TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки: 09.03.04 «Программная инженерия»
Отделение информационных технологий

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Разработка многопользовательского игрового приложения в жанре «пошаговая стратегия»

УДК 004.451:004.451.7.031.43

Студент

Группа	ФИО	Подпись	Дата
8К71	Сиротин Алексей Алексеевич		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Савельев Алексей Олегович	к.т.н.		

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН ШБИП	Маланина Вероника Анатольевна	Кандидат экономических наук		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Черемискина Мария Сергеевна	-		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Чердынцев Евгений Сергеевич	к.т.н.		

Томск – 2021г.

Планируемые результаты обучения по направлению 09.03.04 «Программная инженерия»

Код результатов	**Результат обучения** **(выпускник должен быть готов)**	**Требования ФГОС, критерии АИОР**
Р1	Применять базовые и специальные естественнонаучные и математические знания в области информатики и вычислительной техники, достаточные для комплексной инженерной деятельности.	Требования ФГОС (ОПК-1,2,3, ПК-4, 5, 6), критерий 5 АИОР (п. 1.1)
Р2	Применять базовые и специальные знания в области современных информационных технологий для решения инженерных задач.	Требования ФГОС (ОПК-3, 4, ПК-1, 2, 9), критерий 5 АИОР (п.1.1, 1.2)
Р3	Ставить и решать задачи комплексного анализа, связанные с созданием аппаратно-программных средств информационных и автоматизированных систем, с использованием базовых и специальных знаний, современных аналитических методов и моделей.	Требования ФГОС (ОК-1, 6, ПК-2, 4, 5), критерий 5 АИОР (п. 1.2)
Р4	Разрабатывать программные и аппаратные средства (системы, устройства, блоки, программы, базы данных и т. п.) в соответствии с техническим заданием и с использованием средств автоматизации проектирования.	Требования ФГОС (ОПК-2, 3, ПК-3, 4, 5), критерий 5 АИОР (п. 1.3)
Р5	Проводить теоретические и экспериментальные исследования, включающие поиск и изучение необходимой научно-технической информации, математическое моделирование, проведение эксперимента, анализ и интерпретация полученных данных, в области создания аппаратных и программных средств информационных и автоматизированных систем.	Требования ФГОС (ОПК-4, ПК-6, 7), критерий 5 АИОР (п.1.4)
Р6	Внедрять, эксплуатировать и обслуживать современные программно-аппаратные комплексы, обеспечивать их высокую эффективность, соблюдать правила охраны здоровья, безопасность труда, выполнять требования по защите окружающей среды.	Требования ФГОС (ОПК-3, ПК-7, 8, 9), критерий 5 АИОР (п. 1.5)

	Универсальные компетенции	
Р7	Использовать базовые и специальные знания в области проектного менеджмента для ведения комплексной инженерной деятельности.	Требования ФГОС (ОКП-1, 4, ПК-1, 6, 7), критерий 5 АИОР (п. 2.1)
Р8	Владеть иностранным языком на уровне, позволяющем работать в иноязычной среде, разрабатывать документацию, презентовать и защищать результаты комплексной инженерной деятельности.	Требования ФГОС (ОК-5), критерий 5 АИОР (п. 2.2)
Р9	Эффективно работать индивидуально и в качестве члена группы, состоящей из специалистов различных направлений и квалификаций, демонстрировать ответственность за результаты работы и готовность следовать корпоративной культуре организации.	Требования ФГОС (ОК-6), критерий 5 АИОР (п. 2.3, 2.4)
Р10	Демонстрировать знания правовых, социальных, экономических и культурных аспектов комплексной инженерной деятельности.	Требования ФГОС (ОК2, 3, 4), критерий 5 АИОР (п. 2.5)
Р11	Демонстрировать способность к самостоятельному обучению в течение всей жизни и непрерывному самосовершенствованию в инженерной профессии.	Требования ФГОС (ОК-7), критерий 5 АИОР (п. 2.6)

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки: 09.03.04 «Программная инженерия»
Отделение информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____ Е.С. Чердынцев___
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы В форме:

бакалаврской работы

Студенту:

Группа	**ФИО**
8К71	Сиротину Алексею Алексеевичу

Тема работы:

Разработка многопользовательского игрового приложения в жанре «пошаговая стратегия»	
Утверждена приказом директора (дата, номер)	

Срок сдачи студентом выполненной работы:	11.06.2020

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе *(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к особенностям функционирования (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.).*	Объект исследования – игровое приложение в жанре «пошаговая стратегия», предназначенное для массового использования пользователями торговых игровых площадок. Режим работы: непрерывный; Особые требования: в качестве инструмента разработки использовать игровой движок Unity; игровые модели должны быть выполнены в воксельной графике.
Перечень подлежащих исследованию, проектированию и разработке вопросов *(аналитический обзор по литературным источникам с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования, конструирования; содержание процедуры исследования, проектирования, конструирования; обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе).*	1. Исследование предметной области 2. Аналитический обзор существующих систем многопользовательского взаимодействия 3. Проектирование игрового приложения в жанре «пошаговая стратегия» 4. Разработка игрового приложения в жанре «пошаговая стратегия» 5. Финансовый менеджмент 6. Социальная ответственность
Перечень графического материала *(с точным указанием обязательных чертежей)*	1. Диаграмма вариантов использования 2. Диаграмма классов 3. Диаграмма последовательностей 4. Диаграмма состояний

Консультанты по разделам выпускной квалификационной работы
(с указанием разделов)

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Маланина Вероника Анатольевна
Социальная ответственность	Черемискина Мария Сергеевна

Названия разделов, которые должны быть написаны на русском и иностранном языках:

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	1.03.2020

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Савельев Алексей Олегович	к.т.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8К71	Сиротин Алексей Алексеевич		

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки: 09.03.04 «Программная инженерия»
Уровень образования бакалавриат
Отделение информационных технологий
Период выполнения осенний / весенний семестр 2019 /2020 учебного года

Форма представления работы:

бакалаврская работа

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	16.06.2020

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
16.06.2020	*Раздел 1.Исследование предметной области*	*20*
16.06.2020	*Раздел 2. Разработка игрового приложения*	*40*
16.06.2020	*Раздел 3. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение.*	*20*
16.06.2020	*Раздел 4. Социальная ответственность*	*20*

СОСТАВИЛ:
Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Савельев Алексей Олегович	к.т.н.		

СОГЛАСОВАНО:
Руководитель ООП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР ТПУ	Чердынцев Евгений Сергеевич	к.т.н.		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8К71	Сиротину Алексею Алексеевичу

Школа	ИШИТР	**Отделение школы (НОЦ)**	ОИТ
Уровень образования	бакалавриат	**Направление/специальность**	09.03.04 Программная инженерия

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. Стоимость ресурсов научно-исследовательского проекта: материально-технических, энергетических, финансовых, информационных и человеческих	Бюджет проекта – не более 250 тыс. рублей, в т. ч. Затраты на оплату труда – 127788,2 руб. Отчисления во внебюджетные фонды – 38592,03 руб. Накладные расходы составили 29207,24 руб.
2. Нормы и нормативы расходования ресурсов	Значение показателя интегральной ресурсоэффективности разработки – не менее 4 баллов из 5 Интегральный финансовый показатель разработки– не более 1,00 Интегральный показатель эффективности – не менее 4,4
3. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	Коэффициент отчислений на уплату во внебюджетные фонды – 30,2% Коэффициент накладных расходов - 16%.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения	-Анализ конкурентных технических решений
2. Планирование и формирование бюджета научных исследований	Формирование плана и графика разработки: - определение структуры работ; - определение трудоемкости работ; Формирование бюджета затрат на научное исследование: - материальные затраты; - затраты на специальное оборудование; - заработная плата (основная и дополнительная); - отчисления на социальные цели; - накладные расходы.
3. Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования	- Определение потенциального эффекта исследования
Перечень графического материала *(с точным указанием обязательных чертежей)***:**	
1. Оценочная карта конкурентных технических решений *2. Матрица SWOT* *3. Расчет бюджета затрат*	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН ШБИП	Маланина В.А.	Кандидат экономических наук		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8К71	Сиротин Алексей Алексеевич		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8К71	Сиротин Алексей Алексеевич

Школа	Инженерная школа информационных технологий и робототехники	Отделение (НОЦ)	Отделение информационных технологий
Уровень образования	Бакалавриат	Направление/специальность	09.03.04 «Программная инженерия»

Тема ВКР:

Разработка многопользовательского игрового приложения в жанре «пошаговая стратегия»	
Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	Объект исследования: данные о работе многопользовательского режима. Область применения: многопользовательское игровое приложение.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности: – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	ГОСТ 12.2.032-78. Рабочее место при выполнении работ сидя. ГОСТ 22269-73. Рабочее место оператора. Взаимное расположение элементов рабочего места. ТК РФ Глава 36. Обеспечение прав работников на охрану труда ТК РФ Глава 34. Требования охраны труда и т.п.
2. Производственная безопасность: 2.1. Анализ выявленных вредных и опасных факторов 2.2. Обоснование мероприятий по снижению воздействия	Вредные факторы: -недостаток необходимого естественного освещения; -недостаток необходимого искусственного освещения; -повышенная пульсация светового потока; -отклонение показателей микроклимата и т.п. Опасные факторы: -повышенное значение напряжения в электрической цепи; -замыкание и т.п.
3. Экологическая безопасность:	Атмосфера: ртутное загрязнение и т.п. Гидросфера: бытовой мусор, ртутное загрязнение и т.п. Литосфера: бытовой мусор, ртутное загрязнение и т.п.

4. Безопасность в чрезвычайных ситуациях:	Возможные ЧС: пожары, терроризм, пандемия, аварии на автономных электростанциях с долговременным перерывом элект-роснабжения всех потребителей Наиболее типичная ЧС: пожар.

Дата выдачи задания для раздела по линейному графику	28.02.2021

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Черемискина Мария Сергеевна	-		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8К71	Сиротин Алексей Алексеевич		

Реферат

Выпускная квалификационная работа содержит 115 страниц, 32 рисунка, 16 таблиц, 18 источников.

Ключевые слова: игровое приложение, игровой движок, проектирование, разработка, жанр, моделирование, игровой процесс.

Объектом разработки является трехмерное многопользовательское игровое приложение в жанре пошаговая стратегия.

Цель работы – создание программного обеспечения для расширения рынка игровой индустрии и популяризации жанра пошаговая стратегия.

В результате исследования данная система спроектирована и реализована в полном объёме.

Список терминов и сокращений

Фреймворк – заготовки, шаблоны для программной платформы, определяющие структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта.

UML – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес процессов, системного проектирования и отображения организационных структур.

Git – распределенная система управления версиями.

UI (user interface) – интерфейс пользователя, описывает объекты управления внутри приложения.

UX (user experience) – опыт получаемый пользователем во время использования приложения.

Игровой движок – базовое программное обеспечение, предназначенное для создания компьютерных игр.

Gamedev – направление в разработке программного обеспечения, направленное на создание игровых приложений.

Содержание

Введение

Игровая индустрия – это сектор, связанный с разработкой, продвижением и продажей компьютерных игр. В неё входит большое количество специальностей, по которым работают тысячи человек по всему миру.

Рынок компьютерных игр является одним из крупнейших на данный момент и с каждым годом становится всё крупнее.

Компьютерная игра – это программное обеспечение, созданное для организации игрового процесса и удовлетворения таких потребностей пользователя, как досуг и коммуникация с другими пользователями (в случае многопользовательских игр).

Данная работа посвящена разработке такой игры, как Jackal. Это многопользовательская трехмерная компьютерная игра в жанре пошаговая стратегия.

В данной работе будут отражены такие этапы создания программного обеспечения, как анализ предметной области, проектирование и разработка игрового приложения, а также будет оценена экономическая целесообразность проекта.

Целью разрабатываемого приложения является создание работоспособного приложение, позволяющее создавать кратковременные игровые сессии между несколькими пользователями, а также настройка игрового взаимодействия между ними.

1. Анализ предметной области

1.1. Описание рынка игровой индустрии

Рынок компьютерных игр является одним из крупнейших на данный момент. В 2020-м издание Gamesindustry.biz подвело итоги года, согласно которым к концу 2020-го года объем продаж вырос на 20% по сравнению с 2019 и составил 175.9 млрд. долларов. Значимость компьютерных игр подчеркивает такое явление, как геймификация, которая представляет собой внедрение концепции игрового мышления и динамики игр в различные области человеческой деятельности для вовлечения аудитории и решения задач. Примерами приложений, использующих геймификацию могут служить Duolingo, NRC, Re-mission и др.

Помимо всего прочего, развитие игровой индустрии способствует развитию самих технологий в принципе. Создаются новые, более совершенные видеокарты, мощные средства для работы с 3D графикой и анимацией, физические движки и т.д.

Однако не все игры вносят большой вклад в развитие экономики и другие отросли жизнедеятельности. Существует множество разработчиков игр, которые не обладают большим штабом сотрудников, а иногда и вовсе работают одни. Эти игры не отличаются дорогостоящей качественной графикой и звуковым сопровождением, как игры популярных компании. Их называют инди-игры. Однако они несут ценность в виде интересных игровых механик или увлекательного сюжета, которые способны завлечь пользователей. Даже такие небольшие проекты приносят издателям достаточные для жизни доходы.

Для среднестатистического пользователя экономическая составляющая видеоигр не несёт большой ценности, его интересует исключительно игровой процесс.

В публикации от 2 мая 2017 года сообществом Gamedev, совместно с СПбГУ описали положительное влияние видеоигр на развитие человека. В частности:

- Ряд исследований показывает, что игры способствуют развитию познавательных навыков.
- Хорошие игры, коммерческие и обучающие, заставляют системно мыслить. Они реагируют на каждый шаг геймера — ему приходится постигать всю систему, чтобы выиграть.
- Гейминг может помочь понять, к чему склонен ребёнок. Жанровые предпочтения напрямую связаны со стилем обучения человека. Можно понять, как лучше подавать ему обучающие материалы.
- Игры связаны с креативностью. Геймеры регулярно применяют нестандартные способы решения задач. Однако неясно, развивают ли они творческое мышление, или творческие люди предпочитают игры.
- Видеоигры – идеальная тренировочная площадка для формирования менталитета роста: человек начинает понимать, что возможности развиваются пошагово, а не мгновенно.
- Игры – один из наиболее ярких источников положительных эмоций. Кроме того, геймеры часто описывают своё состояние во время игры как «потоковое».
- Многопользовательские игры воспитывают социальные навыки. Игроки принимают решения, предполагая, кому можно, а кому нельзя доверять, как стать лидером, управлять группой. В то же время, однозначная связь многопользовательских проектов с социальностью игроков пока изучается.
- В современных публикациях всё чаще упоминается геймификация в медицине и образовании. «Серьёзные игры» (serious games) прежде всего не

развлекают, а решают иные задачи, например, повышают мотивацию пациентов к заботе о своем здоровье и ускоряют процесс выздоровления.

- Также в ряде исследований людей пожилого возраста показано, что компьютерные игры, особенно активные, замедляют негативные процессы, связанные с возрастом: угасание когнитивных функций, проблемы с социализацией.

2. Разработка игрового приложения

2.1. Технология разработки игрового приложения

Под технологией разработки понимают оптимальный способ ведения процесса разработки, который при определенных условиях обеспечит получение конечного продукта с заранее заданными свойствами.

В ходе исследования был структурирован существующий материал и на его основе создана технология разработки игровых приложений. Разработка приложения осуществляется по следующему алгоритму:

1. Выбор жанра игрового приложения.

На этом этапе важно сформировать чёткое представление о том, к какому жанру относится разрабатываемое приложение и какие механики и концепции игровой разработки ему присущи. Это позволяет поставить чёткие цели и задачи при создании приложения, так как многие игровые механики имеют чёткие правила и примеры реализации. Игра может включать в себя несколько жанров и необязательно должна соответствовать всем правилам, приписанным к данной жанровой составляющей, порой подобные игры порождают собственные поджанры.

2. Проектирование игрового приложения.

Под структурой проекта понимается совокупность факторов, характеризующих игровой процесс, такие как игровые механики, игровые объекты (персонажи, объекты окружения, элементы интерфейса и пр.), система уровней и сюжетная составляющая. Так как разрабатываемое приложение не

имеет сюжета, то это позволяет опустить разработку системы уровней и сюжетной составляющей.

3. Выбор технологий и средств разработки.

Любая разработка программного обеспечения осуществляется с использованием той или иной технологии, или совокупности технологий. Таким образом для разработки игрового приложения требуется выбор игрового движка, графического редактора для создания игровых объектов, IDE – для написания программного кода.

4. Создание игрового приложения.

На данном этапе осуществляется непосредственная разработка игрового приложения, создаются модели, игровые объекты, пишется программный код, добавляется музыка, анимация. Как правило, это самый крупный этап разработки, в ходе которого могут возникнуть непредвиденные трудности и необходимость в редактировании технического задания.

5. Тестирование.

Этап, в ходе которого в разрабатываемом приложении ищутся ошибки и неисправности, которые возникли на этапе разработки. Тестирование бывает ручным и автоматическим, при ручном тестировании, человек запускает игровое приложение и вручную находит ошибки или поведения, не предусмотренные разработчиком. При автоматическом же тестировании, пишутся модульные тесты, которые проверяют корректность выполнения различных модулей игрового приложения.

6. Эксплуатация.

Этап, на котором готовое игровое приложение публикуется в сети или какой-либо игровой площадке.

Реализация представленной технологии позволяет структурировать процесс разработки игрового приложения, тем самым сокращая временные затраты, необходимые для его создания.

2.2. Определение жанра игрового приложения

Жанр – это совокупность характеристик и принципов, применяемых в разработке игрового приложения, формирующие общие тенденции и создающие узнаваемость этих тенденций в играх, относящихся к одной жанровой принадлежности.

Список общеизвестных жанров включает в себя:

- Аркады. Компьютерные игры с примитивным игровым процессов, где управляемые игроком персонаж выполняет тривиальные действия, такие как бег и прыжки.
- Платформер. Жанр, основная игровая составляющая которого направлена на передвижение по платформам, лестницам, сбору предметов, как правило для него характерен боковой вид камеры и имеет двумерную графическую составляющую.
- Приключение. Игра, в которой хорошо проработанный сюжет переплетается с многообразием игровых локаций, исследуя которые, игрок продвигается по сюжету, раскрывая всё больше игровых деталей.
- Казуальные игра. Игры, предназначенные для широкого круга пользователей, имеющие тривиальные и интуитивно-понятные игровые правила. Как правило это небольшие игровые приложения, рассчитанные на пользователей не обладающими игровыми навыками.
- Шутер. Жанр видеоигр, основная направленность в котором направлено на стрельбу из различного рода оружия. Камера, как правило, имеет вид от первого лица или от третьего лица сзади. Зачастую такие игры являются многопользовательскими.
- Экшн. Игры, в которых важна скорость реакции игрока и умение быстро и правильно принимать тактические решения для достижения победы.
- Стратегия. Жанр видеоигр, который подразумевает использование пользователями стратегического мышления для достижения поставленной

задачи, и оно противопоставляется быстрым действиям и реакции, которые, как правило, не обязательны для успеха в таких играх.

- Спортивные игры. Игры, реализующие различные виды спорта.
- Головоломки. Игры, основной игровой процесс в которых направлен на решение различного рода логических и другого рода задач.
- Симуляторы. Жанр, который имитирует какую-либо сферу человеческой деятельности, как правило не имеет сюжета и прохождения.
- Ролевая. Жанр, в котором пользователь управляет персонажем или группой персонажей, обладающей определенными набором навыков и умений.
- Песочницы. Игры, в которых перед игроком не ставятся определённые цели, он сам ставит перед собой задачи и осуществляет их, как правило в таких играх открытый мир, что означает, что все игровые локации открыты перед игроком и у него нет ограничений на передвижение по миру.

При анализе вышеперечисленных жанров было выявлено, что разрабатываемое приложение относится к жанру стратегия.

Стратегия – это жанр видеоигр, который подразумевает использование пользователями стратегического мышления для достижения поставленной задачи, и оно противопоставляется быстрым действиям и реакции, которые, как правило, не обязательны для успеха в таких играх. Стратегии имеют множество разновидностей, они могут классифицироваться по ходу времени и на основе игрового процесса.

По ходу времени стратегии делятся на:

- Пошаговые стратегии
- RTS стратегии (Стратегии в реальном времени)
- Гибридные (Пошаговые в реальном времени) стратегии

По игровому процессу стратегии делятся на:

- Классические RTS

- Тактические стратегии
- Экономические стратегии
- Экономические онлайн игры
- Варгеймы
- 4X-стратегии
- Глобальные стратегии
- MOBA
- Карточные стратегии

Пошаговая стратегия – это поджанр компьютерных стратегических игр, в которых игровой процесс состоит из последовательности фиксированных моментов времени, именуемых ходами (или шагами), во время которых игроки совершают свои действия.

По игровому процессу, разрабатываемое приложение относится к тактическим стратегиям. В тактических стратегиях игроку не приходится заниматься менеджментом базы или добычей ресурсов, основной упор сделан на тактике ведения «боя» в условиях заранее заданного количества ресурсов.

2.3. Проектирование игрового приложения

Разрабатываемое игровое приложение основывается на одноимённой настольной игре, поэтому техническое задание и проектирование игрового приложения во многом строится за счёт основных игровых концепций оригинальной настольной игры «Jackal».

2.3.1. Техническое задание

1. Общие сведения.

1.1. Наименование организации-заказчика: Томский политехнический университет.

1.2. Название продукта разработки: Jackal.

1.3. Назначение продукта: игровое приложение.

1.4. Плановые сроки начала и окончания работы.

Таблица 1 Этапы выполнения

№	*Этапы выполнения*	*Сроки выполнения этапа*	
		Начало периода	*Конец периода*
1.	Проектирование	24.09.20	1.11.20
2.	Разработка	2.11.20	19.05.21
3.	Тестирование	16.11.20	19.05.21
4.	Составление документации	1.05.21	2.06.21

2. Требования к продукту разработки

2.1. Аппаратные требования:

Таблица 2 Аппаратные требования

	Минимальные	Рекомендуемые
Платформа	Windows XP	Windows 7
Процессор	Core 2 Duo	2.4 GHz Quad Core 2.0 (or higher)
Оперативная память	2 GB ОЗУ	8 GB ОЗУ
Видеокарта	Discreet video card	ATI Radeon HD-Series 4650
Место на диске	77 MB	77 MB
Звуковая карта	Да	Да

2.2. Нефункциональные требования

- Обработка персональных данных пользователя не должна нарушать закон «О персональных данных» РФ;
- Приложение должно быть написано на игровом движке Unity;
- Приложение должно поддерживать онлайн до 20-ти человек;
- Приложение должно быть доступно для скачивания на торговой площадке Steam;
- Регистрация в приложении должна осуществляться через торговую площадку Steam.
- Приложение должно поддерживаться ОС Windows;

- Количество пользователей в одной игровой сессии должно находиться в диапазоне 2-4;
- Приложение должно обеспечивать не менее 30FPS при минимальных настройках графики, для минимальных системных характеристик.

2.3. Функциональные требования

- Возможность создать игровую комнату с названием
- Возможность добавить пароль к комнате
- Возможность подключиться к комнате с паролем и без пароля
- Возможность подключиться к случайной комнате без пароля. В случае, если таких комнат нет, то создать новую комнату со случайным названием.
- Возможность изменить параметры игрового поля, находясь в лобби комнаты, если пользователь является создателем комнаты
- Возможность изменять своё состояние готовности (готов/ не готов) в лобби комнаты
- Должна быть реализована пошаговая система. (см. определение пошаговая система)
- Возможность выбрать стартовую позицию объекта «Корабль» на ячейках «Море» в начале матча.
- Возможность перемещать объекты пользователя «Корабль», «Пират» на соседние ячейки игрового поля.
- Возможность перемещать объект «Пират» с объектом «Монета»
- Должно быть реализовано условие победы (Побеждает тот игрок, который перенёс на свой корабль больше всего монет)
- Возможность выйти из игрового приложения (завершение игрового процесса)
- Возможность перемещения камеры в пространстве (см. 2.5.4.6. RTS камера)

2.3.2. Описание вариантов использования игрового приложения

Игровое приложение предоставляет пользователям обширный список возможностей:

1. Авторизация в приложении;
2. Настройка игровой сессии;
3. Настройки видео и аудио;
4. Игровой процесс.

Подробный список пользовательских возможностей внутри приложения представлен на рисунке 1.

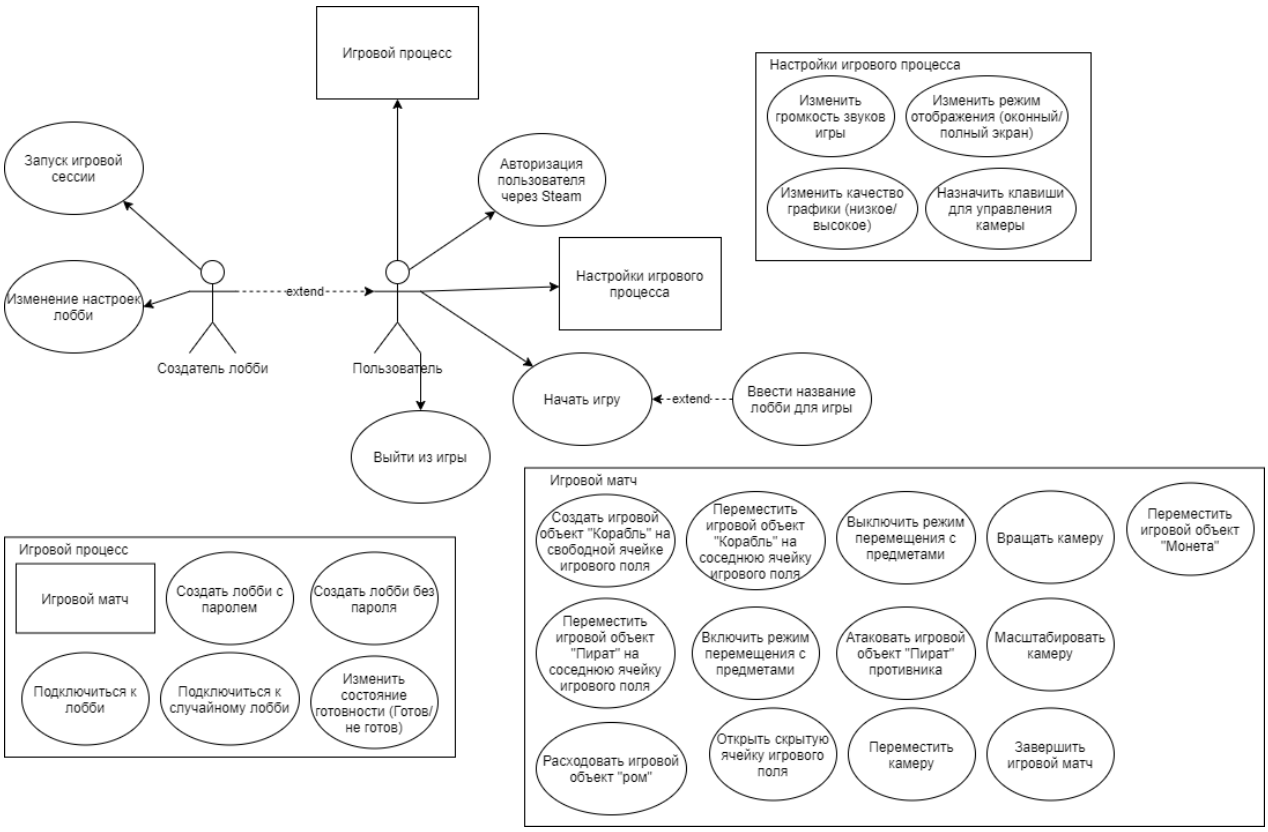

Рисунок 1 – диаграмма вариантов использования

2.3.3. Описание сущностей и связей между ними

Разрабатываемое приложение обладает большим количеством сущностей, рассмотрим основные из них. Связи между всеми сущностями представлены на диаграмме классов (рисунок 2).

1. Ячейка карты (Tile) – единица игрового поля, которая содержит информацию о собственном расположении на карте и основные операции,

описывающие взаимодействие с другими объектами, такими как «Пират» и «Монета».

2. Менеджер игрового поля (MapManager) – сущность, содержащая информацию о всех ячейках карты, расположенных на поле, а также манипулирующая данными ячейками, ячейки карты хранятся в виде двумерного массива.

3. Менеджер матрицы игрового поля (MapMatrixManager) – класс, который служит для создания матрицы, содержащей внутри себя объекты «ячейка карты» на основе матрицы целых чисел, этот класс необходим, так как передавать пакеты с массивами целых чисел по сети между пользователями куда проще и быстрее, чем передавать информацию и ячейках.

4. Пират (Pirate) – класс, в котором содержится информация о корабле, которому принадлежит данный объект, а также описывает методы, для взаимодействия с игровым объектом.

5. Монета (Coin) – сущность, описывающая собираемый объект, необходимый для завершения игры.

6. Корабль (Ship) – сущность, в которой содержится информация об игроке, который им управляет, пиратах, владеющими данным кораблём и собственном местоположении на карте.

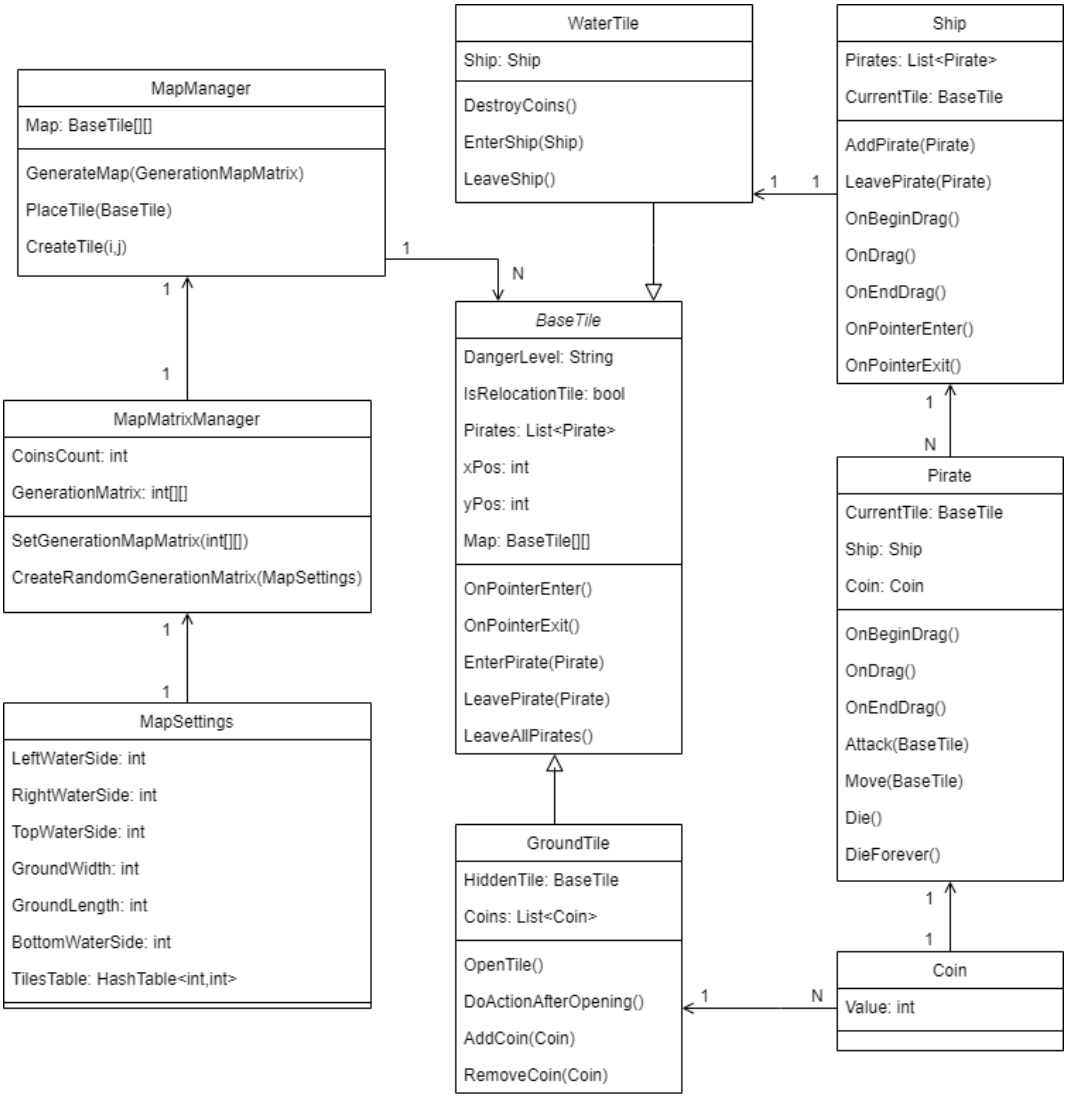

Рисунок 2 – диаграмма классов

2.3.4. Карта приложения

Навигация по приложению является немаловажной составляющей любого игрового приложения, необходимо спроектировать навигацию по приложению таким образом, чтобы пользователям было интуитивно понятно взаимодействовать с игрой. Это достигается за счёт грамотно созданного UI (пользовательский интерфейс) и опираясь, на так называемый, UX (пользовательский опыт). На рисунке 3 представлена диаграмма состояний игрового приложения, которая учитывает вышеприведённые факторы.

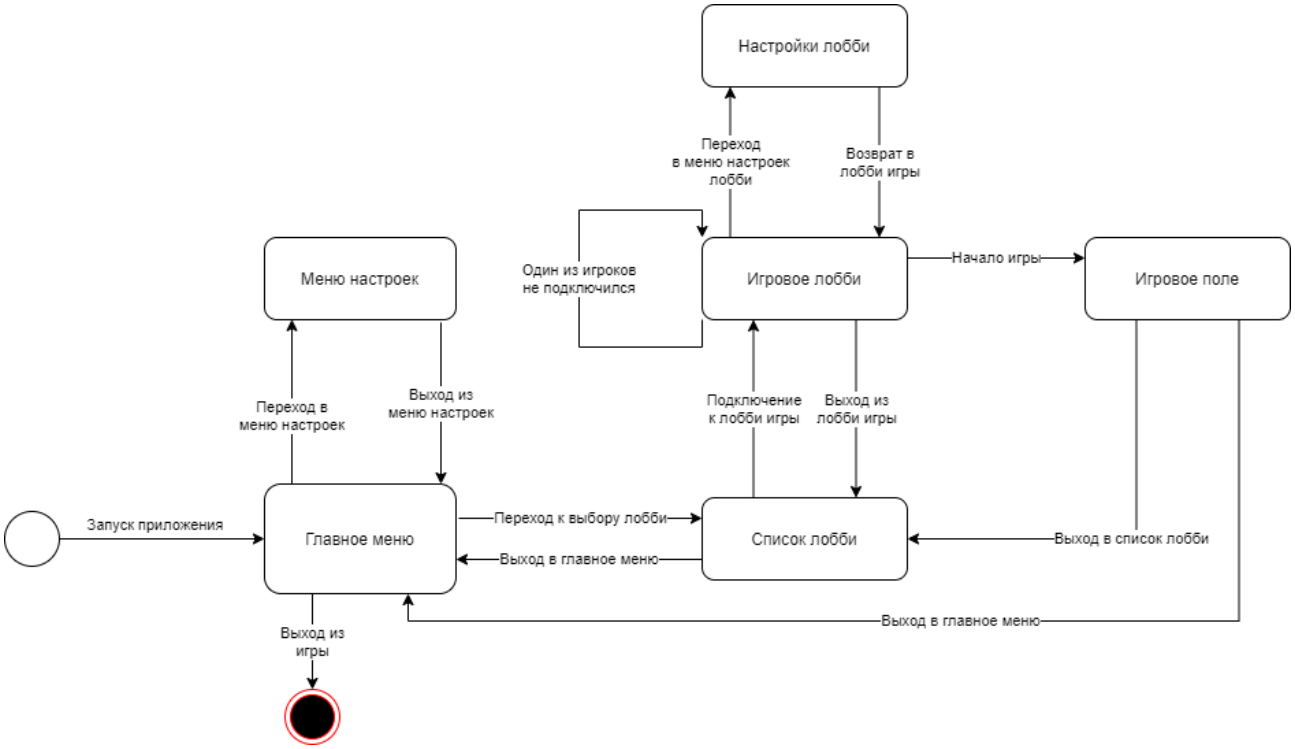

Рисунок 3 – диаграмма состояний игрового приложения

2.3.5. Проектирование последовательностей бизнес процессов

Для формирования представления о работе существующих бизнес процессов была создана диаграмма, описывающиая один из них. На рисунке 4 представлена диаграмма последовательностей подключения к игровой сессии

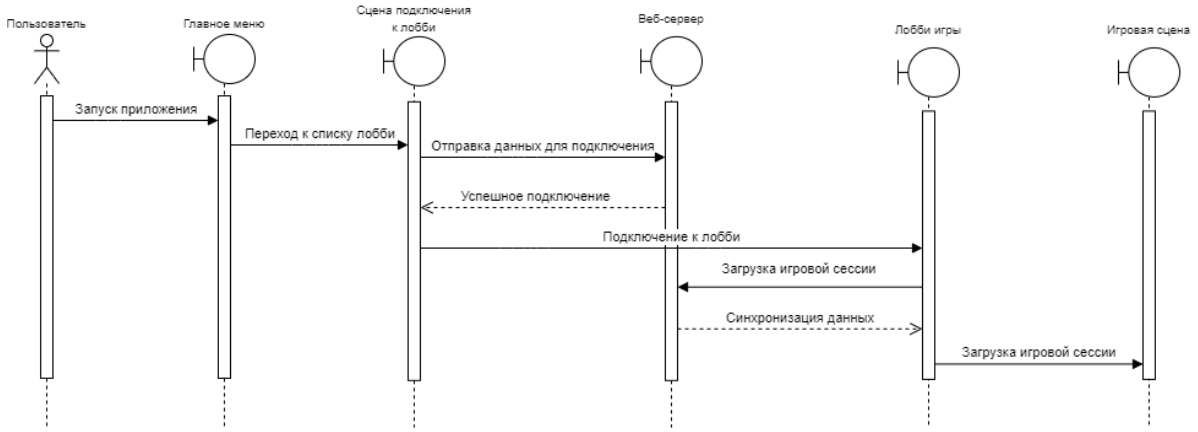

Рисунок 4 – диаграмма последовательностей подключения к игровой сессии

2.3.6. Проектирование многопользовательского взаимодействия

Многопользовательская составляющая игры будет реализована с использованием библиотеки Photon для Unity. Принцип многопользовательского взаимодействия заключается в использовании

доверенного сервера, предоставляемого разработчиками Photon, который позволяет быстро и эффективно обмениваться данными между пользователями во время игры. Таким образом будет использована клиент-серверная архитектура для взаимодействия между пользователями.

Для подключения пользователей к одной игровой сессии в Photon используются так называемые «Room». Которые можно настроить с использованием различных параметров, таких как: название, максимальное количество игроков. Также Photon предоставляет возможность добавление своих параметров для настройки комнаты. В архитектуре разрабатываемой игры будут использоваться комнаты с ограничением на максимальное количество игроков - 4, названием комнаты и паролем к ней.

В начальной версии игры коммуникация между пользователями не предусмотрена, так как каждый пользователь играет сам за себя и наличие коммуникации не является необходимой. В дальнейшем планируется добавление двух внутриигровых чатов: один для коммуникации между пользователями во время поиска подходящей комнаты и второй, уже непосредственно во время игрового процесса.

2.4. Выбор технологий и средств разработки

Для разработки игрового приложения требуется ряд технологий предназначенных для этого, в их число входит: игровой движок, IDE, графический редактор или/и средства моделирования. В первую очередь будут проанализированы существующие технологии, предназначенные для разработки игровых приложений и выбрана та, которая наилучшим образом подходит для реализации поставленной задачи. После чего будут проанализированы и выбраны прочие технологии, необходимые для успешной разработки игрового приложения.

2.4.1. Анализ существующих технических решений

На текущий момент существует множество технологий, предназначенных для разработки компьютерных игр для различных ОС, использующие различные языки программирования, предоставляющие разнообразный функционал и возможности.

В данной работе будут рассмотрены самые популярные решения для разработки компьютерных игр.

Unreal Engine – это игровой движок, поддерживаемый компанией Epic Games, предназначен для создания компьютерных игр с использованием языка программирования – C++. Обладает богатым функционалом и позволяет создавать различные кроссплатформенные игры. Является очень популярным движком для создания крупным игровых проектов, к которым можно отнести: Deus Ex, Mirror’s Edge, BioShock 2, Mass Effect 2.

Достоинства:

- Визуальное программирование (Blueprints) – возможность создания игровых приложений без написания программного кода.
- Мультизадачность и универсальность – Unreal Engine обладает богатым функционалом, который позволяет выполнить самые различные задачи.
- Цена – использование Unreal Engine является бесплатным.
- Кроссплатформенность – игровой движок позволяет создавать игры для различных платформ.

Недостатки:

- Стоимость контента в магазине – большинство работ, созданных пользователями для Unreal Engine являются дорогими.
- Разнообразие контента – отсутствует богатый выбор контента для игр в магазине.

- Требовательность к системе – Unreal Engine использует большое количество вычислительных ресурсов.

Unity3D – межплатформенная среда разработки компьютерных игр, принадлежащая американской компании Unity Technologies. Редактор Unity упрощает взаимодействие объектов на сцене и процесс сборки проекта, что обеспечивает комфортную и гибкую систему разработки игровых приложений, позволяя вести непрерывное тестирование и улучшение игрового процесса. С недавнего времени Unity поддерживает только один язык программирования C#, с помощью которого пишутся все скрипты для игровых объектов. На движке Unity написаны такие популярные игры, как SuperHot, Rust, Pokemon Go, HearthStone.

Достоинства:

- Богатый функционал – Unity предоставляет множество функциональных возможностей;
- Кроссплатформенность – игровой движок позволяет создавать игры для различных платформ;
- Прост в использовании;
- Бесплатная лицензия для коммерческих разработок.

Недостатки:

- Отсутствие возможности обращения к внешним библиотекам кода;
- Невозможность вносить изменения в физическое ядро.

Произведя оценку достоинств и недостатков представленных решений, в качестве среды для разработки игрового приложения был выбран Unity, так как он позволяет создавать маломасштабные игровые проекты, прост в использовании. Помимо всего прочего язык программирования C#, используемый в Unity, проще в использовании чем представленные альтернативы.

2.4.2. Анализ возможностей Unity3D

Unity3D является мощным средством для создания 2D и 3D игровых приложений. В этом разделе будут описаны функциональные возможности Unity, которые были использованы при разработке игрового приложения «Jackal».

Прежде чем переходить к конкретным разделам, используемым в создании проекта, опишем основные особенности, которые стали причиной выбора данного инструмента разработки компьютерных игр:

- Кроссплатформенность;
- Наличие бесплатной версии для коммерческой разработки, прибыль с которой составляет менее 100000$ в год;
- Возможность мгновенного запуска игрового приложения через встроенный эмулятор;
- Возможность настройки пользовательского интерфейса;
- Работа с внутренними ресурсами через систему Drag-and-Drop. Это система, которая позволяет манипулировать объектами, используя компьютерную мышь и перемещая объекты соответственно курсору на экране;
- Наличие магазина Asset Store, в котором можно приобрести необходимые для разработки игровые модели, скрипты, шейдеры, материалы, объекты окружения, расширения редактора, звуковые файлы, анимацию и т.д.;
- Возможность создания 2D и 3D приложений;
- Возможность использовать редактор объектов во время эмуляции.
- Возможность создания анимации, шейдеров, материалов, системы частиц в самом редакторе.

Гибкая система настройки сборки готового проекта даёт возможности выпускать игровое приложения под различные платформы без дополнительных трудностей, связанными с изучением целевой платформы, что безусловно упрощает и ускоряет процесс разработки проекта. Проект нацелен

на платформу Windows и её пользователей, поэтому сборка игрового приложения осуществлялась для данной платформы.

При сборке проекта есть возможность настройки различных параметров, таких как:

- Целевая платформа;
- Архитектура ОС – позволяет выбрать архитектуру ОС (x86, x86_64 и универсальная);
- Тип сборки (Development Build) – включает в сборку консоль, в которую выводятся исключения и ошибки, возникающие в процессе эксплуатации;
- Создание решения Visual Studio – создание файлов для создания решения Visual Studio, которое позволяет запустить последнюю сборку прямо из редактора Visual Studio;
- Сборка в режиме дебага скриптов – позволяет создавать специальные символы в файлах скриптов для дебаггинга готовой сборки проекта;
- Порядок запуска игровых сцен – позволяет настроить порядок, в котором Unity будет запускать игровые сцена.

Ниже приведён список основных терминов Unity, которые используются при создании игрового приложения.

Сцена (Scene). По своей сути сцена представляет собой игровой уровень, который содержит внутри себя информацию об окружении и игровых объектах.

Игровой объект (GameObject). Это фундаментальный объект, который используется для построения сцены, он может представлять собой персонажа, объект окружения, камеру, освещение и многое другое. Функциональности игрового объекта определяется его компонентами.

Компонент (Component). Функциональная часть игрового объекта, определяющая его поведение, игровой объект может содержать

неограниченное количество компонентов. Пользователь может создавать собственные компоненты для игровых объектов.

Скрипт (Script). Компонент, который представляет из себя программный код, описывающий поведение игровых компонентов. Каждый скрипт наследуется от общего класса MohoBehaviour, который определяет такие методы как Start() и Update(). Метод Start() будет вызван Unity до начала игрового процесса. Метод же Update() вызывает каждый кадр и обычно используется для отображения движения, обработки различных событий.

Ресурс (Asset). Это любые данные, которые могут использоваться в проекте Unity. Ресурс может быть получен из файла, созданного вне Unity, например, 3D-модель, аудиофайл или изображение. Помимо этого, ресурсы можно получить из магазина для Unity – AssetStore.

Шаблон экземпляра (Prefab). Это ресурс, который представляет из себя игровой объект, включающий в себя различные компоненты и свойства. Шаблон экземпляра используется для инициализации готовых игровых объектов на сцене.

Меш (Mesh). Основной графический примитив Unity, 3D игровые пространства Unity состоят из мешей. Unity поддерживает треугольные и четырехугольные полигональные сетки. Прочим 3D объекты необходимо преобразовывать в полигоны, для дальнейшей работы с ними.

Шейдер (Shader). Представляет собой программу, которая работает на графическом процессоре, использующие при рендере и описывающие внешний вид объекта.

2.5. Создание игрового приложения

После того, как была спроектирована структура игрового приложения и выбраны необходимые технологии можно приступить к непосредственной разработки игрового приложения.

2.5.1. Описание игрового процесса

Jackal – многопользовательское игровое приложение в жанре пошаговая стратегия, ориентированное на формирование тактических навыков. Основная задача игрового процесса заключается в поиске и собирании монет, за счёт исследования игрового поля и ведения боевых действия с противниками.

Составляющей игрового процесса являются игровые объекты. В разрабатываемом приложении используется следующий перечень игровых объектов:

Пират (рисунок 5). Основной игровой объект, который можно перемещать по ячейкам игрового поля, используя Drag & Drop систему. Служит для перемещения объекта «Монета» по игровому полю, а также для атаки игровых объектов собственного типа. При перемещении на «Закрытые ячейки» игрового поля уничтожает их, вызывая генерацию случайных открытых ячеек.

Рисунок 5 – Пират

Корабль (рисунок 6). Игровой объект, который обладает следующим перечнем специальных возможностей: способен перемещаться только по ячейкам игрового поля типа «Вода»; служит для точки возрождения игровых объектов «Пират»; является пунктом назначения для доставки игрового объекта «Монета»; перемещается, используя Drag & Drop систему.

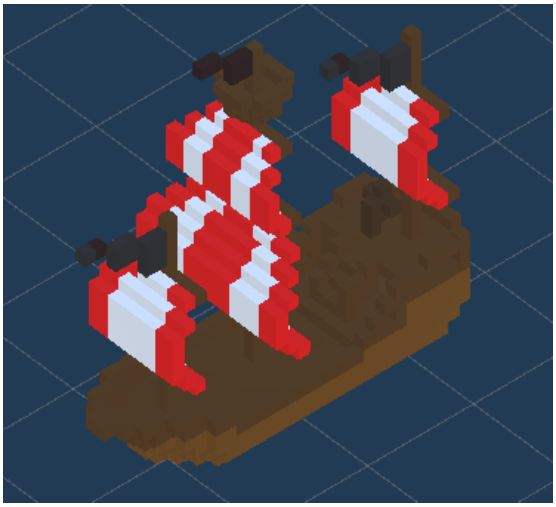

Рисунок 6 – игровой объект «Корабль»

Монета (рисунок 7). Игровой объект монета является единственным ресурсом игрового процесса. При попадании на игровой объект «Корабль» уничтожается, увеличивая счётчик монет у игрока, которому принадлежит «Корабль» на 1. При перемещении монеты на ячейку игрового поля «Вода» уничтожается безвозвратно. Данной объект является нейтральным – это означает, что у него нет владельца, который может им управлять.

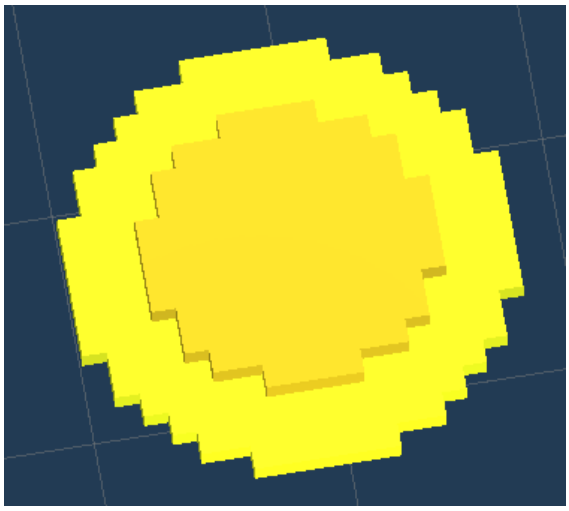

Рисунок 7 – игровой объект «Монета»

Ячейки игрового поля. Основной объект взаимодействия на игровом поле, которые обладают своими уникальными особенностями, которые зависят от типа ячейки. В игровом приложении присутствует 13 видов ячеек игрового поля, некоторые из них обладают схожими свойствами, поэтому будут объединены в одну категорию:

- Закрытая ячейка (рисунок 8) – тип ячейки, которая служит для сокрытия ячейки поля, на которую перемещается игровой объект «Пират». Уничтожается при перемещении и заменяется одной из «открытых ячеек».

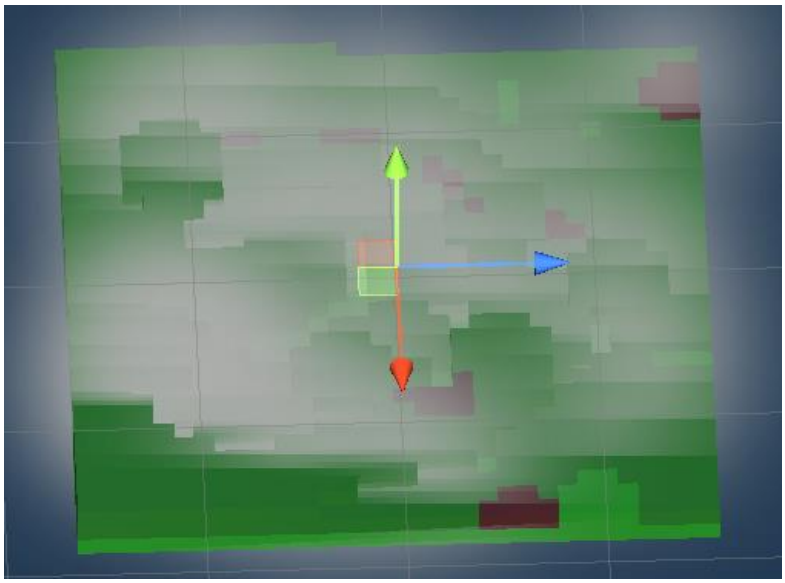

Рисунок 8 – Игровой объект «Закрытая ячейка»

- Пустая ячейка (рисунок 9) – тип открытой ячейки, которая не обладает уникальными свойствами. Перемещении с этой ячейки возможно во всех направлениях, за исключением ячейки «Вода».

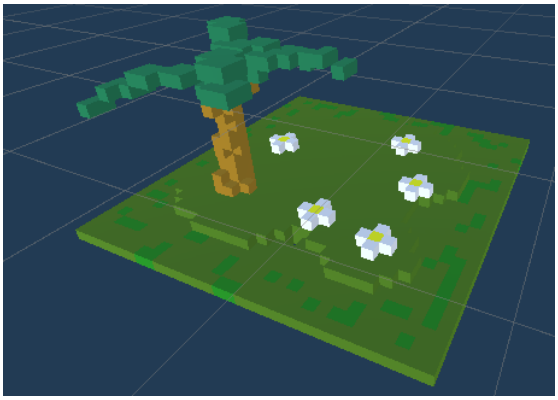

Рисунок 9 – Игровой объект «Пустая ячейка»

- Вода (рисунок 10) – тип ячейки игрового поля, по которой перемещается игровой объект «Корабль». Перемещение объекта «Пират» на данную ячейку возможен только со специальных ячеек-стрелок, позволяющие перемещаться на любой тип ячеек. Перемещение с этой ячейки осуществляется

подобно перемещению с пустой ячейки, но только по ячейкам «Вода». Атаки, совершаемые на данной ячейке уничтожают игровые объекты безвозвратно.

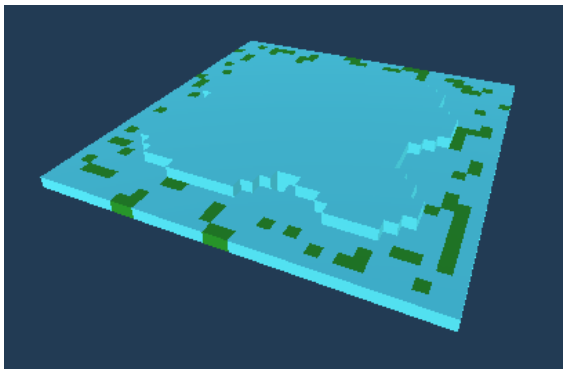

Рисунок 10 – Игровой объект «Вода»

- Стрелки (рисунок 11) – совокупность игровых ячеек, обладающими уникальным принципом работы: перемещение с этих ячеек осуществляется в направлении, соответствующем направлению стрелок, изображенных на данных ячейках; перемещение на данные ячейки не завершает ход игрока; перемещении с данного типа ячеек возможно на любые ячейки.

Рисунок 11 – Игровые объекты «Стрелки»

- Конь (рисунок 12) – подвид ячейки «Стрелки», которая обладает теми же особенностями, но перемещении осуществляется подобно одноимённой шахматной фигуре.

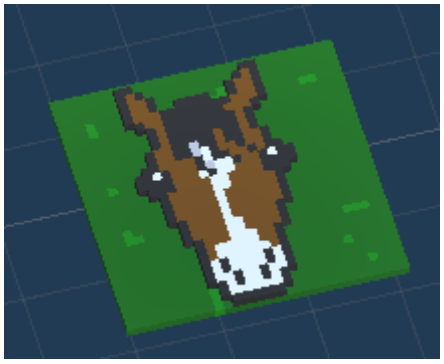

Рисунок 12 – игровой объект «Конь»

- Лёд (рисунок 13) – вид игровой ячейки, которая обладает теми же свойствами, что и игровая ячейка «Стрелки», однако перемещении осуществляется в направлении, повторяющем направление передвижения на эту ячейку.

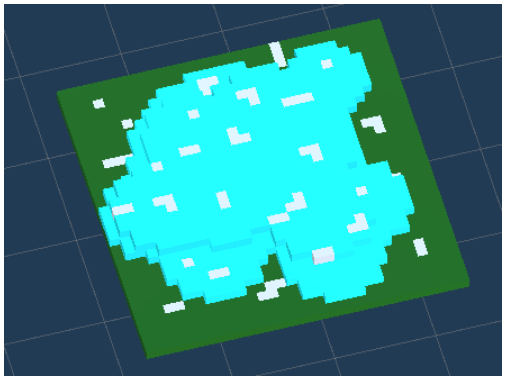

Рисунок 13 – игровой объект «Лёд»

- Пещера (рисунок 14) – вид игровой ячейки, которая взаимодействует с другими игровыми ячейками того же типа. Первое перемещение на игровую ячейку «Пещера» блокирует игровой объект «Пират» до тех пор, пока не будет обнаружена другая ячейка «Пещера», при этом ход игрока не завершается, игровые объекты «Пират» перемещаются на противоположные ячейки «Пещера», после чего могут перемещаться также, как и с пустой ячейки. Повторное перемещение на ячейку «Пещера» позволяет переместиться в любую другую ячейку того же типа на игровом поле.

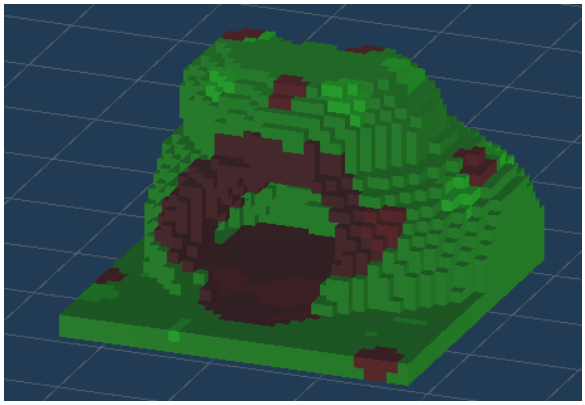

Рисунок 14 – игровой объект «Пещера»

- Сундук (рисунок 15) – вид игровой ячейки, при открытии которой на ней генерируется монета. Обладает теми же свойствами, что и пустая ячейка.

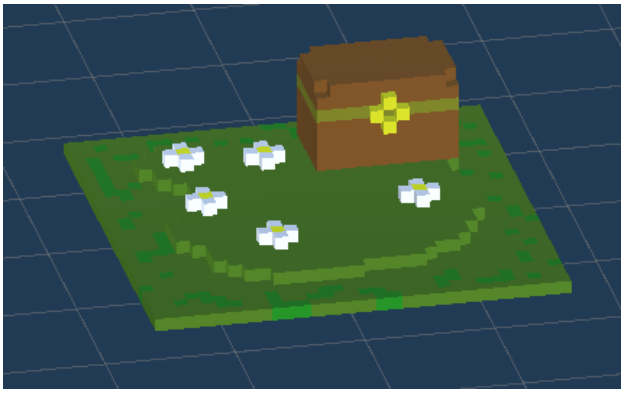

Рисунок 15 – игровой объект «Сундук»

Помимо игровых объектов реализованы концепции различных игровых механик игрового процесса.

Перемещение. Вид пользовательского взаимодействия, принцип которого заключается в перемещении игровых объектов «Пират» и «Корабль» по игровому полю. При этом пират перемещается по всем доступным ячейкам игрового поля, подчиняясь правилам перемещения ячеек различного типа. Корабль может перемещаться только по игровым ячейкам – «Вода». При перемещении игрового объекта «Пират» или игрового объекта «Корабль» ход передаётся следующему игроку, за исключением перемещения на игровые ячейки, описанные в приложении 1. Для перемещения используется Drag & Drop система. В разрабатываемом приложении данная система была несколько изменена, но её принцип сохранился – при нажатии на игровой объект создаётся стрелка, которую можно перемещать по игровому полю, подобно самому игровому объекту. При перемещении объекта «Пират» с корабля применяется специальное правило, при котором возможно только ортогональное перемещении. Примеры перемещений игровых объектов «Пират» и «Корабль» представлены на рисунках 16-17 соответственно.

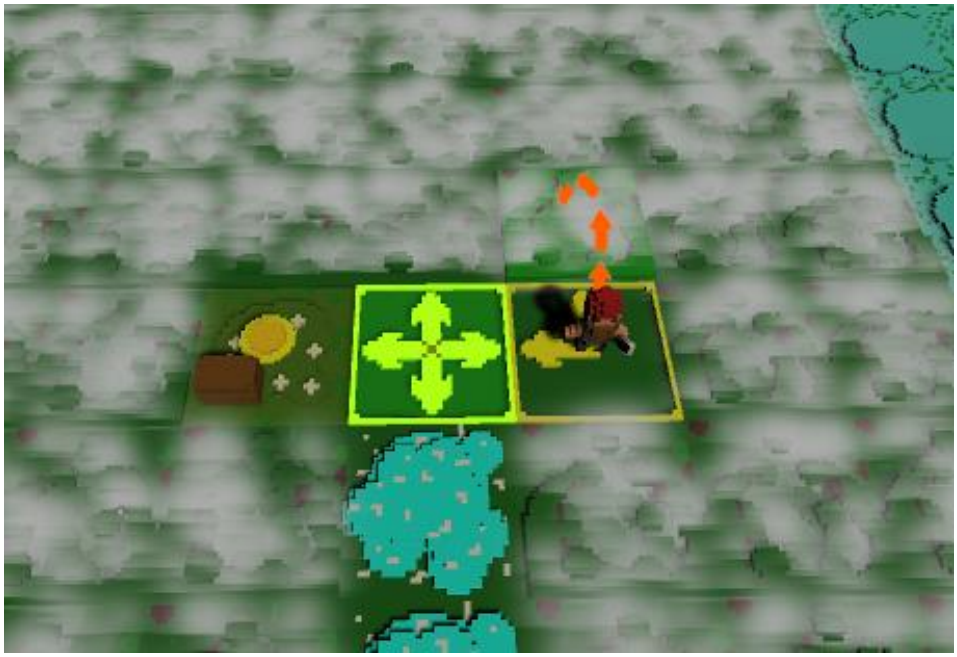

Рисунок 16 – Перемещение объекта «Пират»

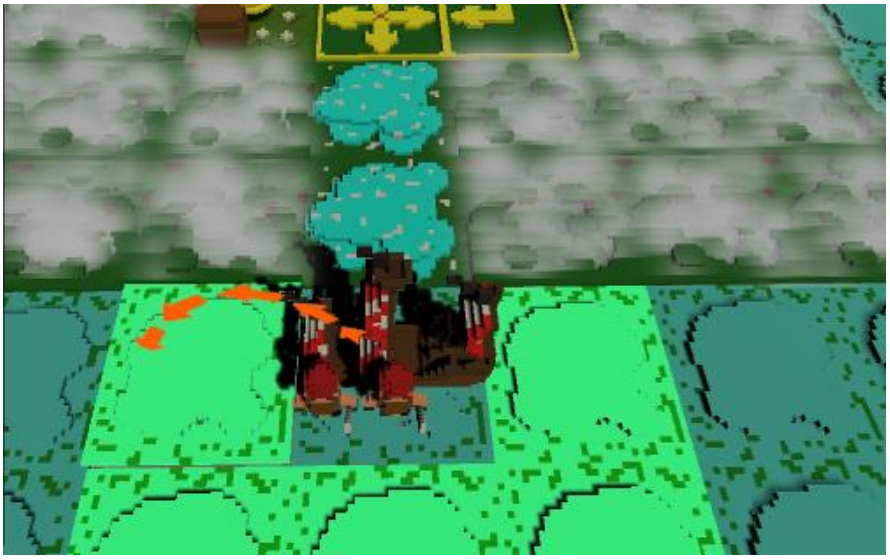

Рисунок 17 – Перемещение объекта «Корабль»

Сбор игровых монет. Пользовательское взаимодействие, принцип работы которого заключается в переносе монет с карты игрового поля на корабль, принадлежащий пользователю, что им управляет. Перемещение осуществляется с помощью игрового объекта – «Пират». Для перемещения монет по игровому полю, необходимо переместить пирата с ячейки игрового поля, на которой находится монета, на любую другую ячейку игрового поля, при этом в пользовательском интерфейсе необходимо включить режим «Перемещение с монетой». При перемещении монеты на корабль – игровой объект «Монета» уничтожается, а в пользовательском интерфейсе счётчик

монет увеличивается на единицу. При перемещении монеты на игровую ячейку – «Вода», на которой отсутствует подконтрольный корабль, игровой объект «Монета» безвозвратно уничтожается, не увеличивая счётчик монет в пользовательском интерфейсе. Демонстрация данной механики представлена на рисунках 18-19.

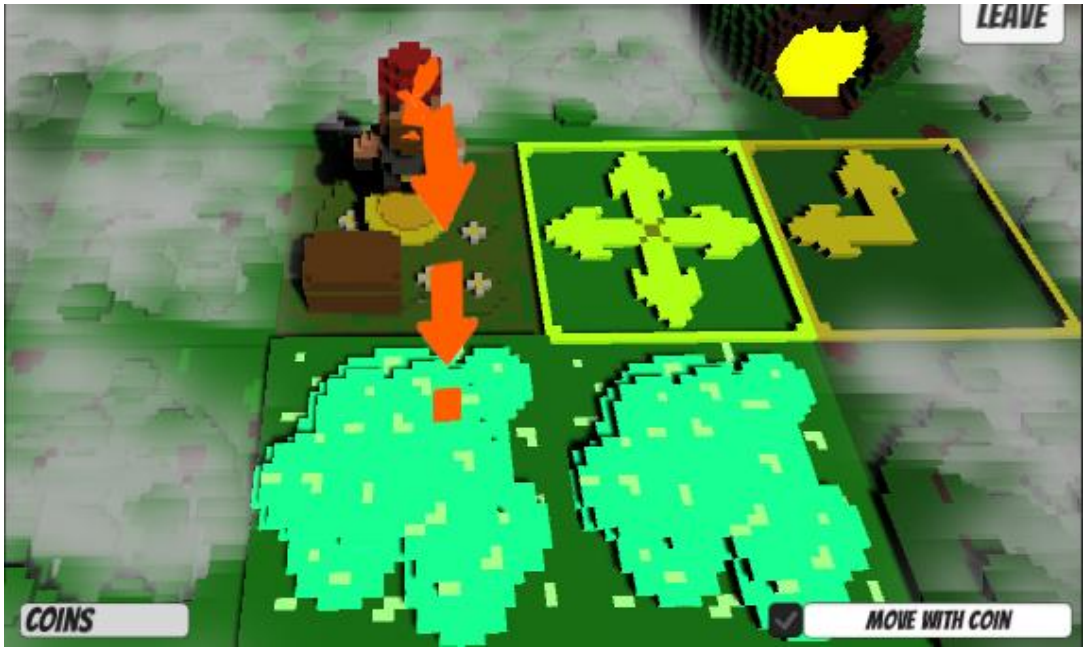

Рисунок 18 – перемещение объекта «Монета»

Рисунок 19 – завершение перемещения объекта «Монета»

Атака. Вид перемещения, при котором объект «Пират» перемещается на ячейку игрового поля с пиратом/ пиратами, принадлежащем другому игроку.

При этом атака выполняется по такому-же принципу, как и в шахматах – объект, на который сходили считается срубленным, но в отличие от шахмат не убирается с доски, а перемещается обратно на корабль. При атаке уничтожаются все объекты «Пират», которые находились в данный момент на ячейке игрового поля. Существуют правила, при которых объект может быть уничтожен окончательно: при перемещении объекта «Пират» на корабль противника, или наоборот, при нападении на ячейке «Вода».

Завершение игрового процесса. Игровой процесс завершается, когда все монеты на игровом поле собраны. Побеждает игрок, собравший наибольшее количество монет.

2.5.2. Игровые сцены

Для успешной работы игрового приложения было создано 4 игровых сцены, каждая из которых служит для определённого рода задач:

Главная сцена (Рисунок 20). Первичная сцена, загружаемая при запуски игрового приложения. В данной версии игрового продукта на главной сцене присутствует 2 активных кнопки и анимированная сцена в левом секторе экрана. Служит для перехода на сцену поиска лобби или выхода из игрового приложения.

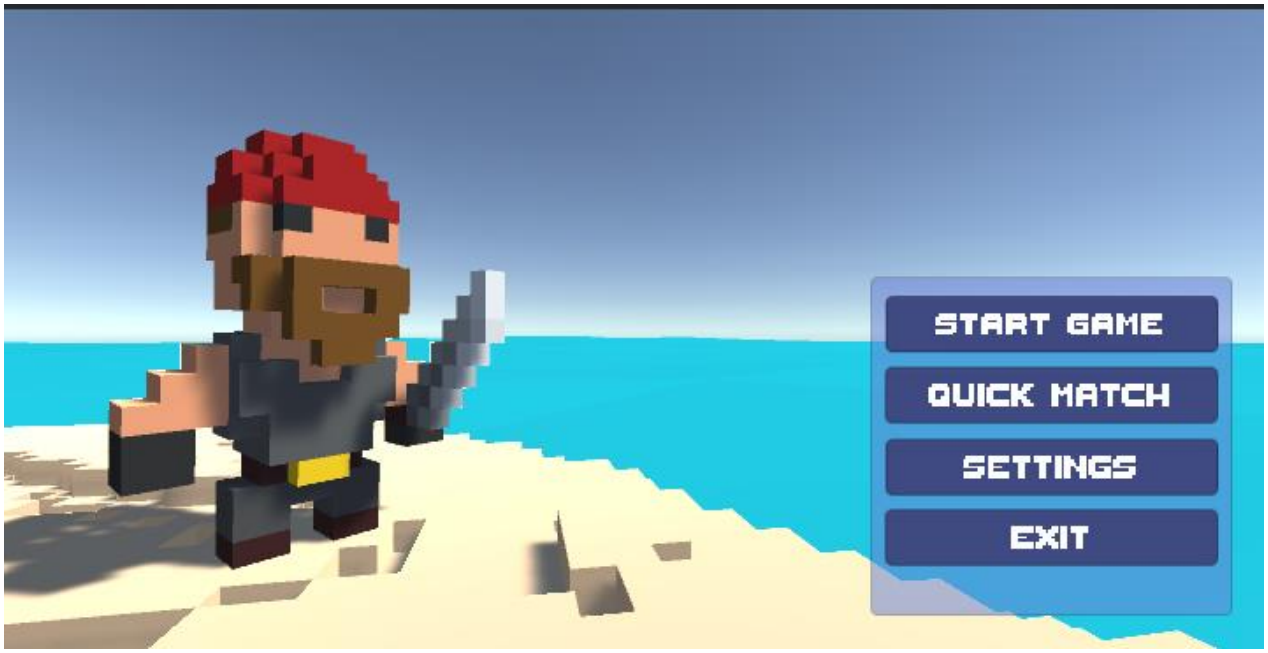

Рисунок 20 – главная сцена

Сцена поиска лобби (Рисунок 21). Сцена предназначенная для создания лобби с уникальным именем или для подключения к уже существующему

игровому лобби. Создание лобби и подключение к уже существующему происходит автоматически, таким образом, если введённая строка названия лобби для подключения уже существует, то происходит подключение к нему, в противном случае – создается новое лобби. Содержит такие элементы пользовательского интерфейса, как кнопка «Подключение к лобби» и окно вывода сообщений о состоянии подключения к серверу и игровой сессии.

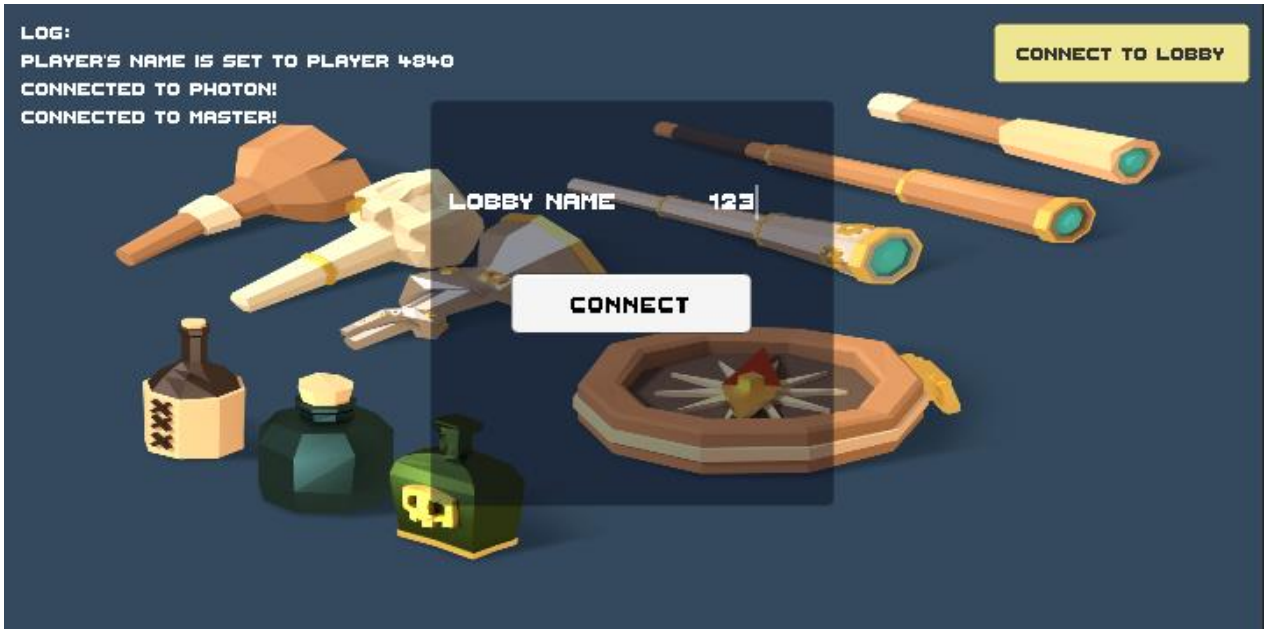

Рисунок 21 – сцена поиска лобби

Сцена игрового лобби (Рисунки 22-23). Сцена, которая служит для формирования лобби и запуска основного игрового процесса. Для всех игроков на сцене существуют такие элементы пользовательского, как кнопка «Готов»; панель подключившихся игроков, на которой отображается готовность к игре конкретного пользователя и его игровой никнейм. Создатель лобби помимо этого обладает дополнительными элементами управления, а именно кнопкой, для запуска игровой сессии, и отдельной панели с настройками игрового поля, где можно настроить такие параметры, как длина, ширина, количество ячеек игрового поля различных типов, всё это настраивается при помощи элемента управления – слайдер.

Рисунок 22 – отображение сцены игрового лобби для создателя лобби

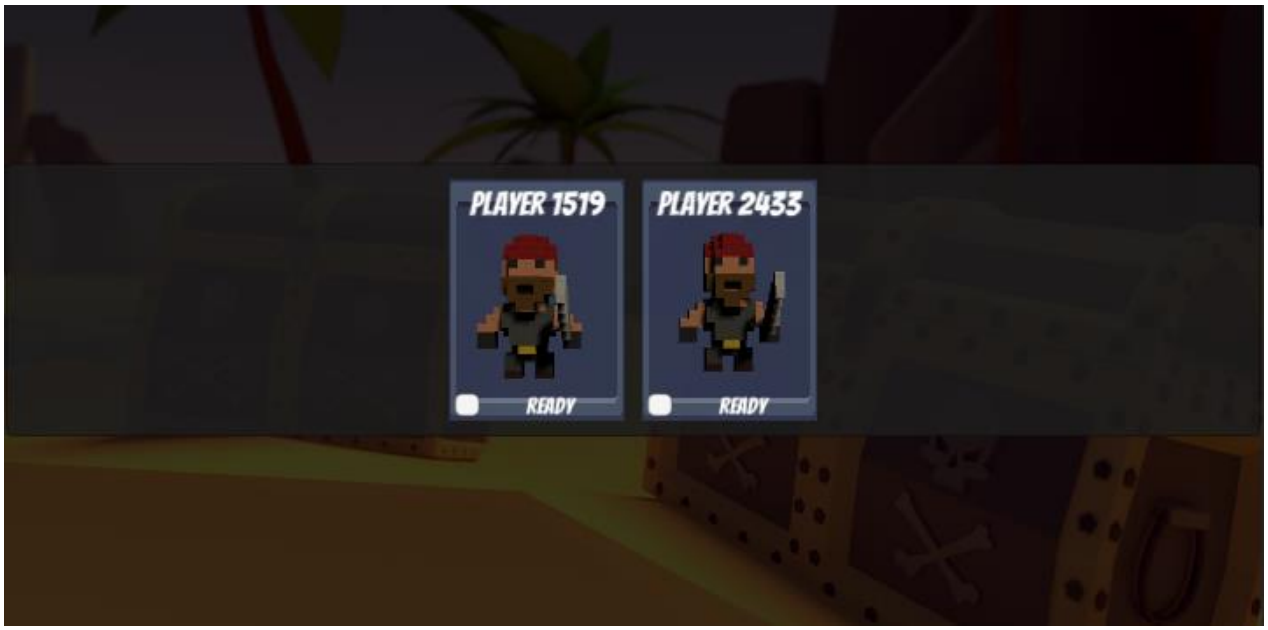

Рисунок 23 – отображение сцены игрового лобби для игрока

Сцена игрового поля (рисунок 24). Сцена, предназначенная для непосредственной игры. Эта сцена содержит игровое поле, которое состоит из ячеек, описанными ранее. Пользовательским интерфейсом, на котором расположена информация о пользователях (количество монет, количество пиратов). При загрузке игрового поля отображается информация текущей очереди для совершении ход (рисунок 25). Размещение игрового объекта «Корабль» на игровое поле является ходом. В правом верхнем углу сцены располагается кнопка для выхода в главное меню, при этом игрок, покинувший

комнаты, считается проигравшим, если в комнате остался только один пользователем.

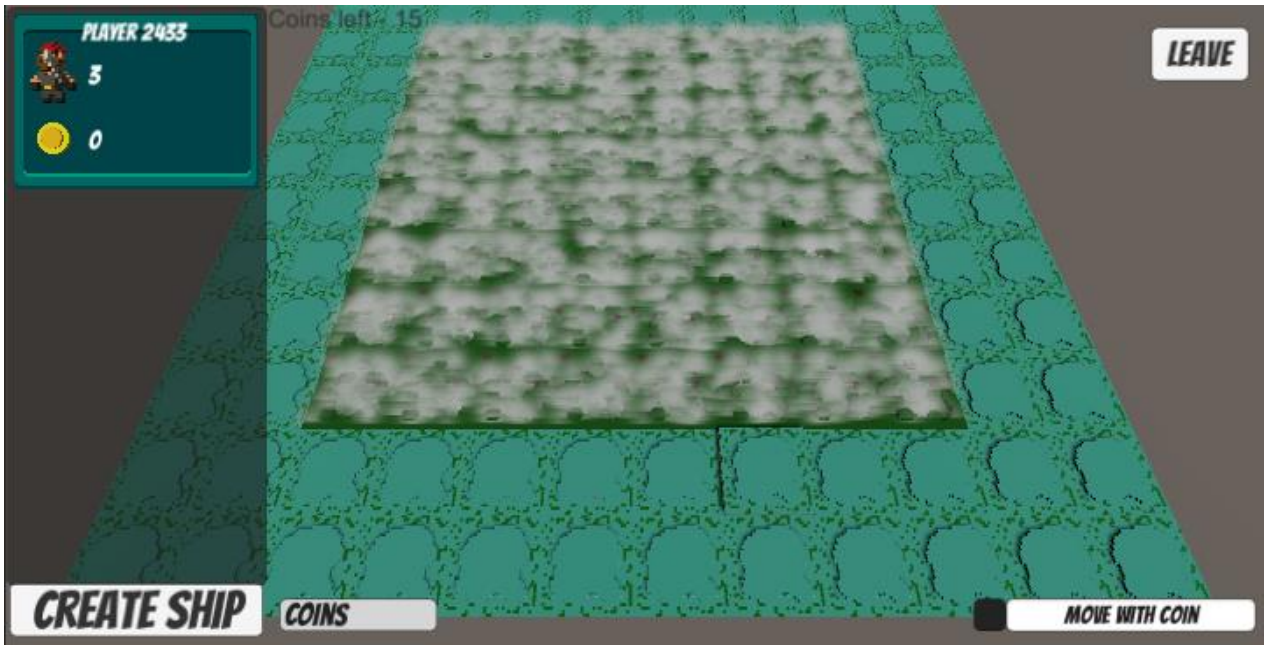

Рисунок 24 – игровая игрового поля

Рисунок 25 – отображение очерёдности хода

2.5.3. Моделирование игровых объектов

Моделирование – метод воспроизведение объектов реального мира или их составляющих в виде упрощенных моделей. Для создания игровых объектов для многопользовательского игрового приложения в жанре пошаговая

стратегия «Jackal» использовалась воксельная графика, которая является аналогом пиксельной графики для трехмерного пространства.

Игровые объекты были созданы в программе MagicaVoxel, после чего импортированы в Unity. На рисунке 26 представлен объект «Корабль» в редакторе MagicaVoxel.

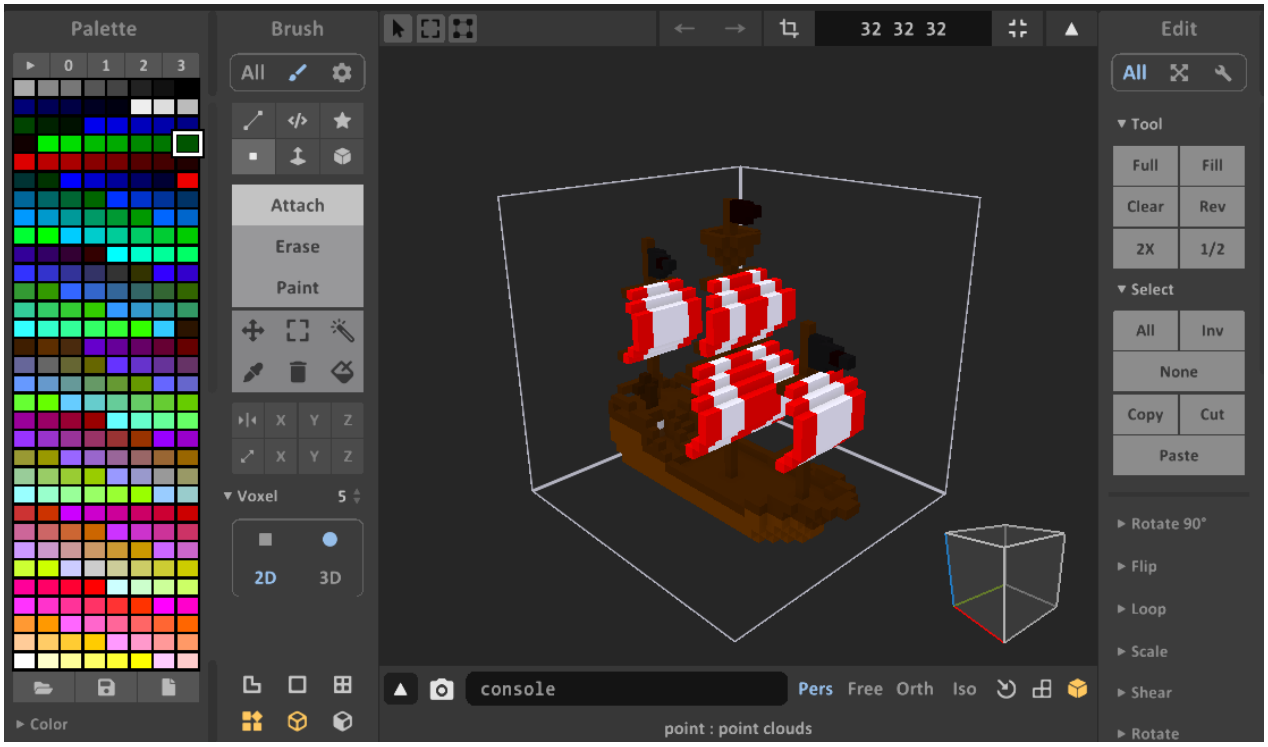

Рисунок 26 – модель корабля в MagicaVoxel

Каждый игровой объект в Unity состоит из 3-х составляющих: трехмерная модель, загруженного 3D объекта, которая представляет из себя совокупность полигонов (рисунок 27); материала – набор свойств объекта, с помощью которых можно изменять внешний вид его поверхности. Материал может содержать в себе: цвет, текстуру, блик, рельеф объекта (рисунок 28); текстуры, которая представляет собой полосу, высотой в один пиксель с набором цветов (рисунок 29).

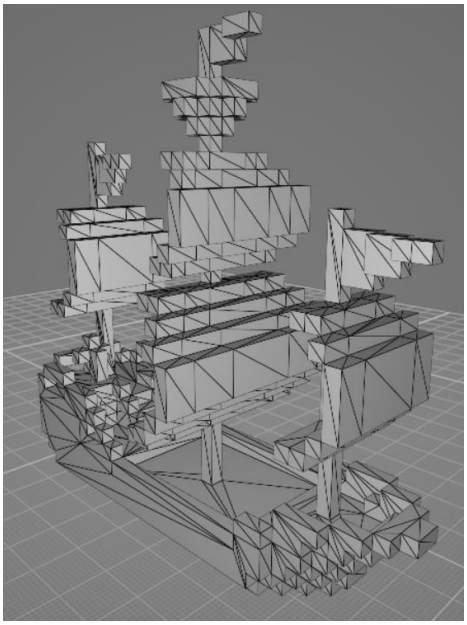

Рисунок 27 – полигоны 3D модели «Корабль»

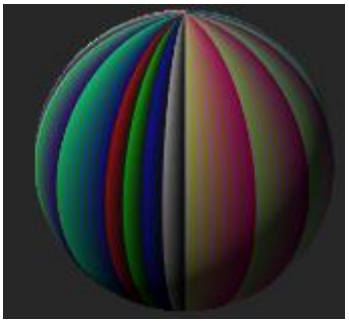

Рисунок 28 – материал, игрового объекта «Корабль»

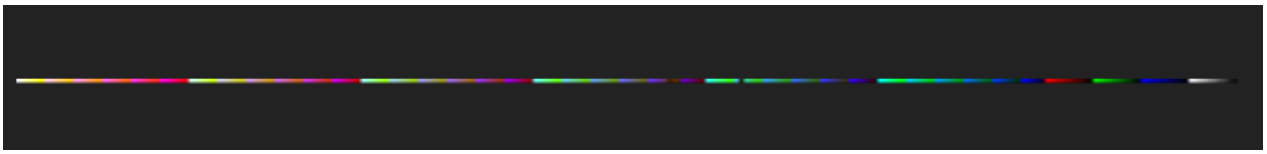

Рисунок 29 – текстура, для игрового объекта «Корабль»

2.5.4. Описание реализации программной части

Перед разработчиком стояла задача реализовать основные игровые механики, описанные ранее, реализовать многопользовательское взаимодействие, генерацию игрового поля, реализовать поведение RTS камеры для перемещения камеры по игровому полю.

2.5.4.1. Генерация игрового поля

Для реализации генерации игрового поля, состоящего из случайно-расположенных игровых ячеек было создано несколько вспомогательных классов.

Класс MapSettings предназначен для формирования параметров генерируемого поля, таких, как размер, количества различных типов ячеек игрового поля.

Класс MapMatrixManager генерирует матрицу, состоящую из целых чисел, которые соответствуют типам ячеек игрового поля. Это необходимо для того, чтобы было проще синхронизировать потоки данных между пользователями. На основе данной матрицы впоследствии генерируется игровое поле. Для того, чтобы генерация игрового поля была одинаковой у всех пользователей, матрица генерируется у создателя лобби, после чего передаётся остальным пользователям.

Класс MapManager служит для генерация поля на игровой сцене на основе матрицы случайных положительных чисел, полученной с помощью класса MapMatrixManager. Для определения соответствия числа к конкретной категорией ячейки игрового поля создаётся хеш-таблица, где хранится информация о числе и типе игровой ячейки. После чего создаётся поле, располагая ячейки на сцене в зависимости от их расположения в матрице.

После генерации игрового поля оно становится полностью интерактивным и готовым к использованию в игровом процессе.

Пример программного кода представлен в Приложении А в листинге классов генерации игрового поля.

2.5.4.2. Ячейки игрового поля

Описание функционала ячеек игрового поля занимает большую часть программного кода. Каждая из ячеек находится в чёткой наследственной иерархии.

В вершине этой иерархии находится абстрактный класс Tile, который описывает базовый функционал, которой присущ всем остальным наследникам. В классе Tile содержится информация о положении ячейки игрового поля на сцене; описывается поведение при совершении событий входа и выхода курсора мыши в область видимости игровой ячейки; содержит стандартную логику перемещения игрового объекта «Пират» на соседние ячейки игрового поля.

Ниже по иерархии находятся классы BasicTile и ClosedTile.

ClosedTile реализует логику открытия новой ячейки, а также реализует метод входа объекта «Пират» в ячейку.

BasicTile содержит информацию об объектах «Пират», которые находятся на нём, также описывает основные метода, которые служат для размещения объектов «Пират» на ячейке; входа и выхода из ячейки; совершение атаки. Прямыми наследниками класса BasicTile являются классы WaterTile и OpenedTile.

Класс OpenedTile служит для реализации открытия игровой ячейки и является прямым наследником для остальных игровых ячеек.

Класс WaterTile реализует функционал перемещения корабля и переопределяет методы входа и выхода объектов «Пират» с ячейки.

Класс GroundTile расширяет класс OpenedTile и добавляет взаимодействие с игровыми объектами «Монета», хотя сам и не порождает их.

Класс CoinTile реализует функционал генерации объекта «Монета», и является прямым наследником класса GroundTile.

Класс ArrowTile служит для реализации механики незамедлительного перемещения игрового объекта «Пират» по игровому полю, а также переопределяет метод отвечающий за перемещение объекта «Пират» таким образом, что объект «Пират» обязан двигаться в направлении стрелки.

Прочие ячейки игрового поля переопределяют существующие методы, такие как перемещение объекта «Пират», расположение объектов «Пират» на ячейке и пр.

Пример кода, описывающий ячейки игрового поля представлен в Приложении А.

2.5.4.3. Игровой объект «Пират»

Игровой объекта «Пират» является главным объектом управления пользователя, который осуществляет основные игровые взаимодействия. Является синхронизируемым объектом, это означает, что информация об объекте передаётся между пользователями с помощью пакетов данных. Синхронизируются такие параметры, как положение на сцене, вращение и размеры объекта.

Класс Pirate реализует множество различных методов, которые можно распределить на три группы:

DragEvents. Обработчики событий Drag & Drop, состоящие из трех методов.

OnBeginDrag – обработчик события, срабатывающий при начале перемещения игрового объекта, внутри данного происходит проверка на принадлежность игрового объекта текущему игроку и очередность хода, а также отображаются доступные для перемещения ячейки игрового поля.

OnDrag – обработчик события, срабатывающий при перемещении игрового объекта. Обработчик событий получает данные местоположения курсора мыши на экране с помощью класса Raycast, который пускает лучи на плоскость игрового поля. На основании текущего местоположения игрового объекта и позиции игровой мыши на сцене отображается стрелка, которая отображает траекторию движения.

OnEndDrag – обработчик события, срабатывающий при завершении перемещения игрового объекта. В теле метода происходит проверка на

возможность перемещения на выбранную ячейку игрового поля. В случае если перемещение возможно – формируется пакет данных, которые передаётся остальным пользователям и вызывается событие перемещения объекта «Пират» на игровую ячейку.

MovableOptions. Группа методов, внутри которых происходит проверка возможности перемещения на ячейку игрового поля, и проверка на перемещение с монетой. Пример фрагментов кода приведён.

MovableActions. Группа методов, отвечающие за перемещение объекта пират; уничтожение объекта; сброс игрового объекта монета.

2.5.4.4. Игровой объект «Корабль»

Игровой объект корабль обладает схожими характеристиками, что и объект «Пират», однако помимо методов передвижения и реализации обработчиков событий для перемещения игрового объекта по сцене, содержит внутри себя информацию об игровых объектах «Пират», которые находятся на данном игровом объекте и которые принадлежат данному игровому объекту.

2.5.4.5. Игровой объект «Монета»

Игровой объект, внутри которого находится метод для перемещения игрового объекта по игровому полю.

Программный код, описывающий игровые объекты «Монета», «Пират» и «Корабль» представлены в Приложении А.

2.5.4.6. RTS камера

При создании игрового приложения в жанре стратегия на игровом движке Unity не был найден скрипт, описывающий поведение RTS камеры, который бы удовлетворял всем потребностям игрового приложения. Именно поэтому было принято решения создания программного кода для перемещения RTS камеры в пространстве с гибкой системой настроек параметров.

Главным отличием от аналогов является наличие автоматически настраиваемых границы камеры в зависимости от положения в пространстве и возможность сохранения положений камеры, которых не было обнаружено ни в одном из существующих аналогов для Unity на площадке AssetStore.

Для реализации передвижения камеры при помощи мыши использовалась система из 4-х прямоугольных областей (рисунок 30), при попадании курсора в одну из областей камера начинает двигаться в соответствующем направлении параллельно горизонтальной плоскости.

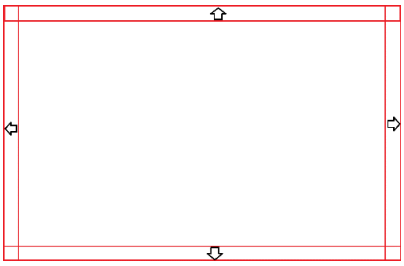

Рисунок 30 – Области видимости курсора мыши

Для реализации перемещения с помощью клавиатуры использовался стандартный функционал Unity, такой, как Input.GetAxis(), который получает в качестве аргумента строку, которая указывает на то, вдоль какой оси будет осуществляться перемещение.

Масштабирование реализовано через базовые методы пакета Unity - Mathf. В частности, использовались методы Clamp и Lerp. Метод Clamp необходим для ограничения максимальной и минимальной величины масштабирования. Метод Lerp необходим для линейной интерполяции высоты камеры.

Для вращения камеры используется базовый функционал Unity, такой, как Rotate и RotateAround, для вращения вокруг собственной оси и вокруг точки в пространстве соответственно. Точка в пространстве, вокруг которой происходит вращение камеры определяется посредством использования стандартного класса Unity – Ray и преобразованием координат центра экрана камеры в луч (ScreenPointToRay), направленный к плоскости. Величина луча вычисляет по теореме Пифагора.

Для изменения величины границы камеры, в зависимости от масштаба, использовалось подобие треугольников. Для этого берётся пересечение области профильной плоскости камеры и линии горизонта и из теоремы подобия высчитывается величина, на которую может переместиться камера в горизонтальной плоскости. При этом границы положения камеры смещаются в пространстве в зависимости от угла наклона к плоскости, чтобы этого не происходило используется формула для определение ближней границы камеры (1).

$$b = \frac{(hmax - h)}{tg(\beta)} \quad , \qquad (1)$$

где b – величина ближней границы камеры

$hmax$ – максимальная высота камеры

h – текущая высота камеры

β – угол между нижней границей обзора камеры и плоскостью

Для определения границ камеры при повороте используется уравнение вращения точки относительно другой в пространстве (2) и изменение границ на величину вращения. При этом вычисление границ камеры и её текущего положения выполняется уже после её вращения и перемещения с использованием метода Clamp, о котором говорилось ранее.

$$x' = x_0 + (x - x_0)\cos\alpha - (y - y_0)\sin\alpha \qquad (2)$$
$$y' = y_0 + (x - x_0)\sin\alpha + (y - y_0)\cos\alpha,$$

где x – начальная позиция точки по оси x

y – начальная позиция точки по оси y

x_0 – позиция точки вращения по оси x

y_0 – позиция точки вращения по оси y

x' - новая позиция точки по оси x

y' - новая позиция точки по оси y

α – угол поворота камеры

Для сохранения информации о положении камеры используются горячие клавиши на клавиатуре, которые пользователь может сам установить,

которые сохраняют информацию о позиции и вращении объекта, для этого используется Vector3 и Quaternion, которые являются базовыми классами в Unity.

Класс, отвечающий за поведение RTS камеры представлен в Приложении А.

2.5.4.7. Описание многопользовательской составляющей

Для реализации многопользовательского режима используется фреймворк Photon Network. В Photon Network используются такие понятия, как комната, игрок, уровень. Под комната понимается игровое лобби, которое служит для создания игровой сессии для пользователей. Игрок – это пользователь, который обладает такими характеристиками, как имя, id, регион. Под уровнем понимается игровая сцена, на которой происходит многопользовательское взаимодействие.

Для начала работы с Photon Network создаётся класс подключения к игровому лобби – LobbyManager. В котором для каждого пользователя создаётся случайное имя, создаётся подключение к серверам Photon Network. Данный класс предназначен для создания игрового лобби или подключения у уже существующему.

Игровые объекты «Пират» и «Корабль» содержат свойство PhotonView, которое служит для того чтобы синхронизировать параметры игровых объектов между пользователями, помимо этого, PhotonView позволяет определить какому пользователю принадлежит игровой объект. При инициализации игрового объекта у всех пользователей срабатывает метод OnPhotonInstantiate.

Для работы с игровыми событиями, которые были созданы для игрового приложения был создан отдельный класс RaiseEventManager, который запускает события для всех пользователей игрового лобби. Каждое событие принимает в качестве аргумента объект типа EventData, которое можно

дополнить с помощью собственно созданных данных. Для каждого отдельного события были созданы свои типы данных и описаны правила сериализации и десериализации в массив байт для передачи между пользователями.

3. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

Целью производства любого продукта является извлечение прибыли. Поэтому помимо проектирования и разработки продукта требуется также и его экономический анализ. Данный анализ позволяет понять, для кого создан продукт, насколько трудозатратно его производство, какие у него существуют конкуренты и как необходимо продавать продукт, чтобы извлечь из него максимальную выгоду.

Целью текущего раздела является проведение анализа продукта для установки его экономической ценности на рынке. Эта оценка необходима для поиска потенциальных покупателей, источников финансирования, установки цены за единицу продукта и успешности продажи продукта на рынке.

Выпускная квалификационная работа представляет собой разработку многопользовательского игрового приложения в жанре пошаговая стратегия. Продукт направлен на расширение сегмента рынка игровой индустрии. Продукт является частью развлекательного сегмента рынка и его основное предназначение представляет собой расширение видов досуга пользователей.

3.1. Оценка коммерческого потенциала и перспективности проведения научных исследований с позиции ресурсоэффективности и ресурсосбережения

3.1.1. Потенциальные потребители результатов исследования

Для анализа потенциальных потребителей необходимо рассмотреть целевой рынок и разбить его на сегменты.

Целевой рынок означает определенную группу людей, которым планируется продать свой товар или услугу.

Сегментирование – разделение покупателей на основные однородные группы, где каждой требуется наш товар или услуга.

Целевым рынком для нашего продукта будут обладатели ПК с операционной системой Windows увлекающиеся настольными играми или жанром стратегия. Это могут быть люди всех возрастов и любой национальности, так как размещение игрового приложения планируется осуществлять на всемирной игровой площадке – Steam.

3.1.2. Анализ конкурентных технических решений

Поскольку рынок находится в постоянном движении и каждый день появляются все новые продукты и решения, для успешных продаж продукт должен быть конкурентоспособен. Это означает, что нужно здраво оценивать свои сильные и слабые стороны и принимать решения в зависимости от них. Далее будет приведен сравнительный анализ разрабатываемого продукта с 2 конкурентными. Автор хочет заметить, что представленные игры тяжело назвать взаимозаменяемыми, сравнение с ними происходит по жанровой составляющей, так как полноценных аналогов разрабатываемого приложения не существует или они являются слишком неконкурентоспособными.

Для понимания приведем обозначения

Ф – фактический разрабатываемый продукт

К1 – серия Heroes of Might and Magic

К2 – серия Civilization

Анализ приведен в таблице 3.

Таблица - 3 Сравнительный анализ конкурентов

Критерии оценки	**Вес критерия**	**Баллы**			**Конкурентоспособность**		
		Бф	Бк1	Бк2	Кф	Кк1	Кк2
Технические критерии оценки ресурсоэффективности							
Разнообразие игровых механик	0,3	2	3	5	0,6	0,9	1,5
Сюжетная составляющая	0,1	0	5	0	0	0,5	0
Многопользовательская составляющая	0,1	5	3	5	0,5	0,3	0,5
Простота в освоении	0,1	5	3	2	0,5	0,3	0,2
Графическая составляющая	0,2	3	4	4	0,6	0,8	0,8
Экономические критерии оценки эффективности							
Стоимость продукта	0,1	5	3	1	0,5	0,3	0,1
Предполагаемый срок эксплуатации	0,1	2	5	5	0,2	0,5	0,5
Итого:	1				**2,9**	**3,6**	**3,6**

Анализ конкурентоспособности показал, что продукт немного отстаёт по техническим показателям от представленных конкурентов, по экономическим же показателям разрабатываемое приложение не уступает конкурентам и даже опережает их. Основными преимущественными факторами являются: наличие удобной многопользовательской составляющей, простота в освоении, а также низкая стоимость продукта.

Важно заметить, что представленные конкуренты разрабатывались крупными компаниями и успели приобрести статус культовых игровых приложений.

3.1.3. Технология QuaD

Технология QuaD (Quality advisor) представляет собой гибкий инструмент измерения характеристик, описывающих качество новой разработки и ее перспективность на рынке и позволяющие принимать решение целесообразности вложения денежных средств в проект.

Таблица 4 - Оценочная карта по технологии QuaD

Критерии оценки	Вес критерия	Баллы	Максимальный балл	Относительное значение	Средневзвешенное значение
Показатели оценки качества работы					
Разнообразие игровых механик	0,3	40	100	0,4	0,12
Сюжетная составляющая	0,1	0	100	0	0
Многопользовательская составляющая	0,1	100	100	1	0,1
Простота в освоении	0,1	100	100	1	0,1
Графическая составляющая	0,2	60	100	0,6	0,12
Показатели оценки коммерческого потенциала разработки					
Стоимость продукта	0,1	100	100	1	0,1
Предполагаемый срок эксплуатации	0,1	40	100	0,4	0,04
Итого:	**1**			**4,4**	**0,58**

.

Средневзвешенное значения показателя качества и перспективности разработки составляет 58. Этот средний показатель оценки по технологии

QuaD говорит о том, что перспективность проекта не слишком высокая, однако достаточная для того чтобы выводить продукт на рынок.

3.1.4. SWOT-анализ

SWOT-анализ – представляет собой комплексный анализ научно-исследовательского проекта. SWOT-анализ применяют для анализа внешней и внутренней среды проекта. Проводится в несколько этапов, где выявляются сильные и слабые стороны, возможности и угрозы.

Таблица 5 - SWOT-анализ

	Сильные стороны научно-исследовательского проекта:	**Слабые стороны научно-исследовательского проекта:**
	С1: Удобная многопользовательская составляющая С2: Более низкая стоимость по сравнению с аналогами С3: Простота в использовании С4: Быстрая скорость работы приложения С5: Использование современных технологий.	Сл1: Постоянное наличие доступа к сети Internet. Сл2: Отсутствие рекламной компании. Сл3: Несовершенство процесса разработки.

Возможности: В1: Увеличение спроса на пошаговые стратегии В2: Расширение функционала В3: Улучшение графической составляющей В4: Повышение стоимости конкурентных разработок	Благодаря уникальным технологиям и низкой стоимости, продукт можно легко расширять и улучшать, увеличивая спрос на продукт и клиентскую базу.	С помощью обратной связи от пользователей и финансирования, процесс можно довести до желаемого результата, а инструкции пользователей включить в проект, при первом использовании.
Угрозы: У1: Отсутствие спроса У2: Введение дополнительных государственных требований к сертификации продукции У3: Прекращение поддержки Photon	При прекращении поддержки некоторых технологий, которые используются в проекте, всегда можно перейти на альтернативные варианты. Введение дополнительных государственных требований можно решить, воспользовавшись услугами юристов.	Провести аудит по функционалу продукта, получить обратную связь от пользователей. Провести рекламную кампанию по популяризации продукта.

По результатам SWOT-анализа были выявлены сильные и слабые стороны научной разработки, а также ее угрозы и возможности. Были рассмотрены альтернативы продвижения продукта на рынке в зависимости от условий.

3.2. Планирование научно-исследовательских работ

3.2.1. Структура работ в рамках разработки программного продукта

В данном разделе необходимо составит перечень этапов и работ в рамках проведения научно-исследовательских работ, провести распределение исполнителей по видам работ.

Для планирования комплекса предполагаемых работ необходимо выполнить следующие задачи:

1. Определить структуры работ;
2. Определить участников каждой работы;
3. Установить продолжительность работ;
4. Построить график проведения работ;

Таблица 6 – Перечень этапов, работ и распределение исполнителей

Основные этапы	**№ работ**	**Содержание работ**	**Должность исполнителя**
Разработка технического задания	1	Составление и утверждения технического задания	Студент, научный руководитель
Проектирование системы	2	Проектирование архитектуры	Студент
	3	Выбор инструментов для разработки	Студент
	4	Проектирование моделей данных	Студент

	5	Проектирование бизнес-процессов	Студент
Разработка приложения	6	Разработка архитектуры	Студент
	7	Разработка моделей, бизнес-процессов	Студент
	8	Тестирование и отладка	Студент
Оформление отчета ВКР	9	Согласование выполненных работ с научным руководителем	Студент, научный руководитель
	10	Оформление отчета	Студент

3.2.2. Определение трудоемкости выполнения работ

Трудовые затраты являются основными затратами на разработку, поэтому необходимо определить их для каждого из исполнителей.

Ожидаемая продолжительность работ $t_{ож}$ с помощью экспертных оценок устанавливается согласно формуле:

$$t_{ожi} = \frac{3t_{min_i} + 2t_{max_i}}{5},$$

где t_{mini} – минимальная продолжительность работ в днях (оптимистическая оценка);

t_{maxi} – максимальная продолжительность работ в днях

(пессимистическая оценка).

Исходя из этого, определяется продолжительность каждой работы в рабочих днях T_p, с учетом параллельного выполнения некоторых работ участниками.

$$T_{pi} = \frac{t_{\text{ож } i}}{\text{Ч}_i},$$

где T_{pi} – продолжительность одной работы, раб. дн;

$t_{\text{ож i}}$ – ожидаемая трудоемкость выполнения одной работы, чел.-дн.;

Ч_i– численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, чел.

Таблица 7 – Трудозатраты на выполнение проекта

Этапы работ	**Исполнители %**	**Продолжительность работ, дни**			**Кол-во исполнителей**	T_{pi}
	Руководитель проекта Студент	t_{min}	t_{max}	$t_{ож}$		
Составление и утверждения технического задания	100	3	9	5,4	2	5,4
Проектирование архитектуры	100	7	18	11,4	1	11,4
Выбор инструментов для разработки	100	1	3	1,8	1	1,8
Проектирование моделей данных	100	1	2	1,4	1	1,4
Проектирование бизнес процессов	100	2	9	4,8	1	4,8
Разработка архитектуры	100	7	21	12,6	1	12,6
Разработка моделей, бизнес-процессов	100	10	30	18	1	18
Тестирование и отладка	100	5	10	7	1	7
Согласование выполненных работ с научным руководителем	100	1	3	1,8	2	1,8
Оформление отчета	100	21	50	32,6	1	32,6
Итого:				96,8		96,8

3.3. Бюджет научно-технического исследования

Для проекта по разработке программного обеспечения для обработки заявок технической поддержки необходимо оценить затраты по следующим статьям:

- Материальные затраты научно-исследовательской разработки;
- Основная и дополнительная заработная плата;
- Отчисления во внебюджетные фонды;
- Амортизационные расходы;
- Накладные расходы.

Это позволит понять какой начальный капитал требуется для производства продукта.

3.3.1 Расчет материальных затрат разрабатываемого приложения

В процессе разработки продукта использовался ноутбук компании Asus, поэтому необходимо рассчитать его амортизацию. Срок полезного использования ноутбука составляет 3 года.

Время написания ВКР – 6 месяцев. Рассчитаем норму амортизации:

$$Ан = 1/n * 100\% = 1/3 * 100\% = 33{,}33\%$$

Годовые амортизационные отчисления:

$$Аг = Стоимость*Ан = 96990 * 0{,}33 = 32330 \text{ рублей}$$

Ежемесячные амортизационные отчисления:

$$Ам = Аг / 12 = 18\ 147 / 12 = 2694{,}17 \text{ рублей}$$

Итого за срок написания ВКР амортизационные отчисления составят:

$$Ам * 5 = 2694 * 5 = 16165 \text{ рубль}$$

Таблица 8 – Список материальных затрат

№	Наименование	Кол-во единиц	Цена единицы, руб.	Общая стоимость, руб.	Амортизационные отчисления, руб.
1	Ноутбук Asus TUF Gaming A15	1	96990	96990	16165
Итого:				96990	16165

3.3.2. Основная заработная плата исполнителей

Таблица 9 – Баланс рабочего времени

Показатели рабочего времени	Студент	Научный руководитель
Календарные дни	365	365
Нерабочие дни (праздники / выходные)	66	66
Потери рабочего времени –отдых по болезни	14	14
Действительный годовой фонд рабочего времени	285	285

Затраты на заработную плату рассчитываются по формуле:

$$З_п = З_{осн} + З_{доп},$$

где З осн – Основная заработная плата, руб;

З доп – Дополнительная заработная плата, руб;

Основная заработная плата состоит из:

$$З_{осн} = З_{дн} * Т_р * (1 + К_{пр} + К_д) * К_р,$$

где Здн – среднедневная заработная плата, руб.;

Кпр – премиальный коэффициент (0,3);

Кд – коэффициент доплат и надбавок (0,3-0,5);

Кр – районный коэффициент (для Томска 1,3)

Тр – продолжительность работ, выполняемых работником, раб. Дни

Среднедневная заработная плата:

$$З_{дн} = \frac{З_м * М}{F_д}, где$$

Зм – месячный оклад работника, руб;

М – количество месяцев работы без отпуска в течение года (для 6-дневной рабочей недели М=10,4);

Fд – действительный годовой фонд рабочего времени персонала, раб. дн.

Согласно окладам, принятым в ТПУ оклад исполнителя (студента) - минимальный размер оклада (1 квалификационный уровень) -13000 руб, оклад сотрудника составляет – 30000 руб. В соответствии с данными цифрами составим таблицу расчета заработной платы для исполнителей темы:

Таблица 10 –Расчет основной заработной платы

Исполнители	Зм, руб.	Здн, руб.	Кпр	Кд	Кр	Тр	Зосн
Студент	13000,0	474,39	0,3	0,3	1,3	96	94725,39
Научный руководитель	30000,0	1094,74	0,3	0,3	1,3	7,2	16394,78
Итого:							**111120,2**

3.3.3. Дополнительная заработная плата исполнителей

Расчет дополнительной заработной платы ведется по формуле:

$$З_{доп} = k_{доп} * З_{осн},$$

где $k_{доп}$ – коэффициент дополнительной заработной платы (на стадии проектирования принимается равным 0,12 – 0,15). Возьмем верхнюю границу и рассчитаем дополнительную зарплату:

Таблица 11 – Расчет дополнительной заработной платы

Исполнитель	**Основная заработная плата, руб**	**Коэффициент дополнительной заработной платы**	**Дополнительная заработная плата, руб.**
Студент	94725,39	0,15	14208,81
Научный руководитель	16394,78	0,15	2459,22
Итого:			**16668,03**

3.3.4. Отчисления во внебюджетные фонды (страховые отчисления)

В данной статье отображаются обязательные отчисления в органы государственного социального страхования (ФСС), пенсионного фонда (ПС) и медицинского страхования (ФФОМС) от затрат на зарплату труда работников.

Величина отчислений рассчитывается по формуле:

$$З_{внеб} = k_{внеб} * (З_{осн} + З_{доп}), \text{ где}$$

$k_{внеб}$ – коэффициент отчислений на уплату во внебюджетные фонды (равен 0,302)

Таблица 12 – Расчет отчислений во внебюджетные фонды

Исполнители	**Основная и дополнительная заработные платы, руб**	**Коэффициент отчислений на уплату во внебюджетные фонды**	**Сумма отчислений, руб**
Студент	108934,2	0,302	32898,13

Научный руководитель	18854	0,302	5693,91
Итого			**38592,03**

3.3.5. Накладные расходы

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов: печать и ксерокопия материалов, оплата услуг связи, электроэнергии и т.д. Считается по формуле:

$$З_{накл} = \sum_{i=1}^{4} статей * К_{нр}, где$$

$К_{нр}$– Коэффициент накладных расходов (равен 0,16)

$З_{накл}$ = 24108,01

3.3.6. Формирование бюджета затрат научно-исследовательского проекта

Таблица 13 – Расчет бюджета затрат разработки

Наименование статьи	Сумма, руб	Примечание
Амортизационные затраты на спецоборудование	16165	Пункт 4.3.1
Затраты на основную заработную плату	111120,2	Пункт 4.3.2
Затраты на дополнительную заработную плату	16668,03	Пункт 4.3.3
Затраты на отчисление во внебюджетные фонды	38592,03	Пункт 4.3.4
Накладные расходы	29207,24	16% от суммы статей 1-4
Бюджет затрат НТИ	**211752,5**	Сумма всех пунктов

3.4. Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования

Определение эффективности проекта происходит на основе расчета интегрального показателя эффективности научного исследования. Его

нахождение связано с определения двух средневзвешенных величин: финансовой эффективности и ресурсоэффективности.

Интегральный показатель финансовой эффективности получают в ходе оценки бюджета затрат нескольких вариантов выполнения научного исследования.

Так как в рамках данной выпускной квалификационной работы не было предоставлено вариантов выполнения научной работы (в связи с спецификой темы), данные показатели будут считаться равными 5 из 5.

3.5. Выводы по разделу

В ходе выполнения раздела финансового менеджмента проведен анализ финансово-экономических аспектов разработки программной системы. Составлен перечень проводимых работ, их исполнителей и продолжительность выполнения этапов работ.

4. СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ

Введение

В рамках выпускной квалификационной работы создано многопользовательское трехмерное игровое приложение в жанре «Пошаговая стратегия», носящее название «Jackal».

Целью разрабатываемого продукта является воплощение оригинальной настольной игры «Jackal» в цифровом виде, предназначенный для расширения рынка игровой индустрии и для дальнейшей коммерциализации проекта.

Данное приложение представляет собой многопользовательскую игру в жанре пошаговая стратегия. Жанр пошаговая стратегия означает, что игровой процесс состоит из последовательных фиксированных моментов времени, именуемых ходами (или шагами), во время которых игроки совершают свои действия. Основная цель игроков заключается в поиске и

собирании монет на случайно-генерируемом поле, путём исследования игрового поля и сражения с другими игроками. Побеждает игрок, собравший наибольшее количество монет.

Проектирование и разработка данного игрового приложения велась с использованием ноутбука аудитории, расположенной по адресу Томская область, г. Томск, ул. Пирогова 18а, в связи с чем автор мог подвергнуться различным вредным факторам, которые рассмотрены далее.

4.1. Правовые и организационные вопросы обеспечения безопасности.

4.1.1. Специальные (характерные для проектируемой рабочей зоны) правовые нормы трудового законодательства.

Работа автора должна осуществляться в условиях отвечающим требованиям охраны труда. Таким образом, согласно статье 219 ТК РФ [1] каждый работник имеет право на рабочее место, соответствующее требованиям охраны труда, может отказаться от выполнения работ в случае возникновения опасности для его жизни и здоровья вследствие нарушения требований охраны труда, за исключением случаев, предусмотренных федеральными законами, до устранения такой опасности средствами индивидуальной и коллективной защиты.

Нормальная продолжительность рабочего времени не может превышать 40 часов в неделю. В течение рабочего дня (смены) работнику должен быть предоставлен перерыв для отдыха и питания продолжительностью не более двух часов и не менее 30 минут, который в рабочее время не включается. Правилами внутреннего трудового распорядка или трудовым договором может быть предусмотрено, что указанный перерыв может не предоставляться работнику, если установленная для него продолжительность ежедневной работы (смены) не превышает четырех часов (в ред. Федерального закона от 18.06.2017 N 125-ФЗ) [1].

4.1.2. Организационные мероприятия при компоновке рабочей зоны.

Разработка игрового приложения ведётся сидя за компьютерным столом. Согласно положению ГОСТ 12.2.032-78 [2] описывающий организацию рабочего места при выполнении работ сидя, конструкция рабочего пространства должна быть выполнена таким образом, чтобы обеспечивалось выполнение трудовых операций в пределах зоны досягаемости моторного поля в вертикальной и горизонтальной плоскостях. Выполнение трудовых операций "часто" и "очень часто" должно быть обеспечено в пределах зоны легкой досягаемости и оптимальной зоны моторного поля, приведенных на рисунке 31.

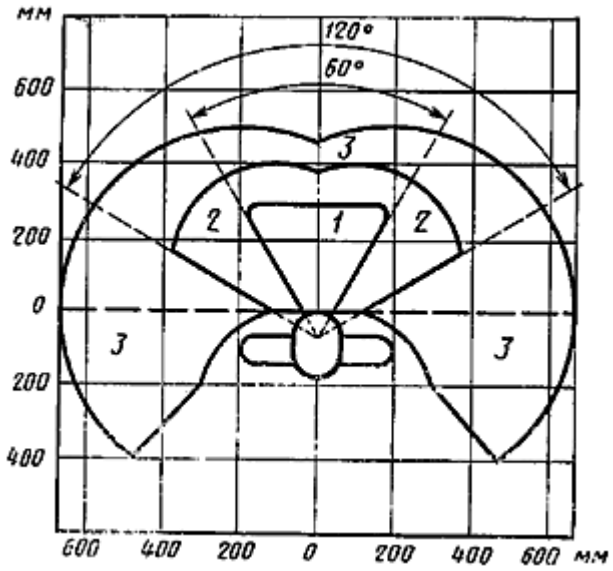

Рисунок 31 - Зоны для выполнения ручных операций и размещения органов управления 1 - зона для размещения наиболее важных и очень часто используемых органов управления (оптимальная зона моторного поля); 2 - зона для размещения часто используемых органов управления (зона легкой досягаемости моторного поля); 3 - зона для размещения редко используемых органов управления (зона досягаемости моторного поля)

Конструкцией производственного оборудования и рабочего места должно быть обеспечено оптимальное положение работающего, которое достигается регулированием высотой рабочей поверхности, сидения и пространства для ног. Регулируемые параметры следует выбирать в соответствии с ГОСТ 12.2.032-78 [2].

Органы управления должны размещаться таким образом, чтобы не было перекрещивания рук, монитор, клавиатура и компьютерная мышь должны быть расположены в зоне 1 (рисунок 1), так как они являются основными органами управления в представляемой разработке. Монитор ноутбука при этом допускается располагать в вертикальной плоскости под углом ±30° от нормальной линии взгляда и в горизонтальной плоскости под углом ±30° от сагиттальной плоскости согласно положению ГОСТ 12.2.032-78 [2].

4.2. Производственная безопасность.

4.2.1. Анализ выявленных вредных и опасных факторов.

Согласно производственным факторам ГОСТ 12.0.003-2015 ССБТ [3], на оператора ЭВМ в течение рабочего дня воздействует множество различных производственных факторов, каждый из которых влияет на производительность, работоспособность и физическое состояние.

Возможные опасные и вредные факторы представлены в таблице 1.

Таблица 14 – Возможные опасные и вредные факторы

Факторы (ГОСТ 12.0.0032015)	Этапы работ			Нормативные документы
	Разработка	Эксплуатация	Обслуживание	
1.Отклонение показателей микроклимата	+	+	+	1. ГОСТ 12.1.038-82 ССБТ Электробезопасность. Предельно допустимые уровни

2. Превышение уровня шума	+	+	+	напряжений прикосновения и токов [4]. 2. Правила устройства электроустановок ПУЭ [5].
3.Отсутствие или недостаток естественного света	+	+	+	3. СанПиН 2.2.4.548-96 Гигиенические требования к микроклимату производственных помещений. [6] 4. СанПиН 2.2.1/2.1.1.1278-03 Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий [7]. 5. СН 2.2.4/2.1.8.562-96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки [8]. 6. СанПиН СП 2.2.3670-20 Требования к организации работ с персональными электронными вычислительными машинами и копировально-множительной техникой [9].
4.Недостаточная освещенность рабочей зоны	+	+	+	
5.Повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека	+	+	+	

4.2.1.1 Микроклимат помещения

Человек постоянно находится в процессе теплового взаимодействия с окружающей его рабочее место средой. Температура, относительная влажность и скорость движения окружающего воздуха характеризуют процесс теплообмена. Данные параметры оказывают комплексное воздействие на процесс теплообмена на рабочем месте.

В соответствии с СанПин 2.2.3670-20, в производственных помещениях, в которых ведётся работа по разработке игрового приложения «Jackal» с использованием ЭВМ является основной и связана с нервно-эмоциональным напряжением, должны обеспечиваться оптимальные параметры микроклимата в соответствии с действующими санитарно-

эпидемиологическими нормативами микроклимата производственных помещений [9].

Исходя из СанПиН 2.2.4.548-96, значения температуры, влажности и скорости движения воздуха устанавливаются для рабочей зоны производственных помещений в зависимости от категории тяжести выполняемой работы, величины избытков явного тепла, выделяемого в помещении, и периода года [6].

В таблицах 2 и 3 соответственно приведены оптимальные и допустимые величины показателей микроклимата на рабочих местах производственных помещений для оператора ЭВМ. В данном случае работа относится к категории труда «легкая-1а».

Таблица 15 – Оптимальные величины показателей микроклимата

Период года	Температура воздуха, C^0	Температура поверхностей, C^0	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	21 – 23	20 – 24	40 – 60	0,1
Теплый	23 – 25	22 – 26	40 – 60	0,1

Таблица 16 – Допустимые величины показателей микроклимата

Период года	Температура воздуха, C^0		Температура поверхностей, C^0	Относительная влажность воздуха %	Скорость движения воздуха, м/с	
	Диапазон ниже оптимальных величин	Диапазон выше оптимальных величин			Для диапазона температур воздуха ниже оптимальных величин	Для диапазона температур воздуха выше оптимальных величин
Холодный	19,0 – 20,9	23,1 – 24,0	18,0 – 25,0	15 – 75	0,1	0,2

Теплый	20,0 – 21,9	24,1 – 28,0	19,0 – 29,0	15 – 75	0,1	0,3

4.2.1.2 Производственное освещение

Помещение, в котором ведётся разработка игрового приложения «Jackal» обладает двумя источниками искусственного освещения и одним источником естественного освещения.

Нормируется искусственное освещение в соответствии с СП 52.13330.2016 [10]. Величина минимальной освещенности относится к нормируемой количественной характеристике искусственного освещения. Качественной нормируемой характеристикой является показатель ослеплённости и дискомфорта, глубина пульсации освещенности (КЕ).

Качество получаемой информации во многом зависит от освещения: неудовлетворительное в количественном или качественном отношении освещение не только утомляет зрение, но и вызывает утомление организма в целом.

Светильники общего освещения в производственных помещениях должны иметь ограничения слепящего действия. Показатель ослеплённости не должен быть выше 20-80 единиц, безусловно, в зависимости от продолжительности работы и её зрительного разряда. Глубина пульсаций газоразрядных ламп выше 10-20 % не допускается, но это требование зависит от характера выполняемой работы.

На рисунке 32 представлен план размещения общего освещения относительно рабочего места в помещении общежития, расположенного по адресу г. Томск, ул. Пирогова 18а, с соответствующими размерами (в метрах). Согласно СанПиН 2.2.2/2.4.1340-03, норма освещённости для рассматриваемого рабочего места составляет 150 лк, нормированное значение

КЕО, равное 0,50%, должно быть обеспечено в центре помещения, коэффициент пульсации освещенности Кп должен быть не более 15% [9].

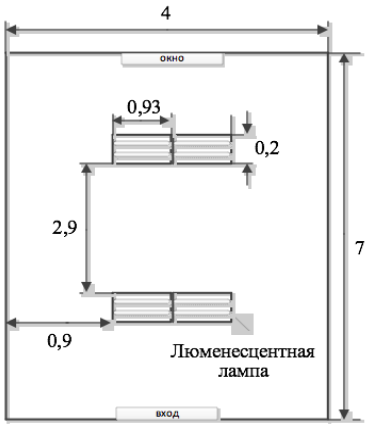

Рисунок 32. План размещения общего освещения

4.2.1.3 Электробезопасность

Опасное и вредное воздействия на людей электрического тока, электрической дуги и электромагнитных полей проявляются в виде электротравм и профессиональных заболеваний.

Степень опасного и вредного воздействия на человека электрического тока, электрической дуги и электромагнитных полей зависит от:

- рода и величины напряжения и тока;
- частоты электрического тока;
- пути тока через тело человека;
- продолжительности воздействия электрического тока или электромагнитного поля на организм человека; условий внешней среды.

Помещение офиса, в котором проводилась разработка проекта «Многопользовательское игровое приложение в жанре пошаговая стратегия», по электробезопасности относится к помещению без повышенной опасности, т.е. сухое, хорошо отапливаемое помещение с непроводящими ток полами, с температурой 18-21° и влажностью 40-50 %, согласно ГОСТ 12.1.019-2017 ССБТ [11].

Электробезопасность в помещениях должна обеспечиваться:

- конструкцией электроустановок;
- техническими способами и средствами защиты;
- организационными и техническими мероприятиями.

4.2.2. Обоснование мероприятий по снижению воздействия вредных и опасных факторов

Главным мероприятием по снижению воздействия вредных и опасных факторов является выбор подходящего помещения для выполнения работы, а также соблюдение требований техники безопасности.

В качестве помещения для исследования необходимо выбрать сухое, хорошо отапливаемое помещение с непроводящими ток полами, с температурой 18-21° и влажностью 40-50 %, согласно ГОСТ 12.1.019-2017 ССБТ [11].

Для разработки игрового приложения «Jackal» было выбрано помещение комнаты общежития по адресу г. Томск ул. Пирогова 18а. Оно удовлетворяет всем требованиям по защите разработчика от действия опасных и вредных факторов.

4.3. Экологическая безопасность.

4.3.1. Анализ влияния объекта разработки на окружающую среду.

В результате анализа влияния объекта исследования - «Многопользовательское игровое приложение в жанре пошаговая стратегия» на окружающую среду было установлено, что разрабатываемое приложение может причинить вред экологии только в том случае, если ЭВМ или приборы, используемые в ходе разработки, такие как люминесцентные лампы, выйдут из строя. При этом их придётся утилизировать и в ходе разложения будет причинён вред окружающей среде.

На сегодняшний день одним из самых распространенных источников ртутного загрязнения являются вышедшие из эксплуатации люминесцентные лампы. Каждая такая лампа, кроме стекла и алюминия, содержит около 60 мг ртути. Поэтому отслужившие свой срок люминесцентные лампы, а также другие приборы, содержащие ртуть, представляют собой опасный источник токсичных веществ.

Под хранением отходов понимается временное размещение их в специально отведенных для этого местах или объектах до их утилизации или удаления. Отработанные люминесцентные лампы, согласно Классификатору отходов ДК 005-96, относятся к отходам, которые сортируются и собираются отдельно, поэтому утилизация люминесцентных ламп и их хранение должны отвечать определенные требованиям.

4.3.2. Обоснование мероприятий по защите окружающей среды.

Хранение и удаление отходов (в данном случае – люминесцентных ламп) осуществляются в соответствии с требованиями экологической безопасности. Наполненную тару с отходами закрывают герметически стальной крышкой (при необходимости заваривают) и передают по договору специализированным предприятиям, имеющим лицензию на их утилизацию.

Эти правила оформлены на законодательном уровне как СанПиН 4607-88 [12], а также регулируются прочими государственными федеральными и региональными законами.

4.4. Безопасность в чрезвычайных ситуациях.

4.4.1. Анализ вероятных ЧС, которые могут возникнуть на рабочем месте при проведении разработки и эксплуатации приложения.

Наиболее типичной ЧС для помещения аудитории, в которой проводилась работа по разработке приложения «Jackal», является пожар. Он может возникнуть вследствие причин электрического и неэлектрического

характеров. К причинам электрического характера можно отнести короткое замыкание, искрение, статическое электричество. К причинам неэлектрического характера относится неосторожное обращение с огнём, курение, оставление без присмотра нагревательных приборов.

4.4.2. Обоснование мероприятий по предотвращению ЧС и разработка порядка действия в случае возникновения ЧС.

Одним из наиболее вероятных видов чрезвычайных ситуаций является пожар.

Всякий работник при обнаружении пожара должен:

- незамедлительно сообщить об это в пожарную охрану;
- принять меры по эвакуации людей, каких-либо материальных ценностей согласно плану эвакуации;
- отключить электроэнергию, приступить к тушению пожара первичными средствами пожаротушения.

Помещения аудитории, где ведётся разработка оснащены датчиками пламени пожарной сигнализации, по два на одну комнату. При срабатывании датчика, автоматически подаётся сигнал на пост ближайшей пожарной бригады. Помимо этого, коридоры общежития оснащены углекислотными огнетушителями ОУ-2 согласно требованиям ГОСТ 12.1.004-91 [13]. Профилактическими мерами по осуществлению пожарной безопасности является обязанность использовать удлинитель с предохранителем, а также использование огнестойких поверхностей для нагревательного оборудования, используемое в помещениях общежития.

При возникновении пожара существуют специальные маршруты для эвакуации людей из помещений общежития, которые обеспечивают наиболее безопасный выход из здания в случая возникновения пожара.

Актуальной чрезвычайной ситуацией на данный момент является пандемия, при поражении которой автор разрабатываемого приложения «Jackal» становится неработоспособным на неопределённый промежуток времени. Действовать при угрозе биолого-социальной чрезвычайной ситуации согласно ГОСТ 22.0.04-97 [14].

4.5. Выводы по разделу

В результате создания социальной части ВКР были изучены правовые и организационные вопросы обеспечения безопасности, изучены правовые нормы трудового законодательства и требования к компоновке рабочей зоны, в которой проводилась разработка приложения «Jackal».

Таким образом, выполняя все правила по организации безопасности на рабочем месте, описанные в данном разделе, осуществляется комфортная и безопасная разработка игрового приложения.

В случае возникновения чрезвычайных ситуаций различного характера, необходимо чётко выполнять порядок действий, описанный нормативными документами, на которые ссылается автор работы. Это позволяет избежать или снизить негативные последствия, которые могут быть причинены здоровью или жизни человека.

Заключение

В результате проделанной работы, было создано многопользовательское трехмерное игровое приложение в жанре пошаговая стратегия «Jackal» на игровом движке Unity.

В начале работы был проанализирован рынок игровой индустрии и определена успешность игрового проекта. Также было описано влияние игровой индустрии на общество и экономику.

Следующим шагом было составление технологии разработки и описание каждого из этапов. После чего было составлено техническое задание, спроектирована архитектура игрового приложения и выбраны технологии, которые будут использованы для разработки проекта.

На этапе разработки были созданы игровые модели с использованием воксельной графики, написана бизнес логика игрового процесса, а также настроен многопользовательский режим и созданы игровые сцены для навигации по приложению.

Разработанное игровое приложение успешно функционирует и доступно для скачивания с облачного хранилища Google Drive разработчика. Репозиторий проекта находится в открытом доступе на GitHub - https://github.com/LehanoLuck/Jackal.

Игровое приложения является полноценным расширением для жанра пошаговая стратегия и сектора игровой индустрии в целом.

Список литературы

1. Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 30.04.2021).

2. ГОСТ 12.2.032-78 ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования.

3. ГОСТ 12.0.003-2015 ССБТ. Опасные и вредные производственные факторы. Классификация.

4. ГОСТ 12.1.028-82 ССБТ. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов.

5. Правила устройства электроустановок ПУЭ.

6. СанПиН 2.2.4.548-96. Гигиенические требования к микроклимату производственных помещений.

7. СанПиН 2.2.1/2.1.1.1278-03. Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий.

8. СН 2.2.4/2.1.8.562-96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки.

9. СП 2.2.3670-20 Требования к организации работ с персональными электронными вычислительными машинами и копировально-множительной техникой.

10. СП 52.13330.2016. Естественное и искусственное освещение.

11. ГОСТ 12.1.019-2017 ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты.

12. СанПиН 4607-88. "Санитарные правила при работе со ртутью, ее соединениями и приборами с ртутным заполнением"

13. ГОСТ 12.1.004-91 ССБТ. Пожарная безопасность. Общие требования.

14. ГОСТ 22.0.04-97. Биолого-социальные чрезвычайные ситуации.

15. Официальный сайт MagicaVoxel [Электронный ресурс]. – Режим доступа: https://ephtracy.github.io/index.html?page=mv_main, свободный - Загл.с экрана.

16. Курс по Unity [Электронный ресурс]. – Режим доступа: https://docs.unity3d.com/Manual/BuildSettingsStandalone.html, свободный - Загл.с экрана.

17. Blender [Электронный ресурс]. – Режим доступа: https://www.blender.org, свободный - Загл.с экрана.

18. Документация Photon [Электронный ресурс]. – Режим доступа: https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro, свободный – Загл.с экрана.

Приложение А

Листинг: Генерация игрового поля

```
enum TilesCategory
   {
      Water,
      Ground,
      Coin,
      Arrow,
      Ice,
      Cave
   }
   public class MapSettings
   {
      public MapSettings()
      {
         TilesCategoryTable = new Hashtable();
         TilesCategoryTable.Add(TilesCategory.Ground, 10);
         TilesCategoryTable.Add(TilesCategory.Coin, 15);
         TilesCategoryTable.Add(TilesCategory.Arrow, 25);
         TilesCategoryTable.Add(TilesCategory.Ice,12);
         TilesCategoryTable.Add(TilesCategory.Cave,8);
      }

      public MapSettings(int width, int length, int coins, int ground)
      {
         int amount = width * length;
         int percents = coins + ground;
         double factor = (double)percents / (double)amount;
         int coinsCount = (int)((double)coins / factor);
         int groundCount = amount - coinsCount;
         TilesCategoryTable.Add(TilesCategory.Ground, groundCount);
         TilesCategoryTable.Add(TilesCategory.Coin, coinsCount);

         GroundWidth = width;
         GroundLength = length;
      }

      public Hashtable TilesCategoryTable = new Hashtable();

      public int LeftWaterSide = 2;
      public int RightWaterSide = 2;
      public int TopWaterSide = 2;
      public int BottomWaterSide = 2;
```

```
        public int GroundWidth = 8;
        public int GroundLength = 8;

        public int MapWidth => GroundWidth + LeftWaterSide + RightWaterSide;
        public int MapLength => GroundLength + TopWaterSide +
BottomWaterSide;
        public int CountGroundTiles => GroundLength * GroundWidth;

        public static MapSettings Default { get; private set; } = new MapSettings();
    }
public class MapMatrixManager
    {
        public static int CoinsCount;
        public static int ArrowsCount;
        public static int StartCoinsCount;

        public static int[][] GenerationMapMatrix;

        public static void SetGenerationMapMatrix(int[][] mapMatrix)
        {
            GenerationMapMatrix = mapMatrix;
            CoinsCount = mapMatrix.SelectMany(t => t).Where(t => t ==
2).ToArray().Length;
            StartCoinsCount = CoinsCount;
        }

        public static int[][] CreateRandomGenerationMapMatrix(MapSettings
mapSettings)
        {
            Random random = new Random();
            var mapMatrix = new int[mapSettings.MapWidth][];

            for (int i = 0; i < mapSettings.MapWidth; i++)
            {
                mapMatrix[i] = new int[mapSettings.MapLength];
            }

            int emptyTilesCount = mapSettings.CountGroundTiles;

            for (int i = 0; i < mapMatrix.Length; i++)
            {
                for (int j = 0; j < mapMatrix[i].Length; j++)
                {
                    if (i < mapSettings.LeftWaterSide ||
```

```
                    i >= mapSettings.MapWidth - mapSettings.RightWaterSide ||
                    j < mapSettings.TopWaterSide ||
                    j >= mapSettings.MapLength - mapSettings.BottomWaterSide)
                {
                    mapMatrix[i][j] = (int)TilesCategory.Water;
                }
                else
                {
                    //Генерируем случайное число в диапазоне от 0 до 1
                    var factor = random.NextDouble();
                    //Барьер, необходимый для того чтобы определить какой тайл
мы поставим на выбранную позицию
                    float categoryBarrier = 0;

                    foreach (TilesCategory category in
mapSettings.TilesCategoryTable.Keys)
                    {
                        //Вероятностое распределние в зависимости от количества
оставшихся свободных тайлов и кол-ва тайлов данной категории
                        categoryBarrier +=
(int)mapSettings.TilesCategoryTable[category] / (float)emptyTilesCount;

                        if (factor <= categoryBarrier)
                        {
                            mapMatrix[i][j] = (int)category;
                            mapSettings.TilesCategoryTable[category] =
(int)mapSettings.TilesCategoryTable[category] - 1;
                            emptyTilesCount--;
                            break;
                        }
                    }
                }
            }
        }

        return mapMatrix;
    }
}
public class MapManager : MonoBehaviour
{
    public CameraMovement camera;
    public int[][] GenerationMapMatrix;
    public ClosedTile ClosedTileTemplate;
    public GroundTile GroundTileTemplate;
```

```
public WaterTile WaterTileTemplate;
public CoinTile CoinTileTemplate;
public List<ArrowTile> ArrowTileTemplates;
public IceTile IceTileTemplate;
public CaveTile CaveTileTemplate;

public Dictionary<int, Ship> ShipsDictionary = new Dictionary<int, Ship>();

public Tile[][] Map;
private Hashtable TilesTable;
public List<Ship> Ships = new List<Ship>();
public Text Log;

void Start()
{
   TilesTable = new Hashtable();
   TilesTable.Add((int)TilesCategory.Ground, GroundTileTemplate);
   TilesTable.Add((int)TilesCategory.Coin, CoinTileTemplate);
   TilesTable.Add((int)TilesCategory.Arrow, ArrowTileTemplates);
   TilesTable.Add((int)TilesCategory.Water, WaterTileTemplate);
   TilesTable.Add((int)TilesCategory.Ice, IceTileTemplate);
   TilesTable.Add((int)TilesCategory.Cave, CaveTileTemplate);

   GenerationMapMatrix = MapMatrixManager.GenerationMapMatrix;
   GenerateMap();

   Log.text = $"Coins left - {MapMatrixManager.CoinsCount}";
   RaiseEventManager.EventMapManager = this;
}

public void GenerateMap()
{
   this.Map = new Tile[GenerationMapMatrix.Length][];
   for (byte i = 0; i < GenerationMapMatrix.Length; i++)
   {
      Map[i] = new Tile[GenerationMapMatrix[i].Length];
      for (byte j = 0; j < GenerationMapMatrix[i].Length; j++)
      {
         CreateTile(i, j);
      }
   }

   var length = Map.Length * 3.2f;
   var width = Map[0].Length * 3.2f;
```

```
        var zShift = 0f; //Добавочная длина, если длина больше ширины

        var xPos = length / 2 - 1.6f;

        if (length > width)
        {
            zShift = -(length - width) / 2;
            width = length;
        }

        var zDistance = (width / (1 - Mathf.Tan(Mathf.Deg2Rad *
camera.longPlaneAngle) / Mathf.Tan(Mathf.Deg2Rad *
camera.shortPlaneAngle)));
        var yPos = Mathf.Tan(Mathf.Deg2Rad * camera.longPlaneAngle) *
zDistance;
        var zPos = (width - 2.4f) - zDistance + zShift;

        camera.maxHeight = yPos;
        camera.startPosition = new Vector3(xPos, yPos, zPos);
        camera.SetPositionToStart();
        camera.mapSize = Mathf.Max(length, width);
    }

    private void PlaceTile(Tile tile, byte i, byte j)
    {
        float tileXSize = ClosedTileTemplate.transform.localScale.x * 3.2f;
        float tileYSize = ClosedTileTemplate.transform.localScale.y * 3.2f;

        tile.XPos = i;
        tile.YPos = j;
        tile.SetTransformPosition(new Vector3(tileXSize * i, 0, tileYSize * j));
        Map[i][j] = tile;
        tile.Map = this.Map;
    }

    private void CreateTile(byte i, byte j)
    {
        var item = TilesTable[GenerationMapMatrix[i][j]];
        object tile;
        if (item is List<ArrowTile>)
        {
            tile = GetRandomArrowTile(item as List<ArrowTile>, i, j);
        }
        else
```

```
        {
            tile = TilesTable[GenerationMapMatrix[i][j]];
        }
        if (tile is OpenedTile)
            CreateGroundTile(i, j, tile as OpenedTile);
        else
            CreateWaterTile(i, j);
    }

    private ArrowTile GetRandomArrowTile(List<ArrowTile> arrows, int i, int j)
    {
        //Реализация псевдорандома через случайное количество тайлов земли и монет
        int groundCount = GenerationMapMatrix.SelectMany(t => t).Where(t => t == (int)TilesCategory.Ground).ToArray().Length;
        int coinCount = GenerationMapMatrix.SelectMany(t => t).Where(t => t == (int)TilesCategory.Coin).ToArray().Length;

        int index = (i ^ j ^ groundCount ^ coinCount) % arrows.Count;

        return arrows[index];
    }

    private void CreateGroundTile(byte i, byte j, OpenedTile linkedTile)
    {
        ClosedTile tile = Instantiate(ClosedTileTemplate, this.transform);


        PlaceTile(tile, i, j);
        tile.LinkedTile = linkedTile;
    }

    private void CreateWaterTile(byte i, byte j)
    {
        WaterTile tile = Instantiate(WaterTileTemplate, this.transform);
        PlaceTile(tile, i, j);
    }

    public void AddShip(Ship ship)
    {
        byte id = (byte)ShipsDictionary.Count;
        ShipsDictionary.Add(id, ship);
        ship.Id = id;
    }
```

```
    public void EndGame(int actorNumber)
    {
        StartCoroutine(EndGameCoroutine(actorNumber));
    }

    private IEnumerator EndGameCoroutine(int actorNumber)
    {
        var winner = PhotonNetwork.CurrentRoom.Players.Values.FirstOrDefault(p
=> p.ActorNumber == actorNumber);
        Log.fontSize = 30;
        Log.text = $"{winner.NickName} is Win!!!";

        yield return new WaitForSeconds(3);

        var controller = FindObjectOfType<ConnectionController>();
        controller.LeaveRoom();
    }
```

Листинг: Ячейки игрового поля

```
public abstract class Tile : MonoBehaviour, IPointerEnterHandler,
IPointerExitHandler
{
    internal Color DefaultColor;
    internal Color EnterColor;

    void Start()
    {
        DefaultColor = GetComponentInChildren<MeshRenderer>().material.color;
        EnterColor = DefaultColor + new Color(0, 0.8f, 0, 0.5f);
    }

    #region MapPosition
    public Tile[][] Map { get; set; }
    public byte XPos { get; set; }
    public byte YPos { get; set; }

    public Vector3 fixedPosition;
    public Vector3 updatePosition;

    protected internal void SetMapPosition(Tile tile)
    {
        this.XPos = tile.XPos;
        this.YPos = tile.YPos;
        this.Map = tile.Map;
```

```
        this.Map[XPos][YPos] = this;
    }

    protected internal void SetTransformPosition(Vector3 position)
    {
        this.transform.position = position;
        this.fixedPosition = position;
        this.updatePosition = new Vector3(position.x, position.y + 0.25f, position.z);
    }
    #endregion

    #region PointerEvents
    public virtual void OnPointerEnter(PointerEventData eventData)
    {
        this.transform.position = updatePosition;
    }

    public virtual void OnPointerExit(PointerEventData eventData)
    {
        this.transform.position = fixedPosition;
    }
    #endregion

    public void ShowAvailableForMoveCells()
    {
        var tilesList = this.GetPossibleTiles();

        foreach (var tile in tilesList)
        {
            tile.GetComponentInChildren<MeshRenderer>().material.color =
tile.EnterColor;
        }
    }

    public void HideAvailableForMoveCells()
    {
        var tilesList = this.GetPossibleTiles();

        foreach (var tile in tilesList)
        {
            tile.GetComponentInChildren<MeshRenderer>().material.color =
tile.DefaultColor;
        }
    }
```

```
    protected virtual IEnumerable<Tile> GetPossibleTiles()
    {
        var tiles = Map.SelectMany(g => g.ToArray()).Where(t =>
IsPossibleForMove(t));
        return tiles;
    }

    public virtual bool IsPossibleForMove(Tile targetTile)
    {
        return (targetTile != this &&
                (Math.Abs(this.XPos - targetTile.XPos) < 2) &&
                (Math.Abs(this.YPos - targetTile.YPos) < 2) &&
                IsCanMoveOnWaterTile(targetTile));
    }

    protected bool IsCanMoveOnWaterTile(Tile targetTile)
    {
        if(!(targetTile is WaterTile))
        {
            return true;
        }
        else
        {
            return ((WaterTile)targetTile).PirateShip != null;
        }
    }
    public abstract void EnterPirate(Pirate pirate);
}
public abstract class BasicTile: Tile, IPirateInteractor,IPointerDownHandler
{
    public List<Pirate> Pirates { get; set; } = new List<Pirate>();
    public int MaxPirateSize = 5;
    public bool isHavePirates => Pirates.Count > 0;
    public int ShipId;

    public override void EnterPirate(Pirate pirate)
    {
        if (isHavePirates)
        {
            this.TryAttack(pirate);
        }
        this.AddPirate(pirate);
        this.ShipId = pirate.Ship.Id;
```

```
    PlacePirateOnTile();
}

public virtual void LeavePirate(Pirate pirate)
{
    this.Pirates.Remove(pirate);
    PlacePirateOnTile();
}

public void LeaveAllPirates()
{
    this.Pirates.Clear();
}

public void KillAllPirates()
{
    foreach(var pirate in Pirates)
    {
        pirate.Die();
    }
    LeaveAllPirates();
}

public virtual void TryAttack(Pirate pirate)
{
    if (ShipId != pirate.Ship.Id)
    {
        KillAllPirates();

        if (pirate.IsMoveWithCoin())
            pirate.DropCoin();
    }
}

public void AddPirate(Pirate pirate)
{
    pirate.CurrentTile = this;
    this.Pirates.Add(pirate);
}

public virtual void PlacePirateOnTile()
{
    int count = Pirates.Count;
```

```
        //Коэффициент для изменения размера пирата
        float factor = (float)(1 / Math.Pow(Math.Log(Math.E * count), 0.4));
        var localScale = new Vector3(factor, factor, factor);

        float radius = count == 1 ? 0 : this.transform.localScale.x * 0.85f;

        for (int i = 0; i < count; i++)
        {
            Pirates[i].transform.localScale = localScale;

            float value = i * 2f * Mathf.PI / count;

            float x = this.transform.position.x + radius * Mathf.Sin(value);
            float z = this.transform.position.z + radius * Mathf.Cos(value);

            Pirates[i].transform.position = new Vector3(x, this.transform.position.y, z);
        }
    }

    public abstract void AddCoin(Coin coin);

    public override void OnPointerEnter(PointerEventData eventData)
    {
        if(eventData.pointerDrag &&
eventData.pointerDrag.GetComponent<Pirate>())
        {
            foreach (var pirate in Pirates)
            {
                pirate.Collider.enabled = false;
            }
        }
        else
        {
            foreach (var pirate in Pirates)
            {
                pirate.Collider.enabled = true;
            }
        }
        base.OnPointerEnter(eventData);
    }

    public override void OnPointerExit(PointerEventData eventData)
    {
```

```
        base.OnPointerExit(eventData);
    }

    public virtual void OnPointerDown(PointerEventData eventData)
    {
        foreach (var pirate in Pirates)
        {
            pirate.Collider.enabled = true;
        }
    }
}
public class ClosedTile : Tile
{
    public OpenedTile LinkedTile;

    public OpenedTile OpenTile()
    {
        var tile = Instantiate(LinkedTile, this.transform.parent);
        tile.SetMapPosition(this);
        tile.SetTransformPosition(this.fixedPosition);

        Destroy(gameObject);
        return tile;
    }

    public override void EnterPirate(Pirate pirate)
    {
        var tile = OpenTile();
        tile.Open(pirate);
    }
}
public abstract class OpenedTile: BasicTile
{
    public virtual void DoAction()
    {
    }

    public virtual void Open(Pirate pirate)
    {
        EnterPirate(pirate);
        DoAction();
    }
}
public class GroundTile : OpenedTile, ICoinInteractor, IPirateInteractor
```

```
{
    public Stack<Coin> Coins { get; set; } = new Stack<Coin>();

    public override void EnterPirate(Pirate pirate)
    {
        base.EnterPirate(pirate);
        if (pirate.IsMoveWithCoin())
        {
            pirate.SelfCoin.Move(this);
        }
        else
        {
            pirate.SelfCoin = this.PeekCoin();
        }
    }

    public override void AddCoin(Coin coin)
    {
        this.Coins.Push(coin);
    }

    public Coin PopCoin()
    {
        return Coins.Pop();
    }

    public Coin PeekCoin()
    {
        if (Coins.Count > 0)
        {
            return Coins.Peek();
        }
        else
        {
            return null;
        }
    }

    public bool IsHaveCoins()
    {
        return Coins.Count > 0;
    }
}
public class CoinTile : GroundTile
```

```
{
    public Coin coinTemplate;

    public override void DoAction()
    {
        Coin coin = Instantiate(coinTemplate, this.transform.parent);
        coin.transform.localScale = coin.transform.localScale * 0.6f;
        coin.transform.position = new Vector3(this.transform.position.x,
this.transform.position.y + 0.1f, this.transform.position.z);
        this.Coins.Push(coin);
        coin.CurrentTile = this;
    }

    public override void Open(Pirate pirate)
    {
        DoAction();
        EnterPirate(pirate);
    }
}
public abstract class ArrowTile : GroundTile
{
    protected int rotationAngle;

    public override void EnterPirate(Pirate pirate)
    {
        base.EnterPirate(pirate);

        pirate.isActive = true;
    }

    public override void Open(Pirate pirate)
    {
        SetRandomRotationAngle(pirate);
        EnterPirate(pirate);
    }

    private void SetRandomRotationAngle(Pirate pirate)
    {
        this.rotationAngle = (XPos ^ YPos ^ pirate.Ship.CurrentTile.YPos ^
pirate.Ship.CurrentTile.XPos) % 4;
    }

    public (int, int) RotateTile(int angle, int x, int y)
    {
```

```
        int xPos;
        int yPos;

        switch (angle)
        {
            case 1:
                xPos = y;
                yPos = -x;
                break;
            case 2:
                xPos = -x;
                yPos = -y;
                break;
            case 3:
                yPos = x;
                xPos = -y;
                break;
            default:
                xPos = x;
                yPos = y;
                break;
        }

        return (xPos, yPos);
    }
}
public class HorseTile: ArrowTile
{
    public override void EnterPirate(Pirate pirate)
    {
        base.EnterPirate(pirate);
        pirate.isActive = true;
    }

    public override bool IsPossibleForMove(Tile targetTile)
    {
        return ((Mathf.Abs(this.XPos - targetTile.XPos) == 2 &&
Mathf.Abs(this.YPos - targetTile.YPos) == 1) ||
                (Mathf.Abs(this.XPos - targetTile.XPos) == 1 && Mathf.Abs(this.YPos -
targetTile.YPos) == 2));
    }
}
public enum CaveStatus
{
```

```
    canEnter,
    canLeave
}

public class CaveTile: GroundTile
{
    public ParticleSystem CaveActiveLight;

    public CaveStatus CaveStatus = CaveStatus.canEnter;

    private void SetActive(bool canEnter)
    {
        if (canEnter)
        {
            CaveStatus = CaveStatus.canEnter;
            CaveActiveLight.Play();
        }
        else
        {
            CaveStatus = CaveStatus.canLeave;
            CaveActiveLight.Stop();
        }
    }

    public override void EnterPirate(Pirate pirate)
    {
        base.EnterPirate(pirate);

        if (CanMoveAfterEntering() && CaveStatus == CaveStatus.canEnter)
        {
            pirate.isActive = true;
        }
    }

    private bool CanMoveAfterEntering()
    {
        var otherCaves = GetOtherCaves();

        if (otherCaves.Any())
        {
            foreach (CaveTile cave in otherCaves)
            {
                if (cave.CaveStatus == CaveStatus.canEnter)
                {
```

```
                if (!(cave.isHavePirates) || otherCaves.Count() == 1)
                {
                    return true;
                }
            }
        }
    }

    return false;
}

public override void LeavePirate(Pirate pirate)
{
    base.LeavePirate(pirate);

    if (CaveStatus == CaveStatus.canEnter)
    {
        RelocateToAnotherCave(pirate);
    }
    else
    {
        SetActive(true);
    }
}

private void RelocatePirate(Pirate pirate)
{
    base.LeavePirate(pirate);
}

public IEnumerable<Tile> GetOtherCaves()
{
    IEnumerable<Tile> caves = Map.SelectMany(t => t).Where(t => t is CaveTile
&& t != this);
    return caves;
}

public void RelocateToAnotherCave(Pirate pirate)
{
    var targetTile = (CaveTile)pirate.TargetTile;

    var other = targetTile.Pirates.FirstOrDefault();

    targetTile.SetActive(false);
```

```
        if (other)
        {
            SetActive(false);
            other.TargetTile = this;
            targetTile.RelocatePirate(other);
            this.EnterPirate(other);
        }
    }

    public override bool IsPossibleForMove(Tile targetTile)
    {
        if (CaveStatus == CaveStatus.canEnter)
        {
            var caves = GetOtherCaves();
            if (caves.Any())
            {
                foreach (var cave in caves)
                {
                    if (cave.XPos == targetTile.XPos && cave.YPos == targetTile.YPos
&& CanEnterInCave())
                    {
                        return true;
                    }
                }
            }

            return false;
        }
        else
        {
            return base.IsPossibleForMove(targetTile);
        }

        bool CanEnterInCave()
        {
            var caveTile = ((CaveTile) targetTile);

            return (caveTile.CaveStatus == CaveStatus.canEnter);
        }
    }

}
```

Листинг: Drag&Drop

```
public void OnDrag(PointerEventData eventData)
{
    if (!photonView.IsMine || !isMyTurn || !isActive)
        return;

    var groundPlane = new Plane(Vector3.up, Vector3.zero);

    Ray ray = GameCamera.ScreenPointToRay(Input.mousePosition);

    if (groundPlane.Raycast(ray, out float position))
    {
        Vector3 worldPosition = ray.GetPoint(position);

        trajectoryMovement.ShowTrajectory(worldPosition);
    }
}

public void OnBeginDrag(PointerEventData eventData)
{
    if (!photonView.IsMine || !isMyTurn || !isActive)
        return;

    CurrentTile.ShowAvailableForMoveCells();
}

public void OnEndDrag(PointerEventData eventData)
{
    if (!photonView.IsMine || !isMyTurn || !isActive)
        return;

    isMoveWithCoin = pirateMovableOptions.IsMoveWithCoin;
    CurrentTile.HideAvailableForMoveCells();
    var isCanMove = IsCanMoveOnTile(eventData);

    if (isCanMove)
    {
        Deactivate();
        var pirateMovementData = new PirateMovementData
        {
            Id = this.Id,
            ShipId = this.Ship.Id,
            XPos = TargetTile.XPos,
            YPos = TargetTile.YPos,
            IsMoveWithCoin = IsMoveWithCoin()
```

```
        };

        RaiseEventManager.RaiseMovePirateEvent(pirateMovementData);
        MoveOnTile();

        if (!isActive)
        {
            StepByStepSystem.StartNextTurn();
            Activate();
        }
    }
    this.trajectoryMovement.HideTrajectory();
}
```

Листинг: Корабль, Пират, Монета

```
private bool IsCanMoveOnTile(PointerEventData eventData)
{
    TargetTile = GetTargetTile();

    if(TargetTile)
    {
        return CurrentTile.IsPossibleForMove(TargetTile);
    }
    else
    {
        return false;
    }

    Tile GetTargetTile()
    {
        var point = eventData.pointerEnter;
        if (point)
        {
            if (point.GetComponent<Tile>())
            {
                return eventData.pointerEnter.GetComponent<Tile>();
            }
        }
        return null;
    }
}

public bool IsMoveWithCoin()
{
    return isMoveWithCoin && SelfCoin != null;
```

```
}

public void MoveOnTile(PirateMovementData data, Tile targetTile)
{
   TargetTile = targetTile;

   MoveOnTile();
}

public void MoveOnTile()
{
   CurrentTile.LeavePirate(this);
   TargetTile.EnterPirate(this);
}

public void Die()
{
   DropCoin();
   this.Ship.CurrentTile.EnterPirate(this);
}

public void DieForever()
{
   Destroy(this.gameObject);
}

private void AddPirateOnShip(int count)
{
   for (int i = 0; i < count; i++)
   {
      byte pirateId = (byte)i;
      Pirate pirate = PhotonNetwork.Instantiate(PirateTemplate.name,
this.transform.position, Quaternion.identity, 0,
         new object[] { pirateId, this.Id }).GetComponent<Pirate>();
   }
}

public void EnterPirate(Pirate pirate)
{
   this.Pirates.Add(pirate);

   if (pirate.IsMoveWithCoin())
   {
      AddCoin(pirate.SelfCoin);
```

```
    }
}

public void KillPirateForever(Pirate pirate)
{
    if (pirate.IsMoveWithCoin())
    {
        Destroy(pirate.SelfCoin.gameObject);
    }
    pirate.DieForever();
}
public class Coin : MonoBehaviour
{
    public ICoinInteractor CurrentTile;

    public void Move(GroundTile tile)
    {
        CurrentTile.PopCoin();
        this.CurrentTile = tile;
        tile.AddCoin(this);
        this.transform.position = tile.transform.position;
    }
}
```

Листинг: Многопользовательский режим

```
public class LobbyManager : MonoBehaviourPunCallbacks
{
    public InputField LobbyName;

    public GameObject LobbyModal;

    public Text LogText;
    // Start is called before the first frame update
    void Start()
    {
        PhotonNetwork.LocalPlayer.NickName = "Player " + Random.Range(1000,
9999);

        Log("Player's name is set to " + PhotonNetwork.LocalPlayer.NickName);
        PhotonNetwork.ConnectUsingSettings();

        RaiseEventManager.ActivateCallbacks();
    }
```

```
private void Log(string message)
{
    Debug.Log(message);
    LogText.text += "\n";
    LogText.text += message;
}

public void ShowCreateLobbyModalDialog()
{
    LobbyModal.SetActive(true);
}

public void JoinRoom()
{
    JoinRoomByName(LobbyName.text);
    LobbyModal.SetActive(false);
}

public void JoinRoomByName(string roomName)
{
    RoomOptions roomOptions = new RoomOptions();
    roomOptions.IsVisible = false;
    roomOptions.MaxPlayers = 4;
    PhotonNetwork.JoinOrCreateRoom(roomName, roomOptions, null);
}

public override void OnCreatedRoom()
{
    Log($"Lobby with name {LobbyName.text} has been created");
}

public override void OnJoinedRoom()
{
    Log($"Joined in lobby with name {LobbyName.text}");
    PhotonNetwork.LoadLevel("LobbyScene");
}

public override void OnJoinRandomFailed(short returnCode, string message)
{
    Log("Failed to Join Room");
}

public override void OnConnected()
```

```
    {
        Log("Connected to Photon!");
    }

    public override void OnConnectedToMaster()
    {
        Log("Connected to Master!");
    }
}
public class ConnectionController : MonoBehaviourPunCallbacks
{
    public Transform EnemyPanel;
    public GameObject Template;
    public Text Log;

    public void LeaveRoom()
    {
        PhotonNetwork.LeaveRoom();
    }

    public override void OnLeftRoom()
    {
        //Когда текущий игрок (мы) покидает комнату
        SceneManager.LoadScene(0);
    }

    void Start()
    {
        foreach (var player in PhotonNetwork.PlayerListOthers)
        {
            var enemy = Instantiate(Template, EnemyPanel);

enemy.GetComponent<GamePlayer>().SetPlayerName(player.ActorNumber);
        }
    }

    public override void OnPlayerLeftRoom(Player otherPlayer)
    {
        Debug.LogFormat("Player {0} left the room", otherPlayer);
    }

    public void LogData(string message)
    {
        Log.text += message;
```

```
        Debug.LogFormat(message);
    }
}
public class RaiseEventManager: IOnEventCallback
{
    public static MapManager EventMapManager;
    public static ShipManager EventShipManager;
    private static RaiseEventManager EventManager { get; set; }

    private const int StartGameEvent = 1;
    private const int MoveShipEvent = 2;
    private const int CreateShipEvent = 3;
    private const int MovePirateEvent = 4;
    private const int SetNewQueuePlayersEvent = 5;
    private const int EndGameEvent = 6;

    private const int ShipMovementCode = 1;
    private const int PirateMovementCode = 2;

    private static RaiseEventOptions raiseEventOptions = new RaiseEventOptions {
Receivers = ReceiverGroup.All };
    private static RaiseEventOptions raiseEventOptionsForOther = new
RaiseEventOptions { Receivers = ReceiverGroup.Others };
    private static SendOptions sendOptions = new SendOptions { Reliability = true
};

    public static void ActivateCallbacks()
    {
        if (EventManager == null)
        {
            EventManager = new RaiseEventManager();
            PhotonNetwork.AddCallbackTarget(EventManager);

            PhotonPeer.RegisterType(typeof(ShipMovementData),
ShipMovementCode, ShipMovementData.Serialize,
ShipMovementData.Deserialize);
            PhotonPeer.RegisterType(typeof(PirateMovementData),
PirateMovementCode, PirateMovementData.Serialize,
PirateMovementData.Deserialize);
        }
    }

    public static void RaiseStartGameEvent(MapSettings settings)
    {
```

```
        int[][] mapMatrix =
MapMatrixManager.CreateRandomGenerationMapMatrix(settings);

        PhotonNetwork.RaiseEvent(StartGameEvent, mapMatrix, raiseEventOptions,
sendOptions);
    }

    public static void RaiseMoveShipEvent(ShipMovementData
shipMovementData)
    {
        PhotonNetwork.RaiseEvent(MoveShipEvent, shipMovementData,
raiseEventOptionsForOther, sendOptions);
    }

    public static void RaiseCreateShipEvent(ShipMovementData
shipMovementData)
    {
        PhotonNetwork.RaiseEvent(CreateShipEvent, shipMovementData,
raiseEventOptions, sendOptions);
    }

    public static void RaiseMovePirateEvent(PirateMovementData
pirateMovementData)
    {
        PhotonNetwork.RaiseEvent(MovePirateEvent, pirateMovementData,
raiseEventOptionsForOther, sendOptions);
    }

    public static void RaiseSetNewQueuePlayersEvent(int[] actorNumbers)
    {
        PhotonNetwork.RaiseEvent(SetNewQueuePlayersEvent, actorNumbers,
raiseEventOptionsForOther, sendOptions);
    }

    public static void RaiseEndGameEvent(int actorNumber)
    {
        PhotonNetwork.RaiseEvent(EndGameEvent, actorNumber,
raiseEventOptions, sendOptions);
    }

    public void OnEvent(EventData photonEvent)
    {
        switch (photonEvent.Code)
        {
```

```
        case StartGameEvent:

MapMatrixManager.SetGenerationMapMatrix((int[][])photonEvent.CustomData);
            PhotonNetwork.LoadLevel("SampleScene");
            break;
        case MoveShipEvent:
            var shipMovementData =
(ShipMovementData)photonEvent.CustomData;
            var ship = EventMapManager.ShipsDictionary[shipMovementData.Id];
            var tile =
EventMapManager.Map[shipMovementData.XPos][shipMovementData.YPos];
            ship.MoveOnTile(tile);
            break;
        case CreateShipEvent:
            shipMovementData = (ShipMovementData)photonEvent.CustomData;
            ship = EventMapManager.ShipsDictionary[shipMovementData.Id];
            tile =
EventMapManager.Map[shipMovementData.XPos][shipMovementData.YPos];
            ship.CreateShip(tile);
            break;
        case MovePirateEvent:
            var pirateMovementData =
(PirateMovementData)photonEvent.CustomData;
            ship = EventMapManager.ShipsDictionary[pirateMovementData.ShipId];
            var pirate = ship.ShipPirates[pirateMovementData.Id];
            tile =
EventMapManager.Map[pirateMovementData.XPos][pirateMovementData.YPos];
            pirate.isMoveWithCoin = pirateMovementData.IsMoveWithCoin;
            pirate.MoveOnTile(pirateMovementData, tile);
            break;
        case SetNewQueuePlayersEvent:
            var actorNumbers = (int[])photonEvent.CustomData;

            List<Player> players = new List<Player>();
            foreach(int actorNumber in actorNumbers)
            {
                var player = PhotonNetwork.PlayerList.First(p => p.ActorNumber ==
actorNumber);
                players.Add(player);
            }

            StepByStepSystem.Players = new Queue<Player>(players);
            break;
        case EndGameEvent:
```

```
            var number = (int)photonEvent.CustomData;
            EventMapManager.EndGame(number);
            break;
        }
    }
}
```