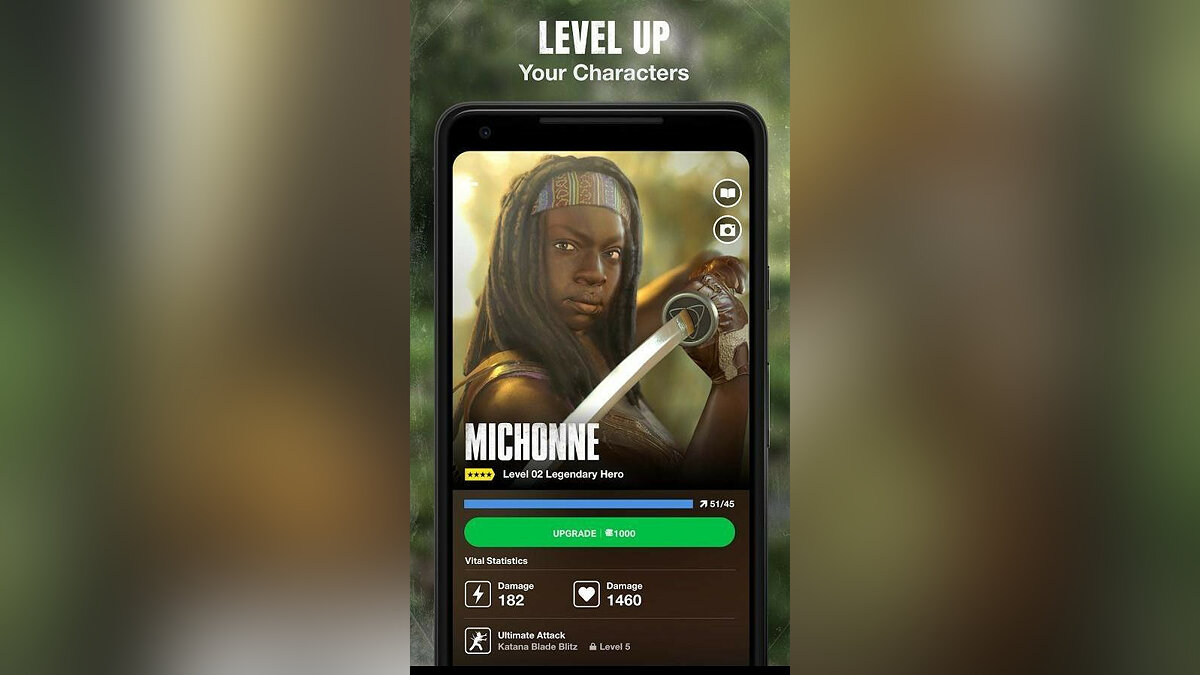


Рисунок 6 – Интерфейс приложения при усилении персонажа

The Walking Dead: Our World - многопользовательская ролевая мобильная игра по мотивам известной франшизы «The Walking Dead» (Ходячие мертвецы) на основе определения местоположения дополненной реальности. [5]

Достоинства приложения:

1. Взятие за основу популярной франшизы, помогающее игре приобрести большое количество пользователей
2. Разноплановый игровой процесс, включающий в себя сражение с мертвецами, союзных персонажей (выживших) до точки с убежищем, строительство виртуальной инфраструктуры, взаимодействие с постройками других игроков, открытие и усиление персонажей, оружия и т.д.

Недостатки приложения:

1. Относительно редкое применение дополненной реальности при подаче контента: сражение с противниками как правило осуществляется на заранее смоделированной игровой сцене.
2. Скучная реализация боев: выстрелы производятся нажатием по противнику.

1.1.3 Night Terrors: Bloody Mary



Рисунок 7 – Постер игры “Night Terrors: Bloody Mary”



Рисунок 8 – Интерфейс приложения во время игровой сессии

Night Terrors: Bloody Mary – хоррор - игра с дополненной реальностью. [6]

Достоинства приложения:

Действительно пугающие сцены (скримеры), что важно для хоррор – игры.

Недостатки приложения:

Практически полное отсутствие геймплея – весь контент заключается в периодическом появлении тех или иных пугающих сцен, с которыми невозможно взаимодействовать.

## Техническое задание

1.2.1 Краткая характеристика области применения

Программа предназначена для массового пользователя и является приложением, предназначенным для отдыха и развлечения.

Приложение устанавливается через USB-кабель или Wi-Fi – соединение.

1.2.2 Основания для разработки программного изделия

Основанием для разработки данного программного продукта является курсовое проектирование.

* + 1. Функциональное и эксплуатационное назначение

Приложение должно содержать следующий функционал:

При запуске приложения будет появляться экран с интерфейсом, содержащим следующие пункты:

1. Название приложения
2. Кнопка перехода к игровой сессии
3. Кнопка вывода туториала, объясняющего правила игры

При нажатии на кнопку перехода к игровой сессии должна загружаться сцена, в которой игроку предстоит найти игровой объект, после сообщить об этом, нажав кнопку.

После нажатия кнопки будет происходить переход к игровой сцене, в которой в сторону игрока двигаются виртуальные противники с постоянным ускорением, а фоном выступает изображение с камеры телефона.

Игроку будет нужно уничтожать таких врагов путем наведения на них прицела и нажатием на экран.

При уничтожении врагов игроку будут начисляться очки, количество которых должно увеличиваться соразмерно расстоянию между игроком и побежденным противником в сцене.

При приближении противника на определенное расстояние к игроку, у него должно убавляться здоровье, пока этот враг не будет побежден.

Помимо противников, в сцене также должны появляться «Кубы исцеления», которые будут уничтожаться подобно врагам и выдавать дополнительные очки, а также восстанавливать здоровье при попадании по ним.

Если количество здоровья игрока падает до нуля, должно появиться всплывающее окно об окончании игры, где игроку предложат начать игру заново или выйти в главное меню нажатием соответствующих кнопок.

Количество здоровья и очков будет отображаться на экране. Также будет присутствовать кнопка паузы, которая останавливает игру и позволяет перезагрузить или продолжить игровую сессию, а также выйти в меню, в зависимости от нажатой кнопки на своем всплывающем окне.

При нажатии на кнопку вывода туториала в главном меню будет выводиться экран с объяснением правил игры, из которого можно будет выйти, нажав соответствующую кнопку.

* + 1. Требования к программе

1. Корректный запуск на тестируемом устройстве (Mi 9 Lite).
2. Исправная работа приложения во время игровых сессий.
   1. Своевременный и корректный отклик на действия игрока.
   2. Правильное отображение графики и фона.
   3. Корректная работа индикаторов (здоровья и очков).
3. Соответствие между нажатыми кнопками и ожидаемыми от них действиями.

1.3 Описание структуры приложения

Приложение состоит из трех сцен:

1. StartScene

Сцена, содержащая начальный интерфейс для перехода в сцену перед основной игрой «PrelevelScene», а также для вывода туториала.

1. PrelevelScene

Сцена, содержащая модельку игрового объекта, которого надо найти игроку и оповестить об этом, нажав кнопку внизу экрана, чтобы перейти в основную игровую сцену «LevelScene».

1. LevelScene

Основная игровая сцена, которая содержит основной контент игры. При нажатии на кнопку паузы, игрок мжет начать игру заново, продолжить текущую игровую сессию или выйти в сцену меню «StartScene».

2 Описание разработки приложения

2.1 Обоснование средств разработки

Возможные варианты среды для разработки:

2.1.1 Android Studio

Android Studio — интегрированная среда разработки (IDE) для работы с платформой Android, являющаяся официальным средством разработки для данной платформы. Используемые языки программирования: Java и Kotlin. [2]

Достоинства среды:

1. Удобный редактор кода
2. Отсутствие необходимости настраивать проект под операционную систему (ОС) Android

Недостатки среды:

1. Отсутствие возможности выстраивания игровых сцен с 3D –элементами.
2. Отсутствие средств для создания игровых объектов (таких как размещение модельки персонажа, удобное присвоение ему скриптов, аудиофайлов и т.д.).
3. Высокая требовательность IDE к производительности аппаратной основы ПК

Android Studio хорошо подходит для создания приложений для Android, но не имеет нормальных средств для разработки игр с 3D сценами.

### 2.1.2 Unreal Engine 5

Unreal Engine — один из популярнейших свободно распространяемых игровых движков, используемый для создания игр различных жанров под большинство популярных ОС. [3]

Достоинства движка:

1. Наличие технологии визуального программирования Blueprints, способного сократить большое количество времени на создании и настройке персонажа.
2. Удобные средства для создания игровых средств и создания спецэффектов высокого качества

Недостатки движка:

1. Высокая сложность интерфейса движка
2. Высокая требовательность движка к производительности аппаратной основы ПК

Unreal Engine 5 хорошо подходит для создания игр различных жанров под большое количество платформ, однако он имеет высокую требовательность к ресурсам ПК, высокую сложность и, как правило, не применяется для создания игр с дополненной реальностью.

* + 1. Unity

Unity - кроссплатформенная среда разработки компьютерных игр различных жанров, являющаяся одним из популярнейших игровых движков. [1]

Достоинства движка:

1. Большое количество игровых ассетов - компонентов, представляющих готовое решение, которое можно использовать в проекте (текстуры, материалы, модельки и т.д.), многие из которых доступны бесплатно. [7]
2. Удобный и простой интерфейс [12]
3. Относительно низкие требования к производительности аппаратной основы ПК
4. Хорошо подходит для создания приложений с дополненной реальностью

Недостатки движка:

1. Созданные проекты занимают относительно много памяти
2. При создании больших проектов высока вероятность понижения производительности игры

Игровой движок Unity хорошо подходит для создания игр, особенно для тех из них, что создаются небольшими командами и не имеют слишком сложную архитектуру.

На основании данного анализа качестве средства разработки был выбран игровой движок Unity версии 2021.3.6f1, поскольку он разработан специально для игр и имеет удобные средства для создания игровых сцен, размещения графики и т.д. [8]

Для создания дополненной реальности в Unity существует специальная среда AR Foundation, позволяющая использовать AR – технологии в приложения для Android и IOS. Однако в процессе ее изучения было принято решения не использовать данную среду, так как она совместима лишь с относительно небольшим количеством устройств обеих платформ. [9]

Плагины для создания дополненной реальности, такие Vuforia, EasyAR, ARCore и аналогичные им также не используются по вышеуказанной причине. [10][11]

Описание используемого языка:

C# — объектно-ориентированный язык программирования общего назначения.

Относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Применяется в различных сферах, таких как создание бизнес-приложений, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android, в том числе видеоигр.

В Unity функциональная часть проектов также разрабатывается при помощи данного языка, дополняя его большим количеством собственных функций, классов и интерфейсов.

2.2 Разработка интерфейса

UI приложения состоит из следующих интерфейсов:

1. Главное меню.
2. Туториал.
3. Сцена перед основной игрой.
4. Игровая сцена.
5. Всплывающее окно паузы.
6. Всплывающее окно окончания игры.

В качестве программы для создания UI приложения используется Adobe XD. [13]

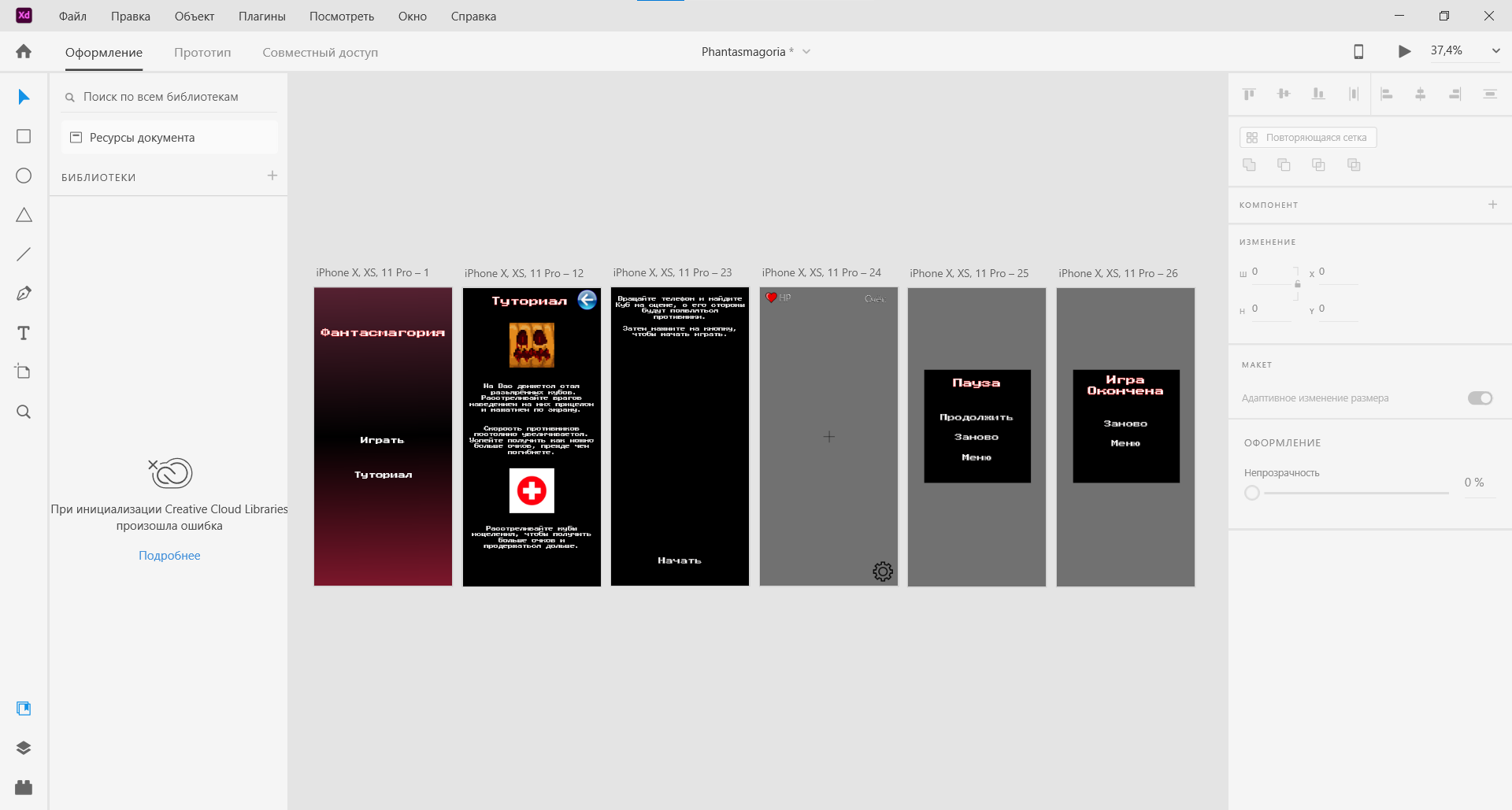


Рисунок 9 – Отображение интерфейса приложения в Adobe XD

1. Главное меню



Рисунок 10 – Интерфейс меню

Содержит кнопку «Играть» перехода на сцену перед основной игрой, а также кнопку «Туториал» для вывода информационного окна.

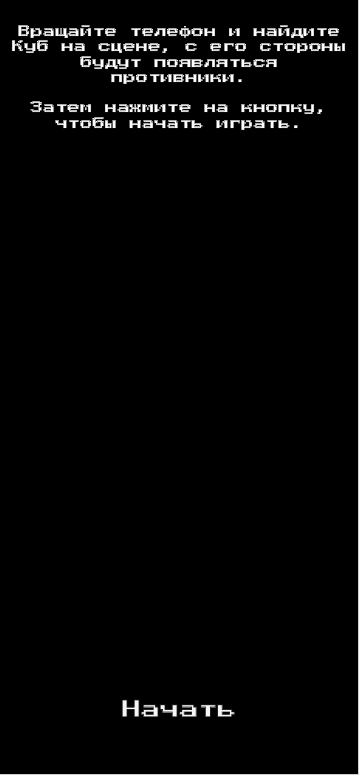
2. Туториал



Рисунок 11 – Интерфейс туториала

Содержит разъяснение основных механик и действий, что необходимо совершать игроку. В левом верхнем углу находится кнопка возвращения в меню.

3. Сцена перед основной игрой



\*\*\*\*\*\*

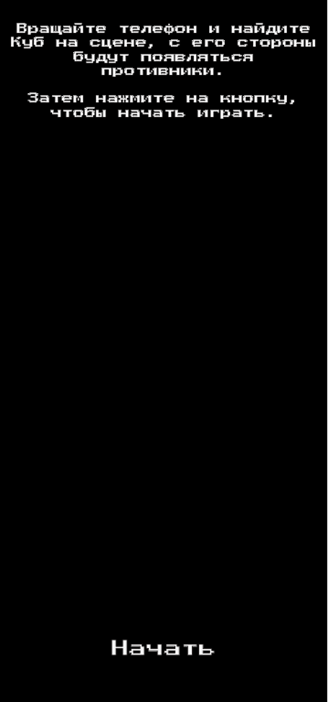


Рисунок 12 – Интерфейс сцены перед основной игрой

Содержит описание действий для игрока и кнопку «Начать» для перехода в игровую сцену.

4. Игровая сцена

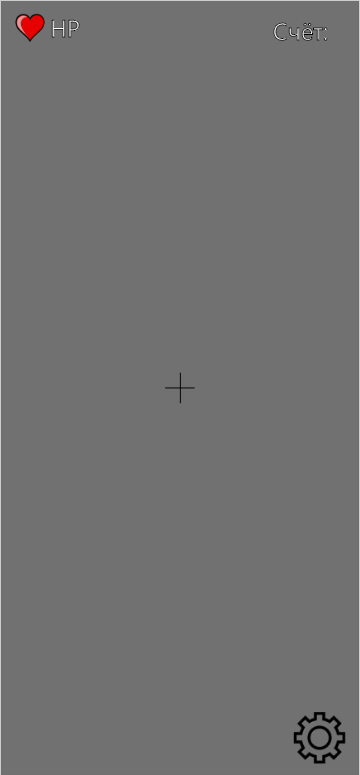


Рисунок 13 –Интерфейс игровой сцены

Содержит индикатор оставшегося здоровья в правом верхнем углу и индикатор набранных очков в левом верхнем углу, а также кнопку паузы правом нижнем углу экрана для, при нажатии по которой выводится всплывающее окно паузы. В центре экрана расположен прицел, который увеличивается и становится красным при наведении на игровой объект.

Фоном для данного экрана и его всплывающих окон служит изображение, получаемое с камеры телефона, а все игровые модели появляются над этим фоном.

5. Вплывающее окно паузы



Рисунок 14 – Интерфейс всплывающего окна паузы

Данное всплывающее окно содержит кнопку «Продолжить» для продолжения игры, кнопку «Заново» для перезагрузки игровой сцены, а также кнопку «Меню» для выхода на начальный экран приложения. При появлении окна паузы, прежний UI экрана, за исключением прицела, сохраняется.

6. Всплывающее окно окончания игры



Рисунок 15 – Интерфейс окна окончания игры

Данное всплывающее окно содержит кнопки «Продолжить» и «Заново» с функционалом, аналогичным кнопкам из всплывающего окна паузы. При появлении окна окончания игры, прежний UI экрана, за исключением прицела, сохраняется.

## 2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта

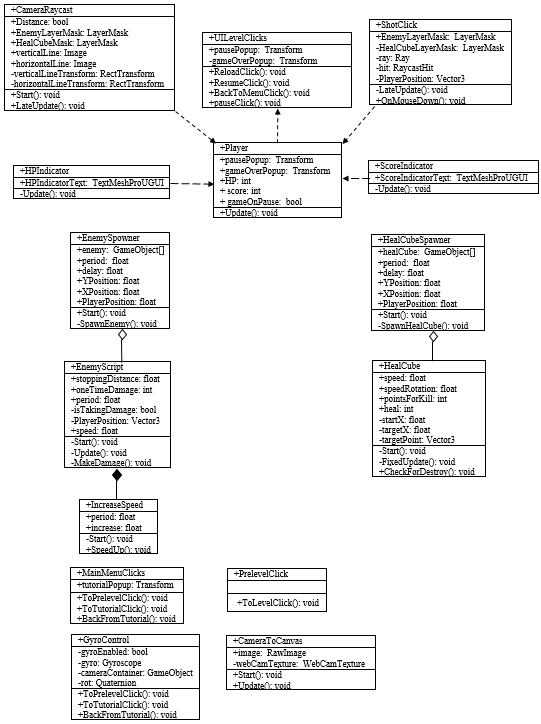


Рисунок 16 – Диаграмма классов приложения

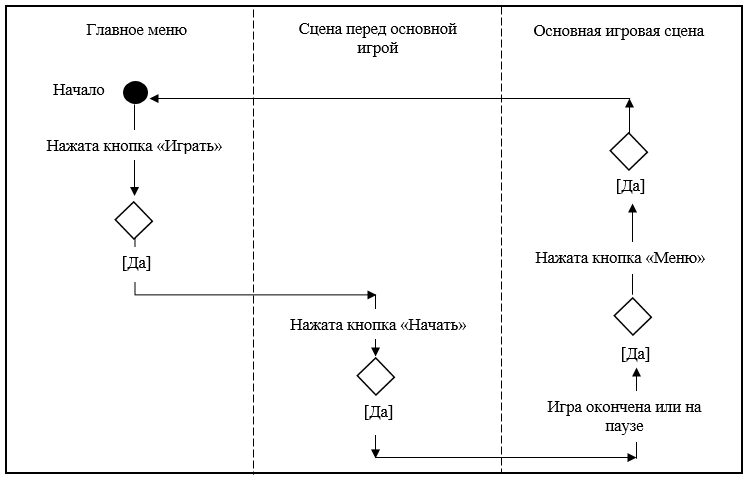


Рисунок 17 – Диаграмма взаимодействия сцен

## 2.4 Описание переменных, компонентов, классов и подпрограмм

Все методы, имеющие в своем названии слово «Update», выполняются каждый новый кадр в приложении, когда класс с этим методом активен.

Методы с названием «Start» выполняются до первого вызова Update.

MonoBehaviour – класс стандартной библиотеки Unity, от которого по умолчанию наследуются все прочие классы.

1. Public class Player : MonoBehaviour

Описание: Класс игрока и текущего состояния игры

Таблица 1 – Переменные класса Player

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| public | Transform | pausePopup | Всплывающее окно паузы |
| public static | Transform | gameOverPopup | Всплывающее окно окончания игры |
| Продолжение таблицы 1 | | | |
| public static | int | HP | Здоровье игрока |
| public static | int | score | Счет игрока |
| public static | bool | gameOnPause | Состояние паузы |

Таблица 2 – Методы класса Player

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Update() | Если игра ставится на паузу или заканчивается, все процессы в ней останавливаются и появляется соответствующее всплывающее окно |

1. Public class CameraRaycast : MonoBehaviour

Описание: Класс прицела в основной игровой сцене

Таблица 3 – Переменные класса CameraRaycast

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| public | float | Distance | Дистанция между камерой и противником |
| public | LayerMask | EnemyLayerMask | Слой противника |
| public | LayerMask | HealCubeMask | Слой куба исцеления |
| public | Image | verticalLine | Вертикальная линия прицела |
| public | Image | horizontalLine | Горизонтальная линия прицела |
| private | RectTransform | verticalLineTransform | Физические данные вертикальной линии прицела |
| private | RectTransform | horizontalLineTransform | Физические данные горизонтальной линии прицела |

Таблица 4 – Методы класса CameraRaycast

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Start() | Присвоение физических данных компонентам прицела |
| private | void | LateUpdate() | Если игра не поставлена на паузу и не окончена, при наведении на любой игровой объект, прицел увеличивается и становится красным. Если игра на паузе или окончена, он становится невидимым |

3. Public class UILevelClicks : MonoBehaviour

Описание: Класс обработчиков нажатия кнопок основной игровой сцены

Таблица 5 – Переменные класса UILevelClicks

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| ic | Transform | pausePopup | Всплывающее окно паузы |
| public | Transform | gameOverPopup | Всплывающее окно окончания игры |

Таблица 6 – Переменные класса UILevelClicks

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| public | void | ReloadClick() | Перезапуск игровой сцены |
| public | void | ResumeClick() | Продолжение текущей игровой сессии |
| public | void | BackToMenuClick() | Переход в главное меню |
| public | void | pauseClick() | Нажатие на паузу |

4. Public class ShotClick : MonoBehaviour

Описание: Класс выстрелов в основной игровой сцене

Таблица 7 – Переменные класса ShotClick

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| public | LayerMask | EnemyLayerMask | Слой противника |
| public | LayerMask | HealCubeLayerMask | Слой куба исцеления |
| private | Ray | ray | Луч, направленный от камеры в определенную сторону |
| private | RaycastHit | hit | Получатель информации об объекте, в который попал луч |
| private | Vector3 | PlayerPosition | Позиция игрока в игровой сцене |

Таблица 8 – Методы класса ShotClick

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | LateUpdate() | Присвоение лучу начальной точки и направления |
| public | void | OnMouseDown() | При нажатии на экран, не занятого кликабельными элементами UI, выявляется объект, в который в этот момент попал луч, если таковой имеется. Если это противник, он уничтожается и игроку начисляются очки, соразмерные расстоянию между камерой в игровой сцене и этим противником. Если это куб исцеления, он также уничтожается, игроку начисляется фиксированное количество очков, а также восстанавливается определенное количество здоровья |

5. Public class HPIndicator : MonoBehaviour

Описание: Класс отображения здоровья игрока в основной игровой сцене

Таблица 9 – Переменные класса HPIndicator

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| public | TextMeshProUGUI | HPIndicatorText | Текстовое поле индикатора здоровья |

Таблица 10 – Методы класса HPIndicator

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Update() | Отображение здоровья на своем текстовом поле |

6. Public class ScoreIndicator : MonoBehaviour

Описание: Класс отображения набранных очков игрока в основной игровой сцене

Таблица 11 – Переменные класса ScoreIndicator

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| public | TextMeshProUGUI | ScoreIndicatorText | Текстовое поле индикатора набранных очков |

Таблица 12 – Методы класса ScoreIndicator

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Update() | Отображение набранных очков на своем текстовом поле |

7. Public class EnemySpowner : MonoBehaviour

Описание: Класс появления противников в основной игровой сцене

Таблица 13 – Переменные класса EnemySpowner

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| public | GameObject[] | enemy | Массив с игровыми объектами противников |
| public | float | period | Периодичность появления противников |
| public | float | delay | Задержка перед первым появлением |
| public | float | YPosition | Верхняя граница области появления, нижняя равна противоположному числу |
| public | float | XPosition | Правая граница области появления, левая равна  противоположному числу |
| public | float | ZPosition | Положение области появления по оси Z в  игровой сцене |
|  | | | |
| Продолжение таблицы 13 | | | |
| private | int | rand | Индекс случайно выбранного противника для появления |

Таблица 14 – Методы класса Player

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| public | void | Start() | Начало действия повторяюшегося метода появления противников |
| private | void | SpawnEnemy() | Случайное определение противника для появления в сцене. Место появления также выбирается случайно в пределах области появления |

8. Public class EnemyScript : MonoBehaviour

Описание: Класс противника

Таблица 15 – Переменные класса EnemyScript

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Продолжение таблицы 15 | | | |
| Модификаторы | Тип | Название | Описание |
| public | float | stoppingDistance | Минимальная дистанция между противником и игроком |
| public | int | oneTimeDamage | Единоразовый урон |
| public | float | period | Период между нанесениями урона |
| private | bool | isTakingDamage | Нанесение урона противником в данный момент |
| private static | Vector3 | PlayerPosition | Позиция камеры в игровой сцене |
| public static | float | speed | Начальная скорость противников |
| public static | float | defaultSpeed | Скорость противников по умолчанию |

Таблица 16 – Методы класса EnemyScript

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Start() | Изменение положения противников, чтобы они при появлении поворачивались к  игроку правильной стороной |
| private | void | Update() | Если расстояние между камерой и противником меньше минимальной дистанции, последний движется в сторону первого. Если нет, враг начинает наносить урон |
| private | void | MakeDamage() | Нанесение урона |

9. Public class IncreaseSpeed : MonoBehaviour

Описание: Класс постоянного повышения скорости противников

Таблица 17 – Переменные класса IncreaseSpeed

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
|  |  | Продолжение таблицы 17 | |
| public | float | period | Периодичность повышения скорости |
| public | float | increase | Значение повышения скорости |

Таблица 18 – Методы класса IncreaseSpeed

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Start() | Начало действия повторяюшегося метода повышения скорости противников |
| public | void | SpeedUp() | Повышение скорости противников |

10. Public class HealCubeSpawner : MonoBehaviour

Описание: Класс появления кубов исцеления в основной игровой сцене

Таблица 19 – Переменные класса HealCubeSpawner

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| public | GameObject[] | healCube | Массив с игровыми объектами кубов исцеления |
| public | float | period | Периодичность появления кубов исцеления |
| public | float | delay | Задержка перед первым появлением |
| public | float | YPosition | Верхняя граница области появления, нижняя равна противоположному числу |
| public | float | ZPosition | Положение области появления по оси Z в игровой сцене |
| public | float | maxRightPosition | Правая граница области появления,  левая равна противоположному числу |
| Продолжение таблицы 19 | | | |
| private | int | rand | Индекс случайно выбранного куба исцеления для появления |
| private | float | startPositionX | Значение по оси X начальной позиции куба исцеления |
| private | int | positionVariant | Вариант значения по оси X начальной позиции куба исцеления |

Таблица 20 – Методы класса HealCubeSpawner

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Start() | Начало действия повторяюшегося метода появления кубов исцеления |
| private | void | SpawnHealCube() | Случайным образом определяется вариант для значения по оси X начальной позиции куба исцеления. В одном из двух случаев это значение равно правой границе области появления, в другом - левой. После определения варианта куб исцеления появляется на позиции, где Х опредлен самим вариантом, Y выбран  случайно в пределах верней и нижней границ области появления, а Z равен данному значению области исцеления |

11. Public class HealCube : MonoBehaviour

Описание: Класс куба исцеления

Таблица 21 – Переменные класса HealCube

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| public | float | speed | Скорость кубов исцеления |
| public | float | speedRotation | Скорость вращения кубов вокруг своей вертикальной оси |
| public static | int | pointsForKill | Количество очков за попадание по кубу исцеления |
| public static | int | heal | Количество очков здоровья, восстанавливаемое игроку за попадание по кубу исцеления |
| private | float | startX | Значение начальной позиции в игровой сцене по оси X |
|  | | | |
| private | float | targetX | Значение конечной позиции в игровой сцене по оси X |
| private | Vector3 | targetPoint | Конечная позиция куба исцеления в игровой сцене |

Таблица 22 – Методы класса HealCube

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Start() | Присвоение переменной startX значения начального положения куба исцеления по оси Х.  В качестве значения X для конечной точки выбирается противоположная сторона области появления |
| private | void | FixedUpdate() | Проверяется, подлежит ли куб уничтожению. Если это условие не соблюдено, он вращается вокруг своей вертикальной оси и движется к своей конечной точке |
| Продолжение таблицы 22 | | | |
| public | void | CheckForDestroy() | Если куб исцеления достиг своей конечной точки, он уничтожается |

12. Public class MainMenuClicks : MonoBehaviour

Описание: Класс обработчиков нажатий для кнопок стартового меню приложения

Таблица 23 – Переменные класса MainMenuClicks

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| public | Transform | tutorialPopup | Всплывающее окно туториала |

Таблица 24 – Методы класса MainMenuClicks

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| public | void | ToPrelevelClick() | Переход на сцену перед основной игрой |
| public | void | ToTutorialClick() | Открытие окна туториала |
| public | void | BackFromTutorial() | Закрытие окна туториала |

13. Public class GyroControl : MonoBehaviour

Описание: Класс поворота камеры в игровой в сцене в соответствии с изменениями положения

телефона в пространстве

Таблица 25 – Переменные класса GyroControl

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Модификаторы | Тип | Название | Описание |
| private | bool | gyroEnabled | Наличие включенного гироскопа |
| private | Gyroscope | gyro | Данные гироскопа |
|  |  | Продолжение таблицы 25 | |
| private | GameObject | cameraContainer | Контейнер для камеры в игровой сцене |
| private | Quaternion | rot | Поворот камеры в игровой сцене |

Таблица 26 – Методы класса GyroControl

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Start() | Назначение контейнера камере |
| private | bool | EnableGyro() | Проверка устройства на наличие включенного гироскопа |
| private | void | Update() | Изменения положения камеры в сцене синхронно с изменениями положения телефона |

14. Public class PrelevelClick : MonoBehaviour

Описание: Класс обработчика нажатия для кнопки перехода со сцены перед основной игрой к игровой сцене

Таблица 27 – Методы класса PrelevelClick

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| public | void | ToLevelClick() | Переход на основную игровую сцену |

15. Public class CameraToCanvas : MonoBehaviour

Описание: Класс отображения изображения с камеры телефона в качестве фона в основной игровой сцене

Таблица 28 – Переменные класса CameraToCanvas

|  |  |  |  |
| --- | --- | --- | --- |
| Переменные | | | |
| Продолжение таблицы 28 | | | |
| Модификаторы | Тип | Название | Описание |
| public | RawImage | image | Картинка под изображение с камеры телефона |
| private | WebCamTexture | webCamTexture | Изображение с камеры телефона |

Таблица 29 – Методы класса CameraToCanvas

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| Модификаторы | Тип возвращаемого значения | Название | Описание |
| private | void | Start() | Проверка поддержки устройства камеры и включение таковой, в случае ее наличия |
| public | void | Update() | Передача изображения с камеры на картинку |

# 3 Тестирование и установка приложения

Тестирование производится на реальном устройстве, поскольку на эмуляторе недоступны камера и гироскоп.

Установка разработанного мобильного приложения на тестируемое устройство:

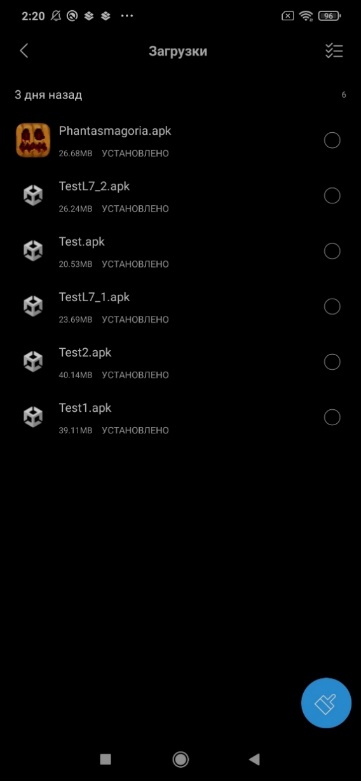


Рисунок 18 – Приложение в папке телефона

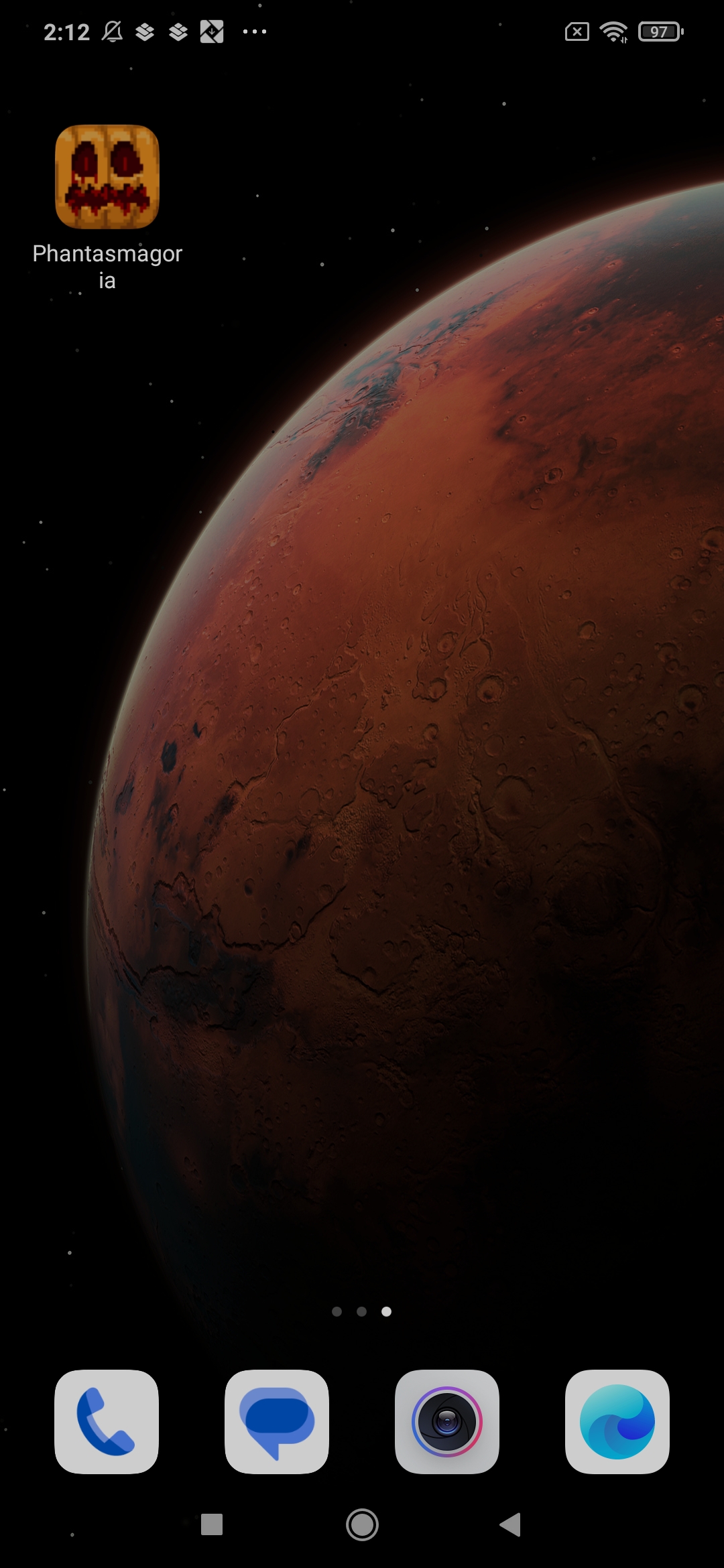


Рисунок 19 – Иконка приложения на экране телефона

Установка приложения на тестируемое устройство прошло успешно.

Тестирование приложения:



Рисунок 20 – Интерфейс приложения при запуске

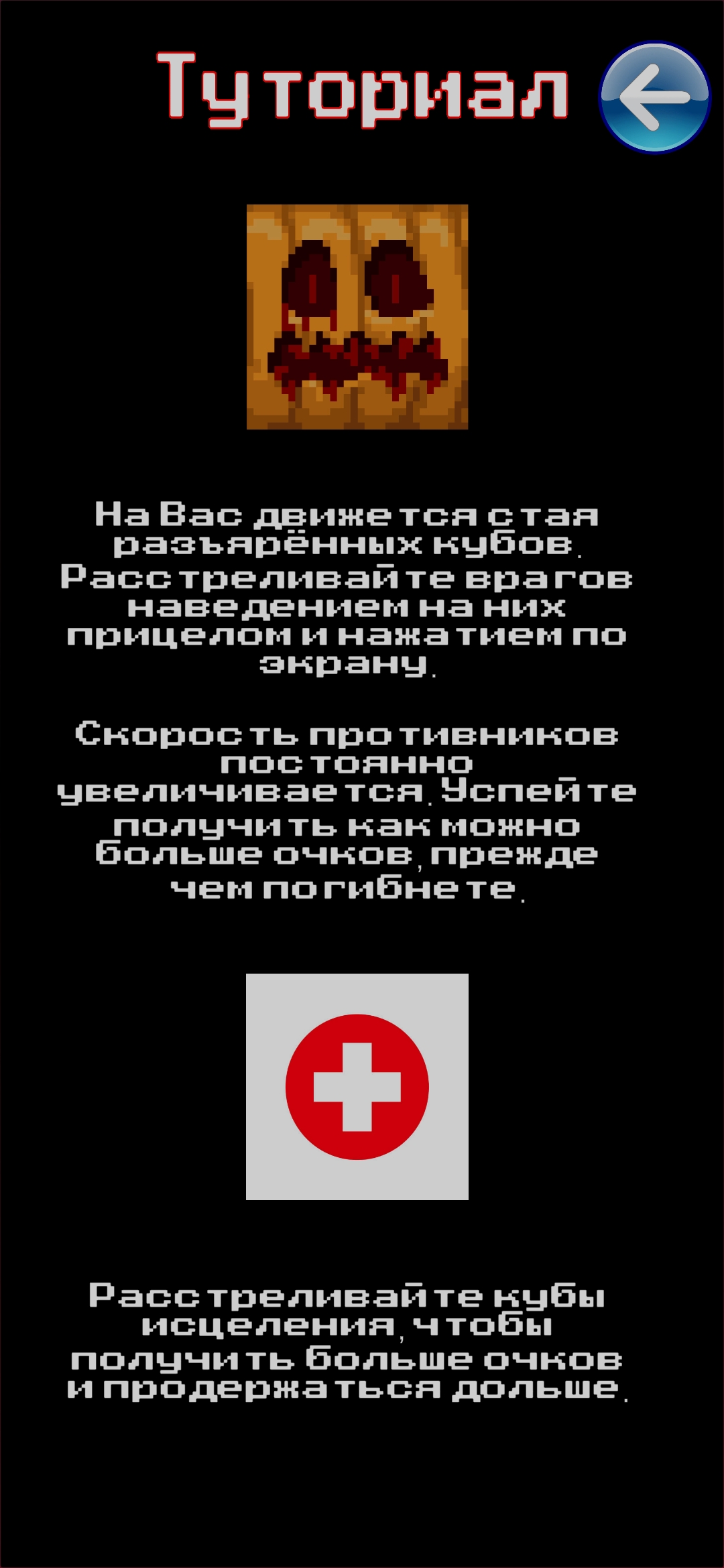


Рисунок 21 – Интерфейс при нажатии на кнопку «Туториал»



Рисунок 22 – Интерфейс при нажатии на кнопку «Играть»

При нажатии синей кнопки в окне туториала происходит возвращение на главное меню. Интерфейс стартовой сцены работает успешно.



Рисунок 23 – Интерфейс при нажатии на кнопку «Начать»



Рисунок 24 – Интерфейс приложения при непродолжительной игре

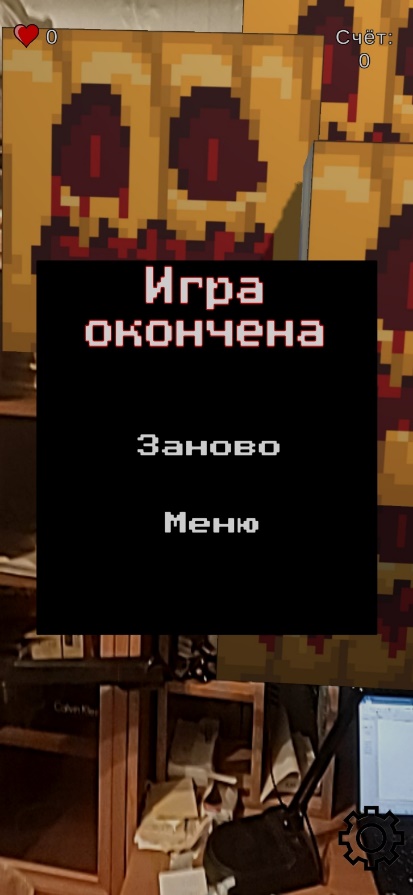


Рисунок 25 – Интерфейс приложения при понижении здоровья до нуля

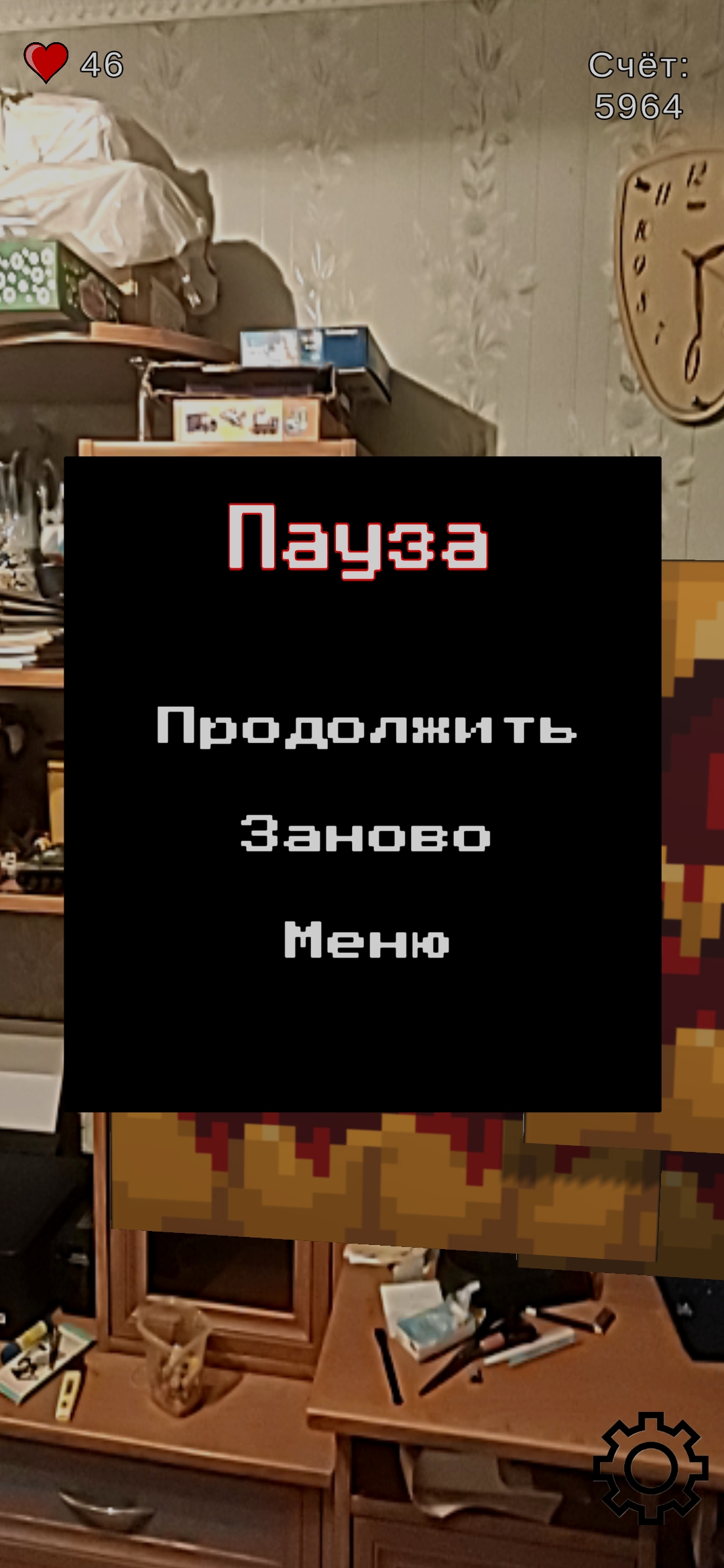


Рисунок 26 – Интерфейс приложения при нажатии на паузу

Кнопка «Заново» на всплывающих окнах перезагружает уровень. Индикаторы здоровья и очков возвращают свои начальные значения.

Кнопка «Меню» возвращает окно стартового меню. При переходе в основную игровую сцену индикаторы показывают ожидаемые значения.

Кнопка «Продолжить» снимает приложение с паузы. Значение индикаторов остается прежним.

При нажатии на кнопку в нижнем правом углу основной игровой сцены игра ставится на паузу и появляется соответствующее всплывающее окно.

Изображение с экрана телефона отображается корректно и стабильно изменяется на фоне во время игровой сессии.

Пользовательский интерфейс основной игровой сцены работает корректно.

Во время игры индикаторы показывают ожидаемые значения при различных событиях в игре, таких как выстрел в противника или получение урона.

В приложении проводятся верные расчеты в ответ на действия игрока и события в игре.

В результате тестирования приложения выяснилось, что оно соответствует своим изначальным требованиям и является корректно работающим продуктом. Все определенные задачи по созданию приложения были успешно выполнены.

# Заключение

За время выполнения курсового проекта было разработано мобильное приложение – AR – игра «Phantasmagoria».

В ходе работы над проектом был определена актуальность разработки подобного продукта; разбор его аналогов, выявление их сильных и слабых сторон; функционал приложения; его интерфейс; средства разработки; а также требования к получившейся программе.

В результате тестирования приложения выяснилось, что оно соответствует своим изначальным требованиям и является корректно работающим продуктом. Все определенные задачи по созданию приложения были успешно выполнены.

# Список использованной литературы

1. Unity – skillfactory media [Электронный ресурс]. – Режим доступа: - <https://blog.skillfactory.ru/glossary/unity/#:~:text=Unity%>
2. Android Studio: среда разработки мобильных приложений - [Электронный ресурс]. – Режим доступа: - <https://arduinoplus.ru/android-studio/>
3. Какой движок выбрать для игры? Преимущества и недостатки Unreal Engine из нашего опыта. - [Электронный ресурс]. – Режим доступа: - <https://dzen.ru/media/id/5fbd4a6e14e1585f869861ae/kakoi-dvijok-vybrat-dlia-igry-preimuscestva-i-nedostatki-unreal-engine-iz-nashego-opyta-5fc8aac4aa857937981323fd>
4. Pokemon Go – Wikipedia - [Электронный ресурс]. – Режим доступа: - <https://ru.wikipedia.org/wiki/Pok%C3%A9mon_Go>
5. Обзор игры The Walking Dead: Our World - [Электронный ресурс]. – Режим доступа: - https://androidelf.com/obzor-igryi-the-walking-dead-our-world/
6. Night Terrors: Bloody Mary - [Электронный ресурс]. – Режим доступа: - https://www.alpha-ag.ru/games-for-android/action\_shooter/20279-night-terrors-bloody-mary.html
7. Сравнение популярных игровых движков: от бесплатного Godot до старейшего Unreal Engine - [Электронный ресурс]. – Режим доступа: - https://www.mvideo.ru/blog/pomogaem-razobratsya/populyarnyh-igrovyh-dvizhkov
8. Руководство пользователя Unity 2021.3 (LTS) [Электронный ресурс]. – Режим доступа: - <https://docs.unity3d.com/Manual/index.html>
9. Официальная страница AR Foundation Unity [Электронный ресурс]. – Режим доступа: - <https://unity.com/ru/unity/features/arfoundation>
10. Официальная страница ARCore Google [Электронный ресурс]. – Режим доступа: -https://developers.google.com/ar?hl=ru
11. Устройства с поддержкой ARCore [Электронный ресурс]. – Режим доступа: - <https://developers.google.com/ar/devices?hl=ru>
12. Изучение интерфейса Unity [Электронный ресурс] - Режим доступа: - <https://docs.unity3d.com/ru/530/Manual/LearningtheInterface.html>
13. Создание дизайна, прототипов и совместная работа в Adobe XD [Электронный ресурс]. – Режим доступа: - <https://helpx.adobe.com/ru/xd/help/adobe-xd-overview.html>

Приложение А. Листинг кода

Класс Player:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

//Класс игрока и текущего состояния игры

public class Player : MonoBehaviour

{

public Transform pausePopup; //Всплывающее окно паузы

public Transform gameOverPopup; //Всплывающее окно окончания игры

public static int HP = 100; //Здоровье игрока

public static int score = 0; //Счет игрока

public static bool gameOnPause = false; //Состояние паузы

//Если игра ставится на паузу или заканчивается, все процессы в ней

//останавливаются и появляется соответствующее всплывающее окно

void Update()

{

if (gameOnPause)

{

Time.timeScale = 0f;

pausePopup.gameObject.SetActive(true);

pausePopup.SetAsFirstSibling();

}

if (HP <= 0)

{

Time.timeScale = 0f;

gameOverPopup.gameObject.SetActive(true);

gameOverPopup.SetAsFirstSibling();

}

}

}

Класс UILevelClicks:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

//Класс обработчиков нажатия кнопок основной игровой сцены

public class UILevelClicks : MonoBehaviour

{

public Transform pausePopup; //Всплывающее окно паузы

public Transform gameOverPopup; //Всплывающее окно окончания игры

//Перезапуск игровой сцены

public void ReloadClick()

{

Time.timeScale = 1f;

pausePopup.gameObject.SetActive(false);

gameOverPopup.gameObject.SetActive(false);

SceneManager.LoadScene("LevelScene");

Player.gameOnPause = false;

Player.HP = 100;

Player.score = 0;

EnemyScript.speed = EnemyScript.defaultSpeed;

}

//Продолжение текущей игровой сессии

public void ResumeClick()

{

Time.timeScale = 1f;

Player.gameOnPause = false;

pausePopup.gameObject.SetActive(false);

}

//Переход в главное меню

public void BackToMenuClick()

{

Time.timeScale = 1f;

Player.gameOnPause = false;

pausePopup.gameObject.SetActive(false);

gameOverPopup.gameObject.SetActive(false);

SceneManager.LoadScene("StartScene");

Player.HP = 100;

Player.score = 0;

EnemyScript.speed = EnemyScript.defaultSpeed;

}

//Нажатие на паузу

public void pauseClick()

{

Player.gameOnPause = true;

}

}

Класс CameraRaycast:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

//Класс прицела в основной игровой сцене

public class CameraRaycast : MonoBehaviour

{

public float Distance; //Дистанция между камерой и противником

public LayerMask EnemyLayerMask; //Слой противника

public LayerMask HealCubeMask; //Слой куба исцеления

public Image verticalLine; //Вертикальная линия прицела

public Image horizontalLine; //Горизонтальная линия прицела

private RectTransform verticalLineTransform; //Физические данные вертикальной линии прицела

private RectTransform horizontalLineTransform; //Физические данные горизонтальной линии прицела

//Присвоение физических данных компонентам прицела

void Start()

{

verticalLineTransform = verticalLine.GetComponent<RectTransform>();

horizontalLineTransform = horizontalLine.GetComponent<RectTransform>();

}

//Если игра не поставлена на паузу и не окончена, при наведении на любой игровой

//объект, прицел увеличивается и становится красным. Если игра на паузе или окончена,

//он становится невидимым

void LateUpdate()

{

Ray ray = new Ray(transform.position, transform.forward);

if(!Player.gameOnPause && Player.HP > 0)

{

if (Physics.Raycast(ray, Distance, EnemyLayerMask) || Physics.Raycast(ray, Distance, HealCubeMask))

{

verticalLine.color = Color.red;

horizontalLine.color = Color.red;

verticalLineTransform.sizeDelta = new Vector2(10, 100);

horizontalLineTransform.sizeDelta = new Vector2(100, 10);

}

else

{

verticalLine.color = Color.black;

horizontalLine.color = Color.black;

verticalLineTransform.sizeDelta = new Vector2(8, 80);

horizontalLineTransform.sizeDelta = new Vector2(80, 8);

}

}

else

{

verticalLine.color = Color.clear;

horizontalLine.color = Color.clear;

}

}

}

Класс ShotClick:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.EventSystems;

//Класс выстрелов в основной игровой сцене

public class ShotClick : MonoBehaviour

{

public LayerMask EnemyLayerMask; //Слой противника

public LayerMask HealCubeLayerMask; //Слой куба исцеления

private Ray ray; //Луч, направленный от камеры в определенную сторону

private RaycastHit hit; //Получатель информации об объекте, в который попал луч

private Vector3 PlayerPosition; //Позиция игрока в игровой сцене

//Присвоение лучу начальной точки и направления

private void LateUpdate()

{

PlayerPosition = Camera.main.transform.position;

ray = new Ray(PlayerPosition,

Camera.main.transform.forward);

}

//При нажатии на экран, не занятого кликабельными элементами UI, выявляется

//объект, в который в этот момент попал луч, если таковой имеется. Если это

//противник, он уничтожается и игроку начисляются очки, соразмерные расстоянию

//между камерой в игровой сцене и этим противником. Если это куб исцеления, он

//также уничтожается, игроку начисляется фиксированное количество очков, а также

//восстанавливается определенное количество здоровья так, чтобы его общий объем не

//превышал максимальный

public void OnMouseDown()

{

if(!Player.gameOnPause && Player.HP > 0)

{

if (Physics.Raycast(ray, out hit, 1000, EnemyLayerMask))

{

Player.score += (int)Vector3.Distance(transform.position, PlayerPosition);

Destroy(hit.collider.gameObject);

}

if (Physics.Raycast(ray, out hit, 1000, HealCubeLayerMask))

{

Player.score += HealCube.pointsForKill;

Destroy(hit.collider.gameObject);

for (int i = 0; i < HealCube.heal && Player.HP < 100; ++i)

{

++Player.HP;

}

}

}

}

}

Класс HPIndicator:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

//Класс отображения здоровья игрока в основной игровой сцене

public class HPIndicator : MonoBehaviour

{

public TextMeshProUGUI HPIndicatorText; //Текстовое поле индикатора здоровья

//Отображение здоровья на своем текстовом поле

void Update()

{

HPIndicatorText.text = Player.HP.ToString();

}

}

Класс ScoreIndicator:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

//Класс отображения набранных очков игрока в основной игровой сцене

public class ScoreIndicator : MonoBehaviour

{

public TextMeshProUGUI ScoreIndicatorText; //Текстовое поле индикатора

//набранных очков

//Отображение набранных очков на своем текстовом поле

void Update()

{

ScoreIndicatorText.text = "Счёт:\n" + Player.score;

}

}

Класс EnemySpowner:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

//Класс появления противников в основной игровой сцене

public class EnemySpowner : MonoBehaviour

{

public GameObject[] enemy; //Массив с игровыми объектами противников

public float period; //Периодичность появления противников

public float delay; //Задержка перед первым появлением

public float YPosition = 16.5f; //Верхняя граница области появления, нижняя равна

//противоположному числу

public float XPosition = 14.5f; //Правая граница области появления, левая равна

//противоположному числу

public float ZPosition = -4.21f; //Положение области появления по оси Z в

//игровой сцене

private int rand; //Индекс случайно выбранного противника для появления

//Начало действия повторяюшегося метода появления противников

public void Start()

{

InvokeRepeating("SpawnEnemy", delay, period);

}

//Случайное определение противника для появления в сцене. Место появления также

//выбирается случайно в пределах области появления

private void SpawnEnemy()

{

rand = Random.Range(0, enemy.Length);

Instantiate(enemy[rand], new Vector3(Random.Range(-XPosition, XPosition), Random.Range(-YPosition, YPosition), ZPosition),

Quaternion.identity);

}

}

Класс EnemyScript:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

//Класс противника

public class EnemyScript : MonoBehaviour

{

public float stoppingDistance; //Минимальная дистанция между противником и

//игроком

public int oneTimeDamage; //Единоразовый урон

public float period; //Период между нанесениями урона

private bool isTakingDamage; //Нанесение урона противником в данный момент

private static Vector3 PlayerPosition; //Позиция камеры в игровой сцене

public static float speed = 8; //Начальная скорость противников

public static float defaultSpeed = speed; //Скорость противников по умолчанию

//Изменение положения противников, чтобы они при появлении поворачивались к

//игроку правильной стороной

void Start()

{

transform.Rotate(Vector3.up \* 180);

}

//Если расстояние между камерой и противником меньше минимальной дистанции,

//последний движется в сторону первого. Если нет, враг начинает наносить урон

void Update()

{

PlayerPosition = Camera.main.transform.position;

if (Vector3.Distance(transform.position, PlayerPosition) > stoppingDistance)

{

transform.position = Vector3.MoveTowards

(transform.position, PlayerPosition, speed \* Time.deltaTime);

isTakingDamage = false;

}

else

{

transform.position = this.transform.position;

if(Player.HP > 0)

{

if (!isTakingDamage)

{

InvokeRepeating("MakeDamage", 0, period);

isTakingDamage = true;

}

}

else

{

CancelInvoke("MakeDamage");

}

}

}

//Нанесение урона

private void MakeDamage()

{

Player.HP -= oneTimeDamage;

}

}

Класс HealCubeSpawner:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

//Класс появления кубов исцеления в основной игровой сцене

public class HealCubeSpawner : MonoBehaviour

{

public GameObject[] healCube; //Массив с игровыми объектами кубов исцеления

public float period; //Периодичность появления кубов исцеления

public float delay; //Задержка перед первым появлением

public float YPosition = 16.5f; //Верхняя граница области появления, нижняя

//равна противоположному числу

public float ZPosition = -4.21f; //Положение области появления по оси Z в

//игровой сцене

public static float maxRightPosition = 14.5f; //Правая граница области появления,

//левая равна противоположному числу

private int rand; //Индекс случайно выбранного куба исцеления для появления

private float startPositionX; //Значение по оси X начальной позиции куба исцеления

private int positionVariant; //Вариант значения по оси X начальной позиции куба исцеления

//Начало действия повторяюшегося метода появления кубов исцеления

void Start()

{

InvokeRepeating("SpawnHealCube", delay, period);

}

//Случайным образом определяется вариант для значения по оси X начальной

//позиции куба исцеления. В одном из двух случаев это значение равно правой

//границе области появления, в другом - левой. После определения варианта куб

//исцеления появляется на позиции, где Х опредлен самим вариантом, Y выбран

//случайно в пределах верней и нижней границ области появления, а Z равен данному

//значению области исцеления

private void SpawnHealCube()

{

positionVariant = Random.Range(0, 2);

if (positionVariant == 0)

{

startPositionX = -maxRightPosition;

}

else

{

startPositionX = maxRightPosition;

}

rand = Random.Range(0, healCube.Length);

Instantiate(healCube[rand], new Vector3(startPositionX, Random.Range(-YPosition, YPosition), ZPosition),

Quaternion.identity);

}

}

Класс HealCube:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

//Класс куба исцеления

public class HealCube : MonoBehaviour

{

public float speed; //Скорость кубов исцеления

public float speedRotation; //Скорость вращения кубов вокруг своей вертикальной оси

public static int pointsForKill = 1200; //Количество очков за попадание по кубу исцеления

public static int heal = 25; //Количество очков здоровья, восстанавливаемое игроку

//за попадание по кубу исцеления

private float startX; //Значение начальной позиции в игровой сцене по оси X

private float targetX; //Значение конечной позиции в игровой сцене по оси X

private Vector3 targetPoint; //Конечная позиция куба исцеления в игровой сцене

//Присвоение переменной startX значения начального положения куба исцеления по оси Х.

//В качестве значения X для конечной точки выбирается противоположная сторона

//области появления

void Start()

{

startX = transform.position.x;

if (startX == HealCubeSpawner.maxRightPosition)

{

targetX = -HealCubeSpawner.maxRightPosition;

}

else

{

targetX = HealCubeSpawner.maxRightPosition;

}

}

//Проверяется, подлежит ли куб уничтожению. Если это условие не соблюдено,

//он вращается вокруг своей вертикальной оси и движется к своей конечной точке

void FixedUpdate()

{

CheckForDestroy();

transform.Rotate(0, speedRotation \* Time.deltaTime, 0);

transform.position = Vector3.MoveTowards(transform.position, new Vector3(targetX, transform.position.y, transform.position.z), speed \* Time.deltaTime);

targetPoint = new Vector3(targetX, transform.position.y, transform.position.z);

}

//Если куб исцеления достиг своей конечной точки, он уничтожается

public void CheckForDestroy()

{

if(transform.position == targetPoint)

{

Destroy(gameObject);

}

}

}

Класс MainMenuClicks:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

//Класс обработчиков нажатий для кнопок стартового меню приложения

public class MainMenuClicks : MonoBehaviour

{

public Transform tutorialPopup; //Всплывающее окно туториала

//Переход на сцену перед основной игрой

public void ToPrelevelClick()

{

SceneManager.LoadScene("PrelevelScene");

}

//Открытие окна туториала

public void ToTutorialClick()

{

tutorialPopup.gameObject.SetActive(true);

tutorialPopup.SetAsLastSibling();

}

//Закрытие окна туториала

public void BackFromTutorial()

{

tutorialPopup.gameObject.SetActive(false);

}

}

Класс GyroControl:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

//Класс поворота камеры в игровой в сцене в соответствии с изменениями положения

//телефона в пространстве

public class GyroControl : MonoBehaviour

{

private bool gyroEnabled; //Наличие включенного гироскопа

private Gyroscope gyro; // Данные гироскопа

private GameObject cameraContainer; // Контейнер для камеры в игровой сцене

private Quaternion rot; // Поворот камеры в игровой сцене

//Назначение контейнера камере

void Start()

{

cameraContainer = new GameObject("Camera Container");

cameraContainer.transform.position = transform.position;

transform.SetParent(cameraContainer.transform);

gyroEnabled = EnableGyro();

}

//Проверка устройства на наличие включенного гироскопа

private bool EnableGyro()

{

if (SystemInfo.supportsGyroscope)

{

gyro = Input.gyro;

gyro.enabled = true;

cameraContainer.transform.rotation = Quaternion.Euler(90f, 90f, 0f);

rot = new Quaternion(0, 0, 1, 0);

return true;

}

return false;

}

//Изменения положения камеры в сцене синхронно с изменениями положения телефона

private void Update()

{

transform.localRotation = gyro.attitude \* rot;

}

}

Класс PrelevelClick:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

//Класс обработчика нажатия для кнопки перехода со сцены перед основной игрой к

//игровой сцене

public class PrelevelClick : MonoBehaviour

{

//Переход на основную игровую сцену

public void ToLevelClick()

{

SceneManager.LoadScene("LevelScene");

}

}

Класс CameraToCanvas:

using UnityEngine;

using UnityEngine.UI;

//Класс отображения изображения с камеры телефона в качестве фона в основной игровой сцене

public class CameraToCanvas : MonoBehaviour

{

public RawImage image; //Картинка под изображение с камеры телефона

private WebCamTexture webCamTexture; //Изображение с камеры телефона

//Проверка поддержки устройства камеры и включение таковой, в случае ее наличия

void Start()

{

if (Application.isMobilePlatform && !WebCamTexture.devices.Length.Equals(0))

{

string deviceName = WebCamTexture.devices[0].name;

webCamTexture = new WebCamTexture(deviceName);

webCamTexture.Play();

}

}

// Передача изображения с камеры на картинку

void Update()

{

image.texture = webCamTexture;

}

}

Приложение Б. Руководство пользователя

При входе в игру нужно нажать кнопку «Туториал», чтобы открыть информационное окно и ознакомиться с механиками игры. Для выхода из данного окна необходимо нажать кнопку выхода в правом верхнем углу.

Интерфейс меню:



Информационное окно туториала:



Чтобы открыть сцену перед основной игрой, необходимо нажать на кнопку «Играть». После этого нужно вращать экран до тех пор, пока не будет найдена моделька противника. После обнаружения нажать кнопку «Начать», чтобы перейти к основной игре.

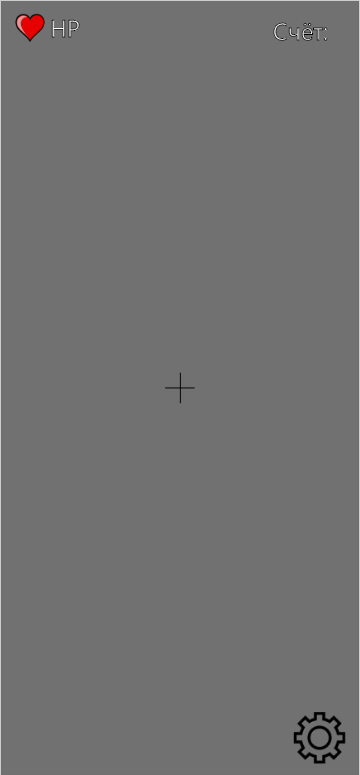
Сцена перед основной игрой:



Во время партии с на экране, фоном которому служит изображение с камеры телефона пользователя, с будут появляться противники и кубы исцеления. Задача игрока состоит в том, чтобы уничтожать врагов и продержаться как можно дольше, постаравшись набрать наивысший счёт.

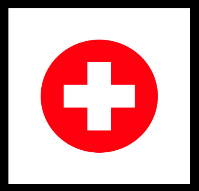
Чтобы уничтожить цель, нужно навести на нее прицел в центре экрана. Последний становится красным, если под прицелом стоит объект, который можно уничтожить. Чтобы осуществить выстрел и уничтожить нацеленную мишень, нужно нажать в любом месте экрана, кроме кнопки паузы в нижнем правом углу. Если игрок попал в цель, она уничтожается и исчезает с экрана, а любые процессы, проводимые этой целью, завершаются.

Интерфейс приложения во время основной игры:



Противники  появляются на экране в разных местах с одинаковой периодичностью. Зона их появления находится за моделькой таково же врага в сцене перед основной игрой. Все они движутся с одинаковой скоростью, которая повышается со временем.

Если противник долетает слишком близко к игроку, он начинает наносить ему урон до тех пор, пока либо сам враг не будет уничтожен, либо не закончится здоровье у игрока.

Кубы исцеления  появляются реже противников и движутся по горизонтали с постоянной скоростью мимо игрока, начиная путь с одного края зоны появления противников, и автоматически уничтожаясь при достижении другого. При уничтожении такого куба игроком, последнему начисляется повышенное количество очков и частичное восстановление здоровья.

Здоровье игрока указывается в верхнем левом углу экрана, а набранные очки – в верхнем правом.

Чтобы приостановить игру, игроку необходимо нажать на кнопку в нижнем правом углу. Изображение на экране по прежнему может меняться, но выстрелы становятся недоступны. Появляется всплывающее окно, на котором игроку становятся доступны 3 кнопки на выбор:

1. «Продолжить» - игра продолжается с того момента, на котором остановилась. Счет и очки остаются прежними.
2. «Заново» - здоровье и счет игрока становятся равным изначальным, как и скорость противников. Все противники и кубы исцеления, присутствующие в сцене, уничтожаются и начинают появляться заново.
3. «Меню» - игрок возвращается в главное меню. При повторном входе в основную игру она начинается как при первом запуске.

Интерфейс всплывающего окна паузы:



При обнулении здоровья игрока игра также приостанавливается и появляется всплывающее окно со схожим функционалом, за исключением отсутствия кнопки «Заново».

Интерфейс окна окончания игры:

