Содержание

Аннотация

Проектом является настольная игра "Шакал стратегия, рассчитанная на 2-4 человек, перенесенная в цифровой формат. В игре пользователям дается возможность почувствовать себя в образе капитана пиратского корябля, обнаружившего необитаемый остров с сокровищем, крокодилами и прочими активностями. Только с ним на этот остров приплыли и другие игроки с той же целью - заработать побольше золота, а, кто больше остальных набрал монет, становится победителем в игре. В приложении игроки смогут разложить поле как в неком игровом мире, так и у себя на столе с помощью системы AR, которая проецирует на экране телефона игровое поле, лежащее на столе или же другой плоской поверхности. С этим приложением пользователи смогут играть в популярную настольную игру не только у себя дома или в антикафе, но и в любый других местах, где есть подключение к интернету, что поспосубствует возвращению настольных игр в мир, где правят компьютерные их аналоги.

Ключевые слова

AR (Augmented Reality) – дополнительная реальность, воссоздаваемая на телефонах с помощью камеры, на экранах которых изображение, создаваемое приложением, накладывается на обычное, дополняя ее тем самым.

ARCore (ARKit) – библиотека для обработки визуальных пространств, движений телефона, передаваемых с датчиков, и визуальной сцены внутри приложения на Android (iPhone), разработанная компанией Google (Apple).

Unity - кроссплатформенная среда разработки компьютерных игр на телефоны, игровые консоли, компьютеры, веб приложения.

Сцена Unity – место внутри Unity, где располагаются все объекты, что будут в игре. Отражение игрового мира внутри среди разработки.

Ресурсы Unity – объект, который обладает свойствами материала, накидываемый на другие объекты, придавая им свой внешних и характеристический вид.

Sprite Unity – наборы разделенных маленьких картиночек, накладываемых на различные объекты.

Слой Unity – битовая маска, устанавливаемая на объект для дальнейшего его распознавания. Каждый слой имеет свой порядковый номер, и маска в двоичном представлении выглядит как множество 0 и 1 на позиции, соответствующей номеру маски, считая справа

налево, начиная с 0.

C# скрипт – код, написанный на языке программирования C#, благодаря которому можно запускать игровые события, изменять свойства компонентов и объектов на сцене, а также позволяет реагировать на действия пользователя.

Photon Unity Networking (PUN) – фреймворк для сетевой игры, которая позволяет разработчикам быстро и легко построить многопользовательские игры на базе Unity. Она помогает разработчикам поддерживать игровую логику в их проекте и обеспечивает масштабируемую сетевую инфраструктуру, которая позволяет игрокам соединяться и участвовать в игре в режиме реального времени.

Remote procedure call (RPC) - класс технологий, позволяющих программам вызывать функции или процедуры в другом адресном пространстве (на удалённых узлах, либо в независимой сторонней системе на том же узле).

User Interface (UI) - это система взаимодействия пользователя с игровым миром. Она включает в себя все элементы интерфейса, например меню, иконки, инструменты, кнопки и т.д., которые позволяют игроку взаимодействовать с игрой. Это обычно включает в себя меню главного меню, меню игровых настроек, меню инвентаря и т.д.

Искусственный интеллект - это область компьютерной науки, занимающаяся разработкой алгоритмов и программ, имитирующих разумное поведение. Целью ИИ является получение компьютерной системы, способной выполнять задачи, которые обычно выполняются человеком, такие как принятие решений, прогнозирование, изучение и воспроизведение человеческого поведения.

Текстуры - это изображения, используемые для задания внешнего вида игровых объектов. Они могут быть использованы для задания поверхности, растительности, дорог и других игровых элементов. Текстуры могут быть в формате изображения или вектора, в зависимости от требований игры. Использование текстур позволяет создать более детализированную и правдоподобную игровую вселенную.

1 Введение

В настоящее время игровая индустрия движется вперед семимильными шагами, а их продукт из обычного «Тетриса» и «Три в ряд» превратился в полноценное художественное произведение, которое сможет затронуть самые потаенные фибры души, или же сможет бросить игрокам вызов, который не все смогут принять, или же сможет дать то, что давно было забыто человеком или давно не использовалось. И к сожалению, к последним относятся настольные игры. Сейчас все меньше и меньше людей собираются, чтобы просто сыграть в них, отдавая свое предпочтение цифровым играм. Так была отложена на полку в свое время популярная игра-бродилка с долей стратегии «Шакал».

«Шакал» - настольная игра, рассчитанная на 2–4 человек. В ней игроки примеряют роль капитана пиратского корабля с 3 матросами, которые будут ходить по необитаемому (или нет) острову в поисках спрятанных сокровищ. На их пути будут попадаться крепости, крокодилы, пустыни и даже людоед, карточки которых обладают уникальными для себя свойствами, помогающие или же наоборот мешающие игрокам. Если между враждующими командами образовалась драка, все нокаутированные бойцы отправляются назад к себе на корабль, оставляя поклажу на месте. Игра заканчивается, когда всё золото будет растащено, а победителем будет считаться тот, кто успел забрать наибольшее число монет.

Целью проекта является реализовать AR-версию игры "Шакал".

Задачами проекта, ведущими к этой цели, являются:

- Изучить механику AR
- Проанализировать существующие приложения с AR технологией
- Реализовать логику игры
- Разработать игровой сервер
- Реализовать многопользовательский режим игры (до 4-х игроков)
- Реализовать мобильное приложение со стандартным интерфейсом 3D игры
- Реализовать мобильное приложение с AR-интерфейсом игры

Таблица 1 – Распределение задач в проекте

И. О. Фамилия участника проекта	**Задачи в проекте**
Р. Р. Черномордин	AR, игровые фишки, игровые изображения, меню, звуковая система, часть игровых карточек, часть User Interface
В. В. Ковешников	Игровое поле, функционал игровых карточек, картинки для карточек
Н. Н. Некрасов	Создание загрузочной сцены, меню, сетевой составляющей игры

2 Сравнительный анализ источников и аналогов

2.1 Unity

В самом начале нужно было выбрать среду для разработки и основной язык программирования. Основными вариантами были Unity, Unreal Engine 4, Android Studio (так как мы делаем прежде всего приложение для андроида). Android Studio нам не подходит, так как первые два более заточены на игры и работа с ARCore там намного лучше настроена. Из Unity и UE4 мы стали использовать первый, ведь в UE4 делают более сложные проекты, которые в свою очередь сильней нагружают телефон. Unity универсальный инструмент, позволяющий легко и быстро в нем освоиться и написать проект с наши техническими заданиями. Чтобы писать код, мы выбрали Rider от JetBrains, а не Visual Code, так как интерфейс продуктов от JetBrains приятней для нас, а код получается везде одинаковый.

2.2 AR

Исследуем популярные мобильные игры/приложения с AR, и одной из них является «Pokemon GO». Самой главной проблемой этого продукта является сильное потребление энергии. Чтобы устранить перегревания и оптимизировать процесс, разработчики сделали альтернативный режим поимки покемонов (изначально они «были в нашем мире» при помощи AR). Мы тоже добавили в свою игру возможность выбирать режимы отображения поля: на месте, где выбрал пользователь, и просто абстрактное поле на сцене в Unity. Для поддержки AR версия Android должна быть не позже 7. На старых моделях телефонов работать с приложением необходимо плавно, ведь иначе встроенные гироскопы, камера и вычисление объектов по ключевым точкам могут не поспеть за движениями человека, и игровое поле может начать само двигаться, так как телефон будет неправильно распознавать движения. Чтобы всем пользователям было удобно играть в «Шакал», будет реализована второй вариант отображения без применения технологий AR.

2.3 Multiplayer framework

Photon Pun - это фреймворк для создания мультиплеера в Unity, который предоставляет разработчикам мощный и простой в использовании инструмент для создания мультиплеерных игр. Существуют и другие фреймворки для создания мультиплеера в Unity, такие как UNet, Forge Networking Remastered и Mirror. Рассмотрим каждый из них подробнее.

1. UNet - это фреймворк для создания мультиплеера в Unity, который входит в стандартную поставку Unity. UNet имеет ряд функций, таких как авторитет сервера, синхронизацию объектов, RPC и многое другое. Однако, UNet больше не поддерживается Unity и не рекомендуется к использованию.

2. Forge Networking Remastered - это фреймворк для создания мультиплеера в Unity, который предоставляет разработчикам мощный и гибкий инструмент для создания мультиплеерных игр. Forge Networking Remastered имеет ряд функций, таких как авторитет сервера, синхронизацию объектов, RPC и многое другое. Однако, у Forge Networking Remastered есть некоторые ограничения, связанные с поддержкой различных платформ и удобством использования.

3. Mirror - это фреймворк для создания мультиплеера в Unity, который является форком UNet и предоставляет разработчикам мощный и гибкий инструмент для создания мультиплеерных игр. Mirror имеет ряд функций, таких как авторитет сервера, синхронизацию объектов, RPC и многое другое. Mirror поддерживает большое количество платформ и имеет удобный интерфейс.

В целом, каждый из этих фреймворков имеет свои преимущества и недостатки. UNet больше не поддерживается Unity. Forge Networking Remastered и Mirror предоставляют разработчикам мощный и гибкий инструмент для создания мультиплеерных игр, но могут иметь некоторые ограничения. Данный выбор был сделан ввиду качественной поддержки функциональности фреймворка со стороны разработчиков, обеспечения высокой производительности, простоты использования и большого комьюнити вокруг него, а также отсутствия платы за него при небольшом количестве пользователей. Он предоставляет удобную библиотеку Photon.Pun и PhotonNetwork, в которых реализованы основные методы для реализации мультиплеера.

В итоге, благодаря Photon Unity Networking (PUN), разработчики имеют мощный инструмент для создания многопользовательских игр на базе Unity, который позволяет игрокам подключаться к игре в режиме реального времени и синхронизировать игровые данные между ними.

3 Ключевые точки проекта

3.1 Структуризация проекта

«Шакал в AR» - командный проект, в котором каждый отвечает за свою часть разработки, где файлов с кодом может быть очень много, поэтому стоит поддерживать корректное дерево файлов. Также это дерево просто необходимо иметь для Unity, так как некоторые вещи среда разработки будет искать в папках с определенными именами.

Так внутри папки Assets (предназначенная для работы пользователя) расположены следующие директории (см. рисунок 1):

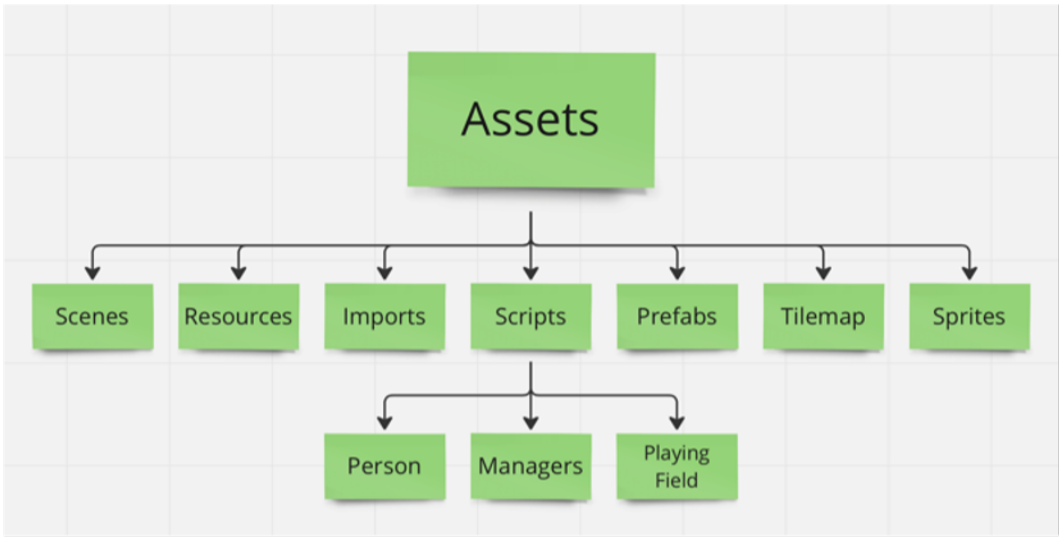

Рис. 3.1: Дерево папок

1 Scenes – все текущие сцены Unity.

2 Resources – все материалы, используемые игровыми объектами.

3 Imports – скрипты, объекты и прочее, что были добавлены из официального магазина Unity

4 Scripts – папки разделяющие скрипты по объектам. То есть в папке “Person” находятся скрипты, отвечающие за этот тип объектов.

5 Prefabs – игровые объекты, которые в дальнейшем размещаются или уже размещены на сцену.

6 Tilemap – картинки, используемые для создания материалов или вырезки sprite.

7 Sprites – все spite-ы игры.

Для поддержки параллельной разработки и контроля версий используется GitHub, в котором уже настроена работа с системными файлами Unity, а также подготовлен соответствующий .gitignore.

3.2 Augmented Reality

Первостепенной задачей при создании настольной игры как приложение является поддержка шарма таких развлечений, то есть возможность увидеть и почувствовать поле, фишки и стопки монет. Нужно сделать игру дополнительной реальностью внутри нашей, с этим поможет технология AR. Работа с AR осуществляется через библиотеки ARCore [1] и ARKit (для Android и iPhone телефонов соответственно), но для грамотной работы необходимо разобраться в том, как это работает.

Разработку приложения будет проходить в Unity, где уже реализованы вспомогательные объекты для работы с ARCore. Ключевой принцип ARCore заключается в том, что с помощью камеры он выделяет ключевые точки на изображении, чтобы помочь более точно их определить, стоит немного подвигать телефон (см. рисунок 2). Далее приложение создает внутри себя псевдо-мир, в котором определяет текущую позицию камеры, который автоматически переносится на сцену в Unity (см. рисунок 3), где уже местная камера оказывается вокруг объектов, которые созданы были в игровом движке, и показывает их. Чтобы определять повороты камеры, приложение использует встроенные в телефон гироскопы, а для определения движения камеры при ходьбе человека, оно использует ускорение точек внутри своего псевдо-мира. Все эти изменения передаются на сцену в Unity, где объект, отвечающий за игрока, тоже начинает то же движение. Таким образом происходит взаимодействие между Unity и реальным миром.

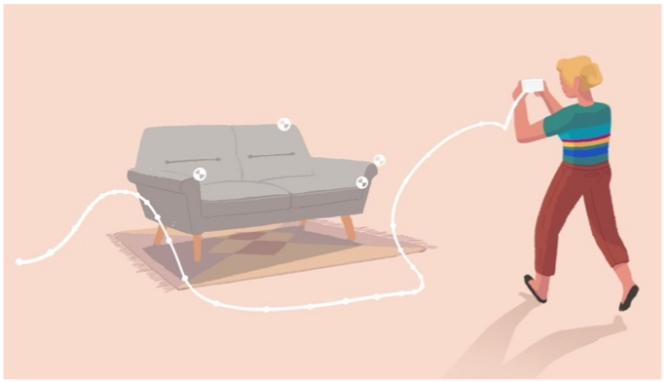

Рис. 3.2: Пример нахождения ключевых точек на объекте

В самом начале игроку предлагается выбрать место, где будет размещено игровое поле, для этого на плоскостях, которые найдет нам ARCore, будет появляться отметка. Чтобы

расположить эту метку, необходимо место для нее на сцене в Unity. Для этого пустим из центра камеры луч и в месте, где он пересечётся с поверхностью выбранного места, создастся объект, который сразу же будет виден на экране телефона, будто он лежит на выбранной плоскости.

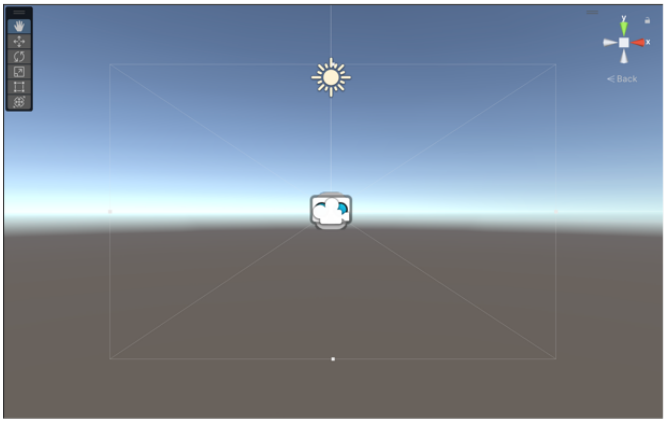

Рис. 3.3: Сцена Unity

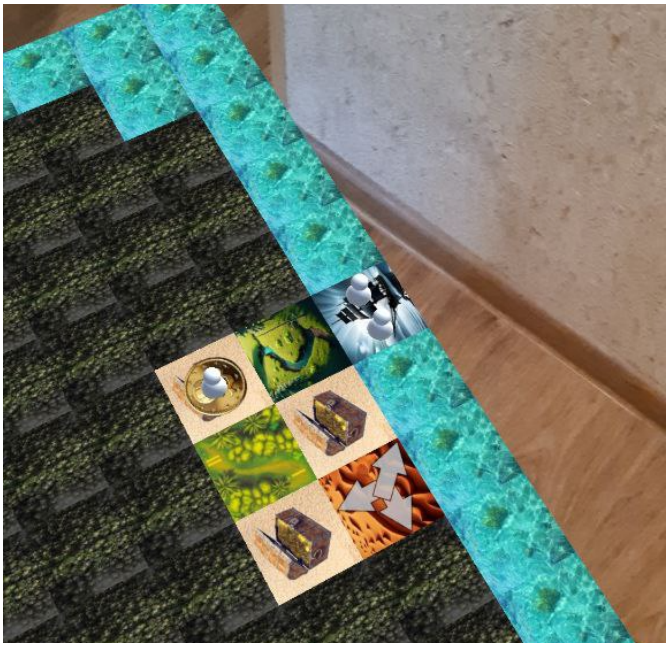

Рис. 3.4: Игровое поле в AR

3.3 Альтернативная версия игры

AR - довольно затратная технология, которую не все телефоны в обществе (а наш продукт расчитан на абсолютно любого человека) могут поддерживать.Также не хочется делать так, чтобы игроки зависили от их места положения. Поэтому в приложении был создан альтернативный режим игры - 3D верися ее, где все поле располагается на сцене и никак не взаимодействует с реальным миром. Данный режим облегчит игру не только

пользователям, но и нам - разработчикам, так как тестировать игру в AR режиме очень сложно: нужно собрать проект, перекинуть на телефон (так как для его работы обязана быть камера, и на эмултяторе поэтому не получалось запускать игру), тестировать и при наличии ошибки повторять все это заново - все это занимало очень много времени, что является непозволительной роскошью для нас. А в альтернативном режиме процесс пошел в гору, так как мы могли, не отходя от компьютеров, исправляеть все ошибки и сразу же тестировать новые версии.

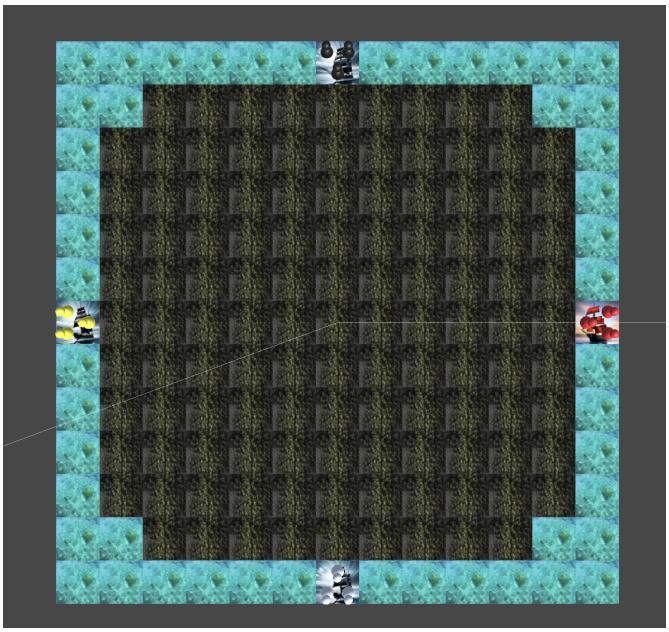

Рис. 3.5: Вид поля в альтернативной версии

Для удобного взаимодействия пользователя с игрой было добавлено несколько механик, помогающих игроку более удобно перемещаться по игровому полю, так как в AR моде игрок всегда может подойти поближе к полю или же обойти его для удобного для себя расположения. Первым нововведением для альтернативной версии стало перемещение камеры, используя палец игрока (классическая механика большинства мобильных игр). Для этого в игре контролируются касания пльцем игрового поля. Если палец докоснулся, то его позицию запоминают как стратовую, далее с частотой кадров фиксируется дальнейшие расположения пальца. Отталкиваясь от разницы между полученными координатами, камера двигается в нужно для пользователя напправлении. На рисунке ниже показано, как при движении пальца менялось положение камеры, используя разницу $(x, y) - (x0, y0)$.

Второй фишкой альтернативного режима является зум камеры - приближение или отдаллении ее относительно игрового поля. Механика довольно стандартна для мобильных игр - если два пальца отдаляются друг от друга, то камера отдаляется, иначе приближается к игровому столу. Реализация зума вытекает из реализации движения камеры, только на этот

раз отслеживается, что и второй палец коснулся экрана. Потом берется первчное расстояние между двух пальцев, потом оно сравнивается с расстоянием на текущий момент между ними. Если изменение пошло в положительную сторону, то это означает, что пальцы раздвинулись и камеру стоит приблизить. Таким же образом принимается решение ее приблизить.

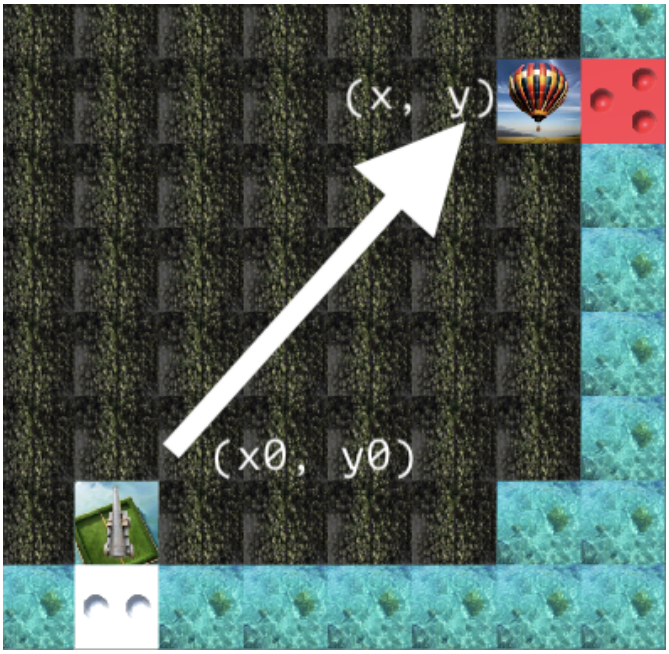

Рис. 3.6: Передвижение камеры

3.4 Мультиплеер

Одной из основных особенностей игры является её мультиплеерная составляющая, так как разработка ведётся на Unity, то необходимо выбрать подходящий фреймворк из Asset store для реализации многопользовательского режима, наш выбор пал на Photon Pun.

Так в PhotonNetwork существуют методы, которые позволяют создавать комнаты и подключаться к ним.

Также PhotonNetwork позволяет производить передачу данных между комнатами и синхронизировать состояние поля у игроков, через технологию Remote procedure call (RPC). Для этого в Photon Unity Networking определенно ключевое слово [PunRPC] над функциями в C#.

У данного метода есть небольшой недостаток, он состоит в том, что передать можно только примитивного типа и несколько стандартных Unity структур. Полный список допустимых данных находится по ссылке.

Для вызова функций с ключевым словом [PunRPC] в PhotonNetwork определён метод RPC, принимающий первым аргументвом название функции, вторым Target, определяющий

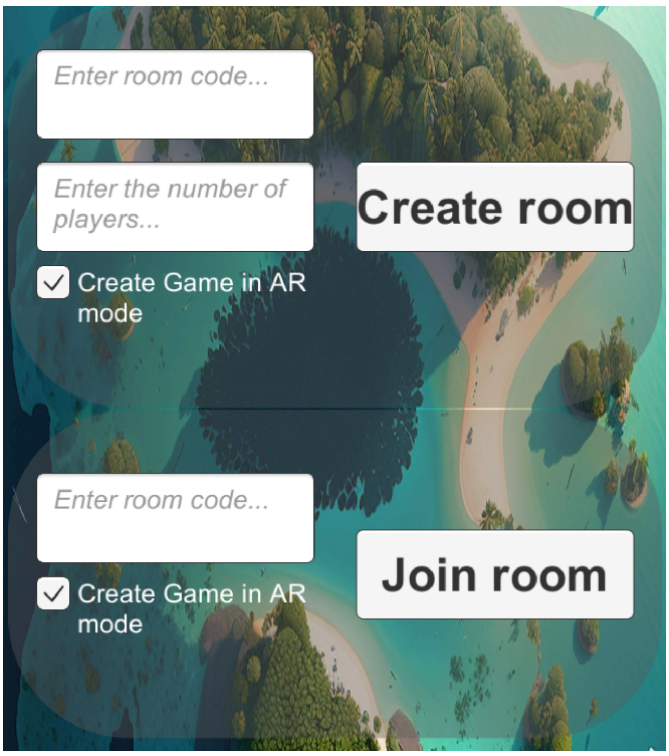

Рис. 3.7: Создание и подключение к комнате

```
[PunRPC]
Николай Некрасов *
public void DebugRpc(int x, int y)
{
    Debug.Log( message: "DebugRpcCalled");
    currGame.PlayingField[x, y].Open();
}
```

Рис. 3.8: Ключевое слово

кому будут переданны данные, а третьим аргументы вызываемой функции.

Игра начинается с главного меню, где игроку предлагается создать комнату по уникальному ключу или присоединиться к существующей. При создании или присоединению к комнате, благодаря PUN, происходит загрузка новой сцены, которая содержит интерфейс игры и игровые объекты. Все игровые данные, включая положение объектов, состояние игры и прочее, синхронизируются со всеми клиентами посредством сетевого взаимодействия, а после окончания игры все данные сохраняются на облачном сервере.

3.4.1 Сцена загрузки

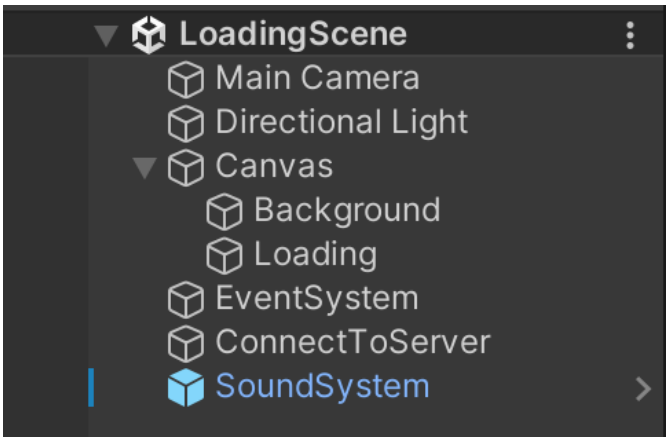

Рис. 3.9: Элементы сцены загрузки

Основным объектом этой сцены является "ConnectToServer который связан со скриптом "ConnectToServer.cs".

В этом скрипте происходит проверка устройства на подключение к сети, инициализация начальных настроек пользователя и загрузка сцены "Menu".

3.4.2 Сцена меню

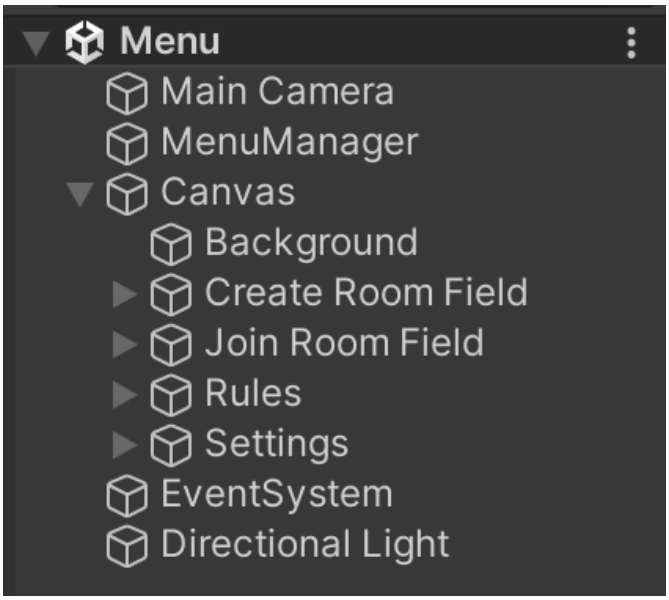

Рис. 3.10: Элементы сцены меню

На этой сцене присутсвует объект "MenuManager связанный со скриптом "MenuManager.cs а также поля для ввода, кнопки для создания и присоединения к комнате.

В данном скрипте реализованна логика кнопок создания и присоединения к комнате.

Так для создания комнаты необходимо заполнить поле "Enter room code... которое будет являться уникальным идентификатором комнаты, по которому другие игроки смогут к ней присоединяться и поле "Enter the number of players... которое задаёт максимальное количество игроков в комнате.

Комната создаётся с помощью метода PhotonNetwork.CreateRoom, первым аргументом которой является уникальный идентификатор комнаты, а вторым настройки комнаты, в настройках комнаты как раз и указывается максимальное количество игроков в комнате.

```
RoomOptions roomOptions = new RoomOptions();
roomOptions.MaxPlayers = (byte)playersNumb;
PhotonNetwork.CreateRoom(createInput.text, roomOptions);
```

Рис. 3.11: Создание комнаты

Для присоединения к комнате необходимо знать её уникальный идентификатор.

Присоединение к комнате происходит с помощью метода PhotonNetwork.JoinRoom, принимающий единственным аргументом уникальный идентификатор комнаты.

```
isAR = isARJoin.isOn;
PhotonNetwork.JoinRoom(joinInput.text);
```

Рис. 3.12: Присоединение к комнате

При создании или присоединения к комнате у пользователя загружается следующая сцена "Game"или "GameAR".

3.4.3 Game

На сцене "Game"и "GameAR"присутсвует объект "RpcConnector связанный со скриптом "RpcConnector.cs".

В данном скрипте происходит синхронизация объектов на поле между игроками с помощью функций помеченных ключевым словом [PunRPC].

3.5 Текстуры

При выборе текстур мы решили не использовать карточки из оригинальной игры, чтобы не было проблем с авторскими правами. Большинство карточек мы генерировали с помощью искусственного интелекта. Изначально для этого мы использовали Midjourney, как один из самых современных и популярных искусственных интеллектов для генерации, но позже мы столкнулись с проблемой, что в этой нейросети есть ограничение на количество

бесплатно сгенерированных картинок. Поэтому мы решили использовать бесплатный аналог BlueWillow, который по качеству картинок не уступает конкуренту. Остальные картинки приходилось делать в редакторе картинок или брать бесплатные стоковые. Например, было очень сложно объяснить искусственному интеллекту, что нужно сгенерировать несколько аналогичных картинок со стрелочками

Таблица 2 – Разделы, написанные в главе студентом

И. О. Фамилия	**Разделы**
Р. Р. Черномордин	Структуризация проекта, Augmented Reality, Альтернативаная версия игры
В. В. Ковешников	Текстуры
Н. Н. Некрасов	Мультиплеер

4 Шакал

4.1 Генерация поля

Карточки воды генерируются на краю поля. Остальные карточки расставляются случайныи образом с помощью самописного метода shuffle, т.к. в C# этого метода нет, но есть метод Random, с помощью которого карточки и расставляются. Псоле этого расставляются корабли игроков в определённое место на поле.

4.2 Игровая карточка

Игровая карточка - объект, с которым игрок взаимодействует посредством передвижения игровой фишки на игровом поле. Игровое поле - все игровые карточки, расположенные в определённом порядке: карточки, обозначающие воду, расположены с краю и окружают “остров”. Все остальные типы карточек замешиваются в случайном порядке (как и в оригинальной игре).

Необходимо было изучить документации о C# и скриптах в Unity [2-3]. Так замешивание карточек реализовано с помощью встроенного пространства имён “UnityEngine.Random”.

Все типы карточек реализованы как классы, которые наследуются от базового класса. Благодаря этому улучшается структурированность кода, а также позволяет в будущем легче добавлять новые типы карт. Каждая карточка имеет информацию об игровых фишках, которые находятся на ней, благодаря чему становится возможным контролировать ходы, а также особое поведение этих фишек. Также все карточки привязаны к своим игровым объ-

ектам на сцене, которые появляются после выбора места для игрового поля игроком, что позволяет менять их текстуры и добавлять новые игровые объекты на карточки (например, появление монет на карточке с сундуком, после её открытия в будущих версиях игры).

4.3 Виды карточек

Всего в игре реализовано 25 видов карточек (как и в оригинальной игре).

1 Пустая карточка – карточка, которая позволяет ходить на все карточки, кроме воды

2 Вода - карточка, которая позволяет ходить только на другие карточки воды или на корабль

3 Лошадь - ходит как в шахматах "буквой Г"

4 Пушка - "выстреливает"пиратом в конец игрового поля (в воду или в корабль)

5 Каннибал - убивает пирата, который встаёт на эту карточку

6 Крепость - карточка, на которой нельзя атаковать других пиратов, а также в неё нельзя проносить монеты

7 Шаманка - работает как крепость, а также даёт возможность воскрешать мёртвых пиратов из своей команды

8 Сундук - при первом открытии на нём появляются монеты

9 Лёд - продолжает ход который был сделан с предыдущей карточки

10 Крокодил - возвращает пирата на предыдущую карточку

11 Вертолёт - даёт возможно единоразово сходить на любую карточку на игровом поле

12 Воздушный шар - отправляет пирата на свой корабль

13 Ловушка - отнимает у пирата возможность ходить, пока его не освобят, придя на эту карточку

14 Ром - пират, наступивший на эту карточку, пропускает ход

А также 7 карточек со стрелками, с которых разрешено ходить только в указанном направлении. И 4 карточки вертушки, на которых надо сделать несколько ходов, чтобы выбраться из них

4.3.1 Корабль

Ещё в игре, есть отдельный объект, реализованный как карточка. У каждой команды есть свой корабль, который появляется в начале игры и служит для передвижения по воде, на другие клетки, а также на сторону противников корабль двигаться не может. Корабль двигается только когда на нём стоит хотя бы один пират. Вместе с кораблём передвигаются и все пираты, стоящие на нём. Слезть с корабля можно только по прямой, а залезть обратно можно и по диагонали. Также, попавшие в воду пираты, могут взабраться на свой корабль или умереть об вражеский. Всё как и в оригинальной игре.

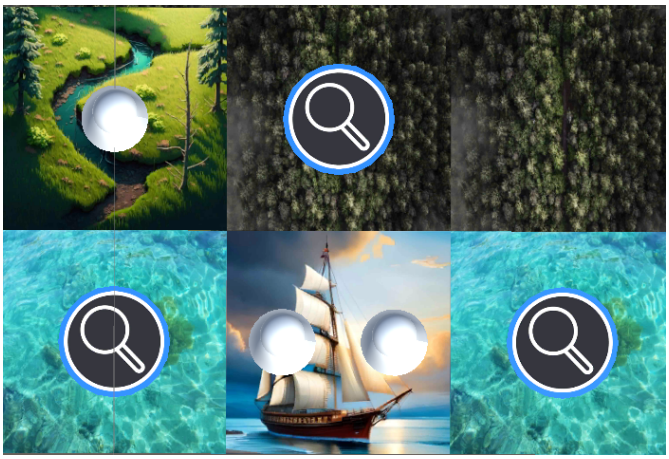

Рис. 4.1: Корабль белой команды

4.3.2 Уникальные свойства

Некоторые карточки обладают свойством "Поворот". При генерации они случайным образом поворачиваются на поле, из-за чего меняются их свойства. Так, например, у стрелок меняется направление в котором игрок обязан ходить пиратом, попавшим на эту карточку. Также этим свойством обладает пушка.

В игре есть несколько карточек, встав на которые, пират должен сразу сходить (именно этот, а не другой из его команды). Например, попавшись на лёд, игрок должен сразу же передвинуть своего пирата в том же направлении, в котором он сходил. Ещё таким свойством обладают стрелка, лошадь, пушка, шар, крокодил.

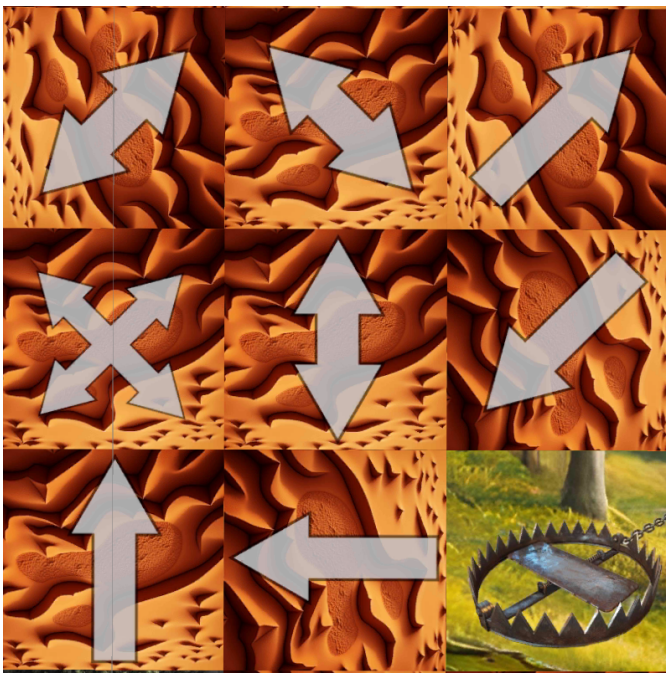

Рис. 4.2: Разные ориентации стрелок

4.3.3 Монеты

Как упоминалось ранее, при попадании на карточку с сундуком на ней генерируются нексколько монет. В зависимости от константного поля сундука, он генерирует определённое количество монет. Всего в игре:

Таблица 3 - Число сундуков с разным количеством монет

Количество сундуков	**Монет в сундуке**
5	1
5	2
3	3
2	4
1	5

Сами монеты можно перетаскивать на соседние открытые карточки с помощью кнопки, которая появляется, когда пират стоит на карточке с монетами. Перенеся монету на корабль, игрок получает её в свою копилку, содержимое которой отображается на экране.

Рис. 4.3: Монеты на карточках

4.3.4 Вертушки

Рис. 4.4: Клетка "Вертушка"

Основной принцип Вертушки в том, что она внутри себя содержит от двух до четырёх этап, которые игрок обязан пройти, чтобы сойти с клетки. У персонажа (см. раздел "Игровая фишка") еимеется два числа, обозначающих его положение на игровом поле - горизонталь и вертикаль. Для клетки "Вертушка"был создан еще один показатель - глубина, которая по умоляанию равна 1. При движении по данной клетке первые два показателя не должны

изменяться, пока клеточка не закончится, а глубина будет увеличиваться, каждый раз соответсвуя этапу клетки, и при взаимодействии разных вишек тоже будет учитываться. Может возникнуть случай, что несколько дружественных персонажей может оказаться на одном и том же этапе клетки, тогда они сливаются визуально в одного персонажа, а над ними просто появится цифра, показывающая их колличество.

4.4 Игровая фишка

Игровая фишка – объект, через который игрок будет управлять процессом. Она должна уметь ходить по полю, определяя куда ей дозволено, а куда – нет, перемещаться вместе с «золотой» монеткой, а также понимать, что на следующей клетке может быть враг и при переходе на нее отправлять всех соперников на их корабль (по правилам игры). Но прежде всего необходимо выбирать персонажа. Для этого используются лучи, выходящие из камеры, где коснулся пользователь (то есть если нажал на левый нижний угол, то луч выйдет из левого нижнего угла). Чтобы луч смог определить, что ему стоит учитывать и останавливаться при пересечении, а что игнорировать и «лететь» дальше, стоит определить специальный слой для фишки. У каждого слоя есть своя битовая маска, которую мы будем передавать, чтобы отслеживать пересечения с конкретными объектами. Определение совпадения между маской, которую мы ищем, и полученной из рассматриваемого объекта происходит через логическое «и», если результатом логического умножения является не ноль, тогда этот объект подходит нам, и на нем можно остановиться, в противном случае «летим» дальше. Так в искомую маску можно указать множество разных слоев, которые нам будут подходить. Испоьзуя эти же маски, можно реализовать множество механик, связанных с игровыми правилами. Так, попадая на карточку "Ром фишка становится неактивной на несколько ходов, а для того меняется слой её, чтобы у пользователя не было возможности взаимодействовать с ней и лучи проходили сквозь неё.

Таблица 4 – Битовые маски

Номер необходимого слоя	**Маска в двоичной системе счисления**	**Маска в десятичной системе счисления**
1	0010	2
3	1000	8
1 и 3	1010	10

Итак, луч пересекается с фишкой, опираясь на ее маску, и возвращает найденный объект в скрипт для дальнейшего использования. Фишка хранит в себе текущие координаты на

матрице карточек, откуда можно узнать какому типу клеточки соответствует та, на которой фишка стоит (так как у разных клеточек свои варианты перемещения могут быть), и какие типы у тех, на которые она может пойти. Это сделано для того, чтобы сохранялись базовые правила - фишка не может «спрыгнуть» в море с суши, и наоборот не может «забраться» на берег из воды. Также при просмотре следующей возможной клеточки можно понять, есть ли на ней противник, и если сеть, то подсветить текущее перемещение как «атака» для игрока. Все возможные перемещение показываются пользователю синим (или же красным при атаке) полупрозрачным кружком со своим личным слоем в Unity.

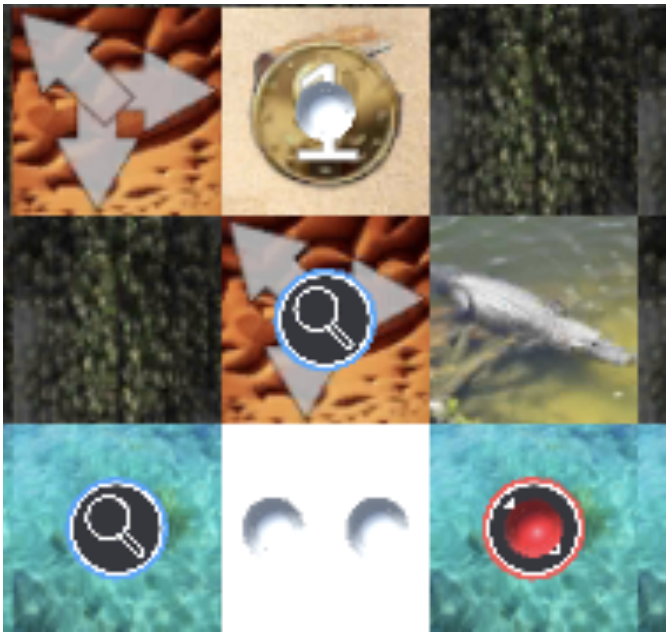

Рис. 4.5: Ход белой фишки, стоящей на корабле

Для совершения действия игрок должен нажать на выбранный вариант. Может произойти случай, что фишка захочет наступить на клетку, на которой уже кто-то стоит, тогда кружок перемещения будет находиться под фишкой и попасть в него пальцем будет почти нереально (или очень тяжело). Для устранения такого неудобства и используются маски, чтобы при выборе хода «лучи» могли чувствовать только круги действий и выбранную фишку, при повторном нажатии на которую кружки исчезали бы. Так мы позицию в маске слоев, соответствующую персонажу, зануляем, а для кружка перемещения устанавливаем единичку. Дальше изменяются позиции, открывается клетка, если она закрыта, происходят характерные для нее действия, и ход завершается.

Таблица 5 – Разделы, написанные в главе студентом

И. О. Фамилия	**Разделы**
Р. Р. Черномордин	Игровая фишка, Виды карточек: Вертушки
В. В. Ковешников	Генерация поля, Игровая карточка, Виды карточек
Н. Н. Некрасов	Корабль

5 User Interface

5.1 Меню

При заупуске приложения пользователи не должны сразу сталкиваться с игрой, а там уже что будет. Им должен быть дан выбор: в какой режим пойти (AR или обычный), создать свою команату или же присоединиться к уже существующей. Все это должно решаться за долго до вступления в игру. Для этого мы создали удобное панель меню, где все интуитивно понятно. Здесь игроки смогут создать свой сервер, указав код доступа к нему и максимульное колличество людей в нем, и выбрать в каком режиме именно они будут играть (остальные могут выбрать другой, какой будет удобней им). Также можно и присоедениться к уже существующей.

Музыка - один из столпов современной игрой индустрии, музыка способна повлиять на чувства людей или же помочь им нескучно провечти время. Настольные игры - не самое быстро врепрепровождения, для этого мы добавили в игру музыкальное сопровождение, чтобы пользователи не сидели в тишине. В разделе S̈ettings(настройки) игроки смогут откалибровать громкость музыки под тот уровень, который им будет приятен, или же вообще выбрать другу композицию.

Шакал - несложная игра, но в нее не получится играть, если не знать ряд обязательных правил. Чтобы пользователи не искали по интернету их, мы создали раздел, где они смогут прочитать описания базовых механик игры и всех типов карточек поля. Но так как всего правил очень много и не хочется выстраивать их в динное полотно, было принято решение разбить их на две взаимодополняющие части.

Рис. 5.1: Меню

5.2 Специальные кнопки

При определённых игровых событиях на экране у игрока появляются кнопки, которые выполняют действия при нажатии на них. Когда игрок встаёт на карточку, на которой лежит монета, то на экране появляется кнопка, при нажатии на которую пират поднимает монету, которую можно перенести на соседнюю клетку, а также появляется другая кнопка, позволяющая положить монету, которая нужна на случай, если игрок передумает ходить этим пиратом с монеткой.

Когда игрок не имеет ходов (например, если попался в ловушку), то появляется кнопка, которая позволяет убить своего пирата.

Если в команде пирата есть мёртвые сокомандники и он стоит на карточке с шаманкой, то появляется кнопка, повзволяющая воскресить его и закончить ход. Воскрешённый пират появляется на карточке с шаманкой

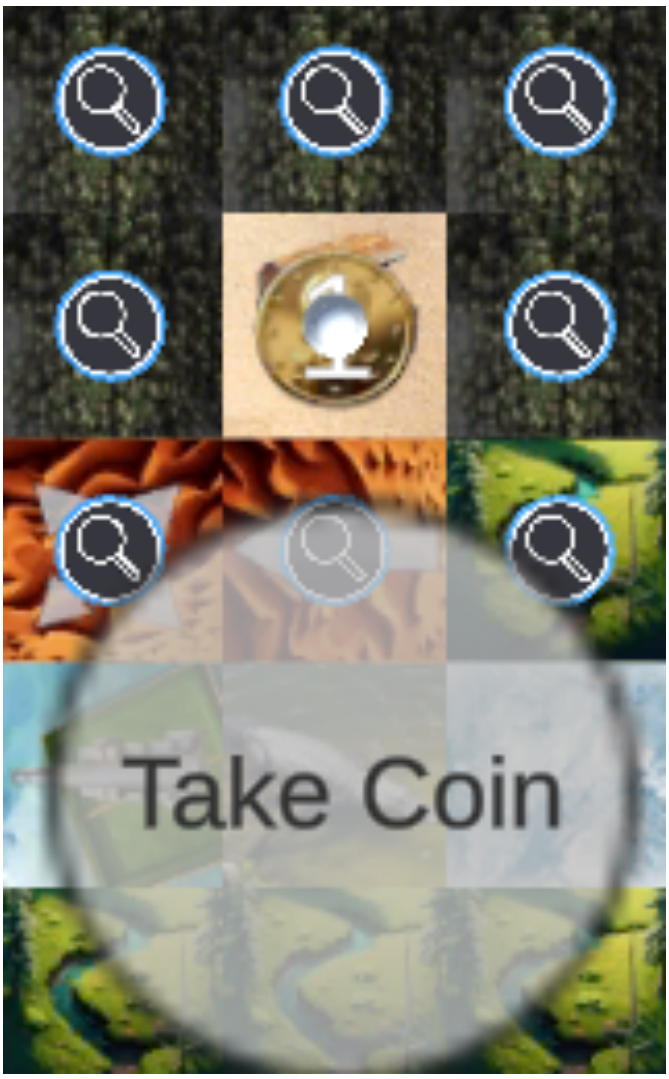

Рис. 5.2: Кнопка взятия монеты

5.3 В игре

В игре присутствует музыка. Чтобы не выходить каждый раз назад в игровое меню ради изменения громкости мелодии, было создано вспшлывающее окно настроек прямо в игре, где пользователи так же могут изменить звучиние и произвежение, а так же выйти из самой игры.

Также во время игры на экране игрока есть две надписи. Одна отображает, какая команда сейчас ходит, а вторая - сколько монет принёс на свой корабль игрок за всю игру

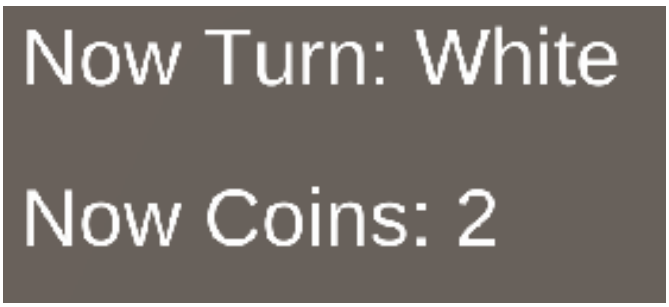

Рис. 5.3: Надписи в игре

Таблица 6 – Разделы, написанные в главе студентом

И. О. Фамилия	**Разделы**
Р. Р. Черномордин	Меню
В. В. Ковешников	Специальные кнопки
Н. Н. Некрасов	В игре

6 Заключение

Итогом нашего проекта является готовое приложение для Android под названием "Шакал AR в котором реализованны:

1 Все основные механики исходной настольной игры - поведение фишек и игровых карточек

2 Два режима игры - AR и обычная в игровом мире

3 Создан удобный User Interface с настрокой звука в игре и выхода из нее

4 Создано первоначальное меню, где пользователь сможет создать игровую сессию с определенным ее кодом для входа в нее другим пользователем и их максимальное колличество, сможет поддключаться по коду к игре, сможет прочесть все правила исходной настольной и информацию по карточкам и настроить звук в игре

5 Создано мультиплеерное составляющие игры

6 Сгенерированы картинки для игровых моделей

Все, что было выше описано, является, безусловно, положительными чертами нашего продукта, особенно хочется выделить принятия решения о создании альтернативной версии игры - без AR режима, так как это очень сильно способствовало разработке итоговой

версии. Из минусов можно выделить непродвинутый дизайн меню: кнопки, холст с правилами. Но нельзя считать, что проекту некуда дальше развиваться. В дальнейшем планируется адаптировать приложение под IOS системы, дать возможность игрокам играть на одном телефоне, внедрить внутриигровой чат, реализовать все курпные дополнения к оригинальной настольной игре, и самое главное - улучшить визуальное состовляющие нашей игры, а точнее главного меню и все разделов, которые из него вытекают.

Наш проект уникален на игровом рынке, так как нигде нет точного аналога нашего продукта, ведь "Шакал AR"поддерживает новизну технологии AR, которую сейчас многие производители обходят стороной, находясь в то же время в секции настольных игр, которые сейчас не так популярны в крупных компаниях, так как этот жанр нельзя назвать прибыльным, когда можно создать обычный шутер и продовать внутриигровую атрибутику за реальные деньги. Наша игра не гонится за деньгами и славой, она хочет привнести спокойствие и умиротворение, утолить голод путешествия по необитаемым островам, развить стратегические способности и просто дать провести замечательно время двум или четырём друзьям или наоборот незнакомым людям. Мы хотим вернуть в игровую индустрию жанр "настольные игры покахав через AR сисему, что это так же может быть интересно и необычно на фоне обычных мобильных приложений!

Список литературы

[1] Официальная документация Google по ARCore.

[2] Официальная документация по C#

[3] Официальная документация по скриптам в Unity

[4] Официальная документация Photon.

[5] Официальный набор Photon в магазине Unity.