**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ ………………………………………………………………………..6](#_Toc168255314)

[1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ 7](#_Toc168255315)

[1.1 Описание предметной области 7](#_Toc168255316)

[1.2 Сравнительный анализ аналогов и прототипов 8](#_Toc168255317)

[1.3 Постановка задачи проектирования 12](#_Toc168255318)

[1.4 Анализ требования к проекту 13](#_Toc168255319)

[1.5 Выбор и обоснование средств и методов решения задач 13](#_Toc168255320)

[1.6 Разработка технического задания 16](#_Toc168255321)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 17](#_Toc168255322)

[2.1 Функциональная структура приложения 17](#_Toc168255323)

[2.2 Проектирование диаграммы вариантов использования 18](#_Toc168255324)

[2.3 Информационное обеспечение проекта 19](#_Toc168255325)

[2.5 Разработка концепции пользовательского интерфейса 21](#_Toc168255327)

[2.6 Безопасность и защита данных 22](#_Toc168255328)

[3 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ 24](#_Toc168255329)

[3.1 Результаты реализации серверной части 24](#_Toc168255330)

[3.2 Результаты реализации клиентской части 24](#_Toc168255330)

[3.3 Разработка программной документации 64](#_Toc168255331)

[3.4 Тестирование 65](#_Toc168255332)

[ЗАКЛЮЧЕНИЕ 80](#_Toc168255341)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 81](#_Toc168255342)

[ПРИЛОЖЕНИЕ А (обязательное) Техническое задание 83](#_Toc168255343)

[ПРИЛОЖЕНИЕ Б (справочное) Диаграмма вариантов использования 87](#_Toc168255353)

[ПРИЛОЖЕНИЕ В (обязательное) Программа и методика испытаний 90](#_Toc168255355)

# ВВЕДЕНие

В современном мире мобильные игры занимают особое место в культуре, играя ключевую роль в повседневной жизни миллионов людей. Их важность простирается далеко за рамки простого развлечения, оказывая влияние на социальные, психологические и экономические аспекты современного общества. Игры стали значимой формой досуга для широкого круга пользователей, так как мобильные устройства обеспечивают их доступность в любое время и в любом месте. Это создает уникальную возможность расслабиться, отвлечься и погрузиться в виртуальные миры даже во время коротких перерывов между учебой, работой или личными делами.

Немаловажно, что гейминг оказывает значительное влияние на развитие технологий. Их постоянное совершенствование стимулирует инновации в области графики, виртуальной и дополненной реальности, искусственного интеллекта и других технологических областях. Таким образом, игры становятся катализатором технического прогресса. Наконец, игровые приложения оказывают экономическое воздействие. Индустрия мобильных игр приносит значительные доходы, стимулируя развитие инноваций и создание новых рабочих мест. Многие разработчики и компании находят в играх источник прибыли, что также способствует развитию цифровой экономики.

Мировой рынок игр, включая мобильные и другие платформы, в 2022 году составил более 184 млрд. долл. По прогнозам аналитиков Newzoo, к 2025 году он вырастет до 211 млрд. долл.

При этом количество скачиваний мобильных игр оценивается десятками миллиардов. Например, только в Google Play Store насчитывается более 2 млн. игр. Эта статистика наглядно свидетельствует о том, что роль мобильных игр как важной составляющей современной культуры и развлечений продолжает активно расширяться. Цифры доказывают, что игровые приложения не только обрели широкую популярность, но и оказывают значительное влияние на повседневную жизнь миллионов людей по всему миру. [1]

Среди прочих выделяется наиболее популярный в последние несколько лет жанр гибридно-казуальных игр, стремительно ворвавшийся в индустрию. Гибридно-казуальная игра – это относительно новый стиль мобильных игр, сочетающий в себе простоту гиперказуального игрового процесса с более сложной механикой развития, которая также понравится аудитории среднего уровня. Например, в игре может быть простой базовый цикл, позволяющий научиться играть в секунды, сочетая при этом более глубокую метаигру, конкурентные социальные функции и надежную внутриигровую экономику.

Жанр развивался по мере того, как разработчики начали разнообразить гиперказуальные игры и менять свое дизайнерское мышление, чтобы удерживать пользователей на более длительный период времени. [2]

В рамках данного дипломного проекта была поставлена цель создания мобильного игрового приложения «Jess` Stories» в жанре гибридно-казуальных игр, в связи с чем были выделены определенные задачи анализа, проектирования и разработки, которые будут описаны в следующих главах.

# АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ

## Описание предметной области

В связи с тем, что данное игровое приложение будет разрабатываться в жанре гибридно-казуальных игр, при его создании будут учтены принципы, описывающие суть гиперказуальных и казуальных игр.

Гиперказуальными обычно называют высоко реиграбельные игры, в которых игрок выполняет простые действия. Например, это игры 2042, Flappy Bird и hole.io.

Некоторые гиперказуальные игры выглядят или играются нетипично, но всегда соответствуют нескольким или всем перечисленным характеристикам:

* Ориентировка на массового игрока. Такая игра использует понятные большинству темы, чтобы снизить порог входа: животные или спорт, рутинная работа или конфликт между людьми, геометрические формы или базовые природные элементы, вроде воды и огня
* Геймплей начинается сразу после запуска. Во многих играх перед началом игры пользователю предлагают пройти обучение или настроить игру под себя. В гиперказуальных играх пользователя от игры отделяет максимум кнопка «Играть»
* Простой геймплей. Правила игры должны быть настолько очевидны для игрока, чтобы обойтись без любых подсказок. По этой причине гиперказуальные игры часто основаны на одной механике
* Непрерывная концентрация внимания на процессе игры. Игровые сессии гиперказуальных длятся быстро, поэтому для включения игрока в состояние потока не нужно балансировать игровой темп интенсивными и спокойными частями — игра должна требовать внимание игрока непрерывно.
* Короткие игровые сессии. В гиперказуальные игры заходят ситуативно: в метро, очереди или такси – поэтому игрок должен успеть быстро завершить сессию. Гиперказуальные игры обычно основаны на множестве коротких сессий, чем на небольшом количестве длинных
* Счётчик очков. Гиперказуальные игры обычно ведут счёт заработанных пользователем очков за сессию, чтобы мотивировать игрока побить свой рекорд не откладывая на потом.
* Простая и приятная графика. Игроки заходят в гиперказуальные игры, чтобы расслабиться и отвлечься, поэтому такие игры обычно эстетически приятны и не используют сложные визуальные формы и образы. [3]

Казуал – это игры для массовой аудитории. У них простые правила и управление, геймерам не нужно тратить кучу времени и обладать какими-то особенными навыками для прохождения. Чаще всего их устанавливают на телефон, чтобы нескучно провести время по дороге на работу или отвлечься перед сном от навязчивых мыслей. Особой популярностью игры в казуальном стиле пользуются у детей и женщин. Вполне возможно, что это связано с простотой, сочной картинкой и мультяшной графикой. Один из ярких примеров – это «три в ряд».

Обычно казуальные игры имеют ряд особенностей:

* Легкий геймплей. Это не означает, что он скучный или примитивный. Как раз наоборот: простота игрового процесса затягивает на длительное время, потому что у игры простые правила, которые не требуют долгих разбирательств. Отличным примером служит Subway Surfers.
* Несложное управление. Не нужно нажимать заковыристые комбинации на клавиатуре или как-то изворачиваться в телефоне. Достаточно сделать пару кликов мышкой или ткнуть в экран смартфона, чтобы совершить действие.
* Короткие игровые сессии. Это не те игрушки, которые требуют от пользователя многочасового залипания. Казуальные игры как раз рассчитаны на то, чтобы человек не находился в них дольше получаса. Они не успевают надоесть, поэтому заставляют игрока возвращаться к ним снова и снова. [4]

Как можно заметить, данные игровые жанры отлично сочетаются и дополняют друг друга, что и привело к появлению нового популярного жанра, который станет основой для разработки данного дипломного проекта.

## Сравнительный анализ аналогов и прототипов

Прежде, чем приступить к реализации, следует проанализировать существующие на данный момент программные продукты подобного характера и схожей тематикой. Данный анализ поможет сделать верные выводы о том, как необходимо создавать собственный программный продукт на основе достоинств и недостатков рассмотренных аналогов

Так как жанр разрабатываемого игрового приложения совмещает в себе принципы казуальных и гиперказуальных игр, в качестве аналогов будут рассмотрены представители обоих направлений.

Первым рассматриваемым аналогом будет являться представитель гиперказуального жанра – «Comics Bob».

На каждом из уровней Comics Bob, пользователю нужно принимать решения от имени главных героев, отдавая предпочтение одному из двух возможных вариантов. Внимательно проанализировав окружающую обстановку, игроку нужно выбрать первый или второй вариант, чтобы вызвать действие с позитивными последствиями. В случае ошибочного выбора, потребуется повторить попытку и выбрать правильный вариант решения. [5] Скриншот главного экрана игры представлен на рисунке 1.1.



Рисунок 1.1 – Главный экран «Comics Bob»

В данной игре можно выделить определенные достоинства и недостатки.

Для начала опишем положительные стороны данного приложения:

* наличие захватывающего сюжета;
* простота интерфейса;
* красочная и при этом простая графика.

Благодаря описанным выше качествам, проект «Comics Bob» великолепно решает вопросы, связанные с рекламированием и распространением проекта в массы.

Несмотря на позитивные стороны данной игры, у нее присутствуют недостатки:

* отсутствие более глубокого геймплея;
* отсутствие возможности внедрения комплексных игровых балансов;
* отсутствие возможности проведения внутриигровых временных событий.

В качестве следующего аналога можно рассмотреть игру «Love & Pies», главный экран которой представлен на рисунке 1.2.



Рисунок 1.2 – Главный экран «Love & Pies»

Love & Pies – это увлекательная казуальная игра, сочетающая в себе два очень популярных жанра в мобильном игровом секторе: игры по восстановлению и украшению, а также головоломки со слиянием. Геймплей в Love & Pies следующий: чтобы продвинуться в реконструкции кофейни, игроку потребуются деньги, которые он может получить, готовя заказы в кафе. [6]

Рассмотрим достоинства данной игры:

* увлекательный сюжет;
* простой, но глубокий геймплей, который позволяет добавлять временные игровые события;
* красочная графика;
* большое количество игрового контента.

Также следует отметить и недостатки:

* рекламирование проектов с подобной графикой и геймплеем требуют больших затрат.

Последним рассматриваемым аналогом является игровой проект «Makeover Merge», представленный на рисунке 1.3.



Рисунок 1.3 – Игра «Makeover Merge»

Makeover Merge — это 2D-игра-головоломка, в которой пользователю предстоит сочетать головоломки с задачами по преображению персонажа и дома. Цель состоит в том, чтобы собрать на доске необходимые элементы для выполнения задач, объединяя одни и те же объекты по два. [7]

Выделим следующие сильные стороны данного приложения:

* простой, но глубокий геймплей, который позволяет добавлять временные игровые события;
* большое количество игрового контента;
* возможность изменения образов персонажей, которая может привлечь женскую аудиторию.

Также выделим слабые стороны данного приложения:

* графика, не соответствующая современным стандартам;
* слабая техническая реализация.

Благодаря рассмотренным аналогам можно сделать вывод, что игровое приложение, проектируемое в рамках дипломного проекта, должно соответствовать следующим критериям:

* наличие крайне простого и всем понятного геймплея;
* разнообразный игровой процесс;
* использование современной красочной графики;
* наличие анимированных сюжетных сцен, задача которых заключается в привлечении игроков.

## Постановка задачи проектирования

В рамках дипломного проектирования осуществляется разработка мобильной гибридно-казуальной игры «Jess` Stories». Функциональное назначение разрабатываемого программного средства состоит в развлечении пользователей.

Для того чтобы в результате выполнения дипломного проекта получилось законченное, стабильно работающее и конкурентноспособное приложение, необходимо решить определенный набор задач на этапе проектирования.

В качестве первостепенных задач можно выделить:

* выбор языка программирования;
* выбор среды разработки;
* выбор целевой платформы;
* детальная проработка функционала игровых возможностей будущего приложения;
* разработка технической документации.

Выбор языка программирования является одним из первых и ключевых шагов в разработке любых проектов. От того, насколько правильно определиться с технологией, зависит производительность проекта, его масштабируемость и успешность на рынке. [8] Выбор языка программирования для разработки данного дипломного проекта будет происходить среди современных высокоуровневых языков, обычно используемых в сфере разработки игр.

Также немаловажную роль играет правильный выбор среды разработки. В случае создания игр, обычно в качестве среды разработки выступают специальные платформы – так называемые «игровые движки». От правильного выбора движка будет зависеть удобство и скорость разработки проекта, а также наличие или отсутствие необходимости реализации отдельного функционала для разных целевых платформ.

Как было замечено ранее, для одного и того же программного продукта могут существовать различные целевые платформы. На разных целевых платформах пользователи играют по-разному: у них могут быть разные устройства ввода, различные паттерны поведения и продолжительность сессий, что ставит перед разработчиком задачу по их выбору.

В предыдущей главе были рассмотрены аналоги разрабатываемого дипломного проекта, были выделены их слабые и сильные стороны. На основе этих данных появляется возможность детальной проработки функциональной составляющей данного проекта. Это необходимо для того, чтобы на этапе проектирования не упустить важных моментов и не допускать ошибок в разработке, которые могут повлиять на стабильность, работоспособность и привлекательность для пользователей будущего приложения.

Техническую документацию необходимо разработать для того, чтобы разработанное приложение отвечало всем требованиям, которые выдвигались на этапе проектирования.

## Анализ требования к проекту

При разработке мобильного приложения необходимо решить следующие задачи:

* определение логико-функциональной структуры приложения;
* проведение анализа и оптимизации основных программных модулей;
* разработка современного и интуитивно понятного интерфейса;
* интеграция графики, анимации, звуков;
* реализация игровых механик;
* интеграция аналитики;
* интеграция внутриигровых покупок;
* интеграция рекламных сервисов;
* дистрибьюция проекта на разные целевые платформы.

Интерфейс данного приложения должен быть не только интуитивно понятным для будущего пользователя, но также должен выполнять все возложенные на него функции.

Приложение должно иметь равный функционал и возможности для всех выбранных целевых платформ. По мере разработки должно проводится комплексное тестирование разрабатываемого продукта.

## Выбор и обоснование средств и методов решения задач

Выбор языка программирования для разработки данного дипломного проекта происходил среди современных высокоуровневых языков, обычно используемых в сфере разработки игр. Главными кандидатами были следующие языки программирования:

* C#;
* C++;
* JavaScript.

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей. [9]

Еще одним существенным преимуществом C# является его мощная библиотека классов, включающая множество предварительно написанных фрагментов кода и функций, которые можно использовать для быстрого создания сложных игровых систем. В C# также реализовано автоматическое управление памятью, что означает, что разработчикам не нужно беспокоиться о выделении и освобождении памяти вручную, поскольку сборщик мусора языка обрабатывает это автоматически.

Однако, пожалуй, наиболее значительным преимуществом C# для разработки игр является его кроссплатформенная совместимость. Код C# можно скомпилировать для работы на нескольких платформах, включая Windows, macOS, Linux, Android, iOS и даже веб-браузеры. Такая кроссплатформенная совместимость облегчает разработчикам создание игр для различных устройств и операционных систем, что может увеличить потенциальную аудиторию игры.

C# можно использовать для различных аспектов разработки игр, включая игровые движки, инструменты и сценарии. Многие популярные игровые движки, такие как Unity и Godot, используют C# в качестве основного языка сценариев. Например, в Unity C# используется для создания игровой логики, пользовательских интерфейсов и других игровых систем.

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является одним из самых популярных и распространенных языков.

С++ является компилируемым языком, а это значит, что компилятор транслирует исходный код на С++ в исполняемый файл, который содержит набор машинных инструкций. Но разные платформы имеют свои особенности, поэтому скомпилированные программы нельзя просто перенести с одной платформы на другую и там уже запустить. Однако на уровне исходного кода программы на С++ по большей степени обладают переносимостью, если не используются какие-то специфичные для текущей ос функции. А наличие компиляторов, библиотек и инструментов разработки почти под все распространенные платформы позволяет компилировать один и тот же исходный код на С++ в приложения под эти платформы. [10]

C++ выбирается разработчиками игр благодаря своей высокой производительности и возможности полного контроля над ресурсами компьютера. Игровые приложения требуют максимальной эффективности для обеспечения плавного геймплея и реалистичной графики. Возможность напрямую взаимодействовать с аппаратурой компьютера делает C++ идеальным выбором для разработчиков, стремящихся к оптимизации своих игр. [11]

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. JavaScript – это мощный инструмент для создания игр, который используется многими разработчиками в всем мире. JavaScript позволяет создавать насыщенное и интерактивное игровое содержимое, которое может быть запущено на большинстве платформ, включая настольные компьютеры, мобильные устройства и веб-браузеры.

Для создания игр на JavaScript можно использовать готовые фреймворки и библиотеки, которые упрощают и ускоряют процесс разработки. Некоторые из популярных фреймворков и библиотек, которые используются для разработки игр на JavaScript, включают Phaser, Pixi.js, Three.js, Babylon.js и другие. [12]

Как уже было упомянуто ранее, для разработки игр часто используются игровые движки. Рассмотрим самые популярные из них:

* Unity;
* Unreal Engine;
* Phaser.

Unity — это кроссплатформенный игровой движок, созданный компанией Unity Technologies в 2005 году. С его помощью делают одиночные и многопользовательские игры с современной 2D- и 3D-графикой для таких платформ, как ПК, PlayStation, Xbox, Nintendo Switch.

Также на Unity создают игры для браузеров и устройств, работающих на Android и iOS. С помощью Unity можно делать игры с использованием технологий AR и VR. Технологии Unity используются в кино для создания компьютерной анимации, а также в промышленности для разработки иммерсивных 3D-приложений для AR и VR.

В Unity очень легко внедрить инструменты, которые отвечают за монетизацию игр, выпущенных на мобильных платформах, — такие, как показ рекламы и внутриигровые платежи. [13]

Unreal Engine — это движок для создания игр, один из двух наиболее популярных в мире. Unreal Engine традиционно лучше всего подходит для создания трехмерных мощных игр для компьютеров и консолей. Он изначально создавался для этой цели, возможность делать игры в 2D и под другие устройства в нем также имеется. На Unreal Engine написаны многие известные игры AAA-класса, то есть высокобюджетные, с хорошей графикой и рассчитанные на широкую аудиторию, — игры-блокбастеры. Впрочем, UE подходит и для создания низкобюджетных инди-игр, особенно если разработчикам важны графика и оптимизация. [14]

Phaser — это фреймворк для 2D-игр, используемый для создания игр HTML5 для компьютеров и мобильных устройств. Это бесплатное программное обеспечение, разработанное Photon Storm.

Phaser использует внутренние средства визуализации Canvas и WebGL и может автоматически переключаться между ними в зависимости от поддержки браузера. Это обеспечивает быстрый рендеринг на компьютерах и мобильных устройствах. Он использует библиотеку Pixi.js для рендеринга.

Игры можно развернуть на iOS, Android и в нативных приложениях для компьютера с помощью сторонних инструментов, таких как Apache Cordova и phonegap.

Игры, созданные с помощью Phaser, разрабатываются либо на JavaScript, либо на TypeScript. [15]

С учётом выбранного игрового жанра разрабатываемого в рамках дипломного проекта приложения было принято решение, что наиболее подходящим языком реализации программного продукта является C#, игровой движок – Unity, а целевые платформы – Android и iOS.

## Разработка технического задания

На основании анализа сущности проблемы и исходных данных были прописаны следующие требования к функциональности разрабатываемого игрового приложения:

* обеспечить разделение игрового функционала на структурные блоки;
* обеспечить высокую скорость работы игры на устройствах пользователей;
* разработать инструменты для интеграции в проект графики и анимации;
* обеспечить стабильную работу интегрированных сервисов;
* учесть особенности жанра гибридно-казуальных игр.

Расширенное техническое задание для проектируемого приложения представлен в приложении А.

# Проектирование ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Функциональная структура приложения

Приложение предназначено для развлечения пользователя. Для этого приложение должно обладать удобным и интуитивно понятным интерфейсом и соответствовать всем критериям гибридно-казуальных игр.

Исходя из приведенных характеристик и требований технического задания (приложение А), была разработана функциональная структура данного игрового приложения, которая представлена на рисунке 3.1.

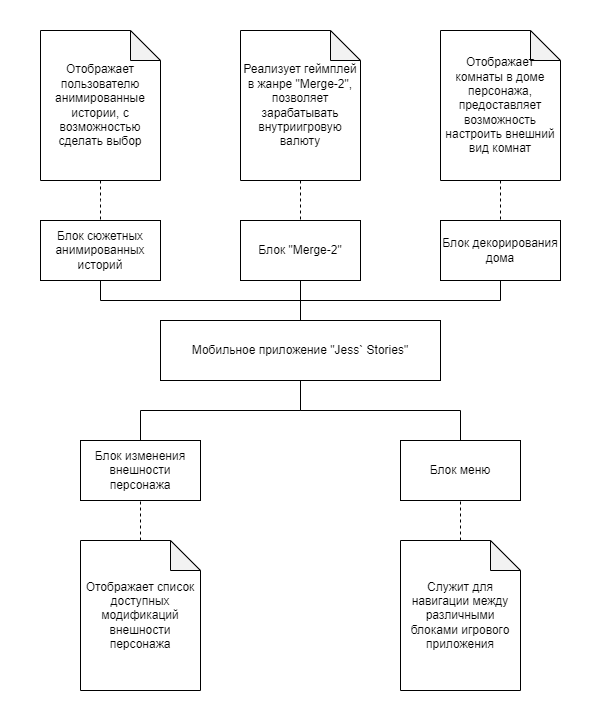


Рисунок 3.1 – Функциональная структура приложения

Приложение состоит из пяти основных обширных блоков:

* Блок сюжетных анимированных историй. В данном блоке пользователю будут отображаться красочные анимированные сюжетные истории. У игрока будет возможность сделать выбор развития историй.
* Блок «Merge-2». Реализует геймплей в жанре «Merge-2»: игроку необходимо объединять одинаковые предметы для того, чтобы выполнять заказы и зарабатывать внутриигровую валюту.
* Блок декорирования дома. В данном блоке пользователю будут предоставлены комнаты, в которых сначала необходимо убраться, а затем у него будет возможность расставить предметы на выбор.
* Блок изменения внешности персонажа. Будет отображать список всех доступных модификаций внешности персонажа, разбитый на категории.
* Блок меню. Обеспечивает навигацию внутри всего игрового приложения, также в данном блоке будут реализованы игровые настройки.

## Проектирование диаграммы вариантов использования

Диаграммы вариантов использования описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе.

Важно понимать, что диаграммы вариантов использования не предназначены для отображения проекта и не могут описывать внутреннее устройство системы. Диаграммы вариантов использования предназначены для упрощения взаимодействия с будущими пользователями системы, с клиентами, и особенно пригодятся для определения необходимых характеристик системы. Другими словами, диаграммы вариантов использования говорят о том, что система должна делать, не указывая сами применяемые методы.

Вариант использования описывает, с точки зрения действующего лица, группу действий в системе, которые приводят к конкретному результату.

Варианты использования являются описаниями типичных взаимодействий между пользователями системы и самой системой. Они отображают внешний интерфейс системы и указывают форму того, что система должна сделать (именно что, а не как).

При работе с вариантами использования важно помнить несколько простых правил:

* каждый вариант использования относится как минимум к одному действующему лицу;
* каждый вариант использования имеет инициатора;
* каждый вариант использования приводит к соответствующему результату (результату с «бизнес-значением»).

Варианты использования также могут взаимодействовать с другими вариантами использования. Три наиболее часто встречающихся типа взаимодействия между вариантами использования приведены ниже.

* Включение указывает, что вариант использования встраивается в другой вариант использования.
* Добавление указывает, что в определённых ситуациях или в некоторой точке (называемой точкой расширения) вариант использования будет расширен другим.
* Обобщение указывает, что вариант использования наследует характеристики «родительского» варианта использования и может переопределить некоторые из них или добавить новые, подобно наследованию в классах.

Действующее лицо является внешним источником (не элементом системы), который взаимодействует с системой через вариант использования. Действующие лица могут быть как реальными людьми (например, пользователями системы), так и другими компьютерными системами или внешними событиями.

Действующие лица представляют не физических людей или системы, а их роли. Это означает, что, когда человек взаимодействует с системой различными способами (предполагая различные роли), он отображается несколькими действующими лицами. [16]

Для игрового приложения «Jess` Stories» можно выделить одного актера – игрока. Игрок управляет развитием событий в сюжетных сценках, убирает и декорирует комнаты, меняет внешний вид персонажа, зарабатывает внутриигровую валюту в режиме «Merge-2», а также имеет возможность управлять настройками игры. Подробная диаграмма вариантов использования представлена в приложении Б.

## Информационное обеспечение проекта

В состав информационного обеспечения проекта на различных этапах разработки игрового приложения входят семейство стандартов UML, .NET Core C#, игровой движок Unity.

Программный код будет разрабатываться в IDE JetBrains Rider, разработанной специально для .NET и, в частности, для удобной работы с Unity.

Исходный код проекта будет сохраняться с помощью системы контроля версий git.

Для анимированных сценок будет использоваться библиотека Spine, которая разработана для воспроизведения анимации, созданной в программе Spine.

Для сериализации данных с целью последующего сохранения прогресса игрока будет использоваться библиотека Protobuf.Net.

## Разработка структуры конфигурации

Главной целью проектирования игры является выбор оптимальных паттернов программирования, использование которых сделает разработку приложения удобным и легким для расширения.

Ниже предоставлен список наиболее часто используемых паттернов в сфере разработки игровых приложений:

* одиночка;
* пул объектов;
* адаптер;
* наблюдатель;
* стратегия.

Рассмотрим данные паттерны детальнее.

Одиночка – это порождающий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.

Все реализации одиночки сводятся к тому, чтобы скрыть конструктор по умолчанию и создать публичный статический метод, который и будет контролировать жизненный цикл объекта-одиночки.

Если у разработчика есть доступ к классу одиночки, значит, будет доступ и к этому статическому методу. Из какой точки кода его бы ни вызвали, он всегда будет отдавать один и тот же объект. [17]

Применение паттерна «пул объектов» может значительно повысить производительность системы; его использование наиболее эффективно в ситуациях, когда создание экземпляров некоторого класса требует больших затрат, объекты в системе создаются часто, но число создаваемых объектов в единицу времени ограничено.

Процессы запрашивают объекты из пула объектов. Когда эти объекты больше не нужны, они возвращаются в пул для дальнейшего повторного использования.

Если при очередном запросе все объекты пула заняты, то процесс будет ожидать освобождения объекта. Для исключения подобной ситуации пул объектов должен уметь создавать новые объекты по мере необходимости. При этом он также должен реализовывать механизм периодической очистки неиспользуемых объектов. [18]

Адаптер – это структурный паттерн проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе.

Это объект-переводчик, который трансформирует интерфейс или данные одного объекта в такой вид, чтобы он стал понятен другому объекту. При этом адаптер оборачивает один из объектов, так что другой объект даже не знает о наличии первого. Например, можно обернуть объект, работающий в метрах, адаптером, который бы конвертировал данные в футы. Адаптеры могут не только переводить данные из одного формата в другой, но и помогать объектам с разными интерфейсами работать сообща. [19]

Наблюдатель – это поведенческий паттерн проектирования, который создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах.

Паттерн «наблюдатель» предлагает хранить внутри объекта издателя список ссылок на объекты подписчиков, причём издатель не должен вести список подписки самостоятельно. Он предоставит методы, с помощью которых подписчики могли бы добавлять или убирать себя из списка. Когда в издателе будет происходить важное событие, он будет проходиться по списку подписчиков и оповещать их об этом, вызывая определённый метод объектов-подписчиков. Издателю безразлично, какой класс будет иметь тот или иной подписчик, так как все они должны следовать общему интерфейсу и иметь единый метод оповещения. [20]

Стратегия – это поведенческий паттерн проектирования, который определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс, после чего алгоритмы можно взаимозаменять прямо во время исполнения программы.

Паттерн «стратегия» предлагает определить семейство схожих алгоритмов, которые часто изменяются или расширяются, и вынести их в собственные классы, называемые стратегиями. Вместо того, чтобы изначальный класс сам выполнял тот или иной алгоритм, он будет играть роль контекста, ссылаясь на одну из стратегий и делегируя ей выполнение работы. Чтобы сменить алгоритм, разработчику будет достаточно подставить в контекст другой объект-стратегию. Важно, чтобы все стратегии имели общий интерфейс. Используя этот интерфейс, контекст будет независимым от конкретных классов стратегий. С другой стороны, разработчик сможет изменять и добавлять новые виды алгоритмов, не трогая код контекста. [21]

Исходя из анализа паттернов было принято решение, что все вышеперечисленные паттерны будут полезны при реализации различных блоков разрабатываемого в рамках данного дипломного проекта игрового приложения.

## Разработка концепции пользовательского интерфейса

При разработке гибридно-казуальной игры концепция пользовательского интерфейса имеет важную роль. Пользовательский интерфейс должен быть красочным, чтобы привлекать внимание игрока, но при этом с первого взгляда пользователю должно быть понятно, что ему необходимо сделать на текущем экране.

Согласно техническому заданию, программное обеспечение разрабатывается под мобильные платформы Android и iOS.

Мобильные средства использующие эти платформы имеют различный размер экрана, вырезы и прочие элементы, которые меняются от устройства к устройству. При разработке пользовательского интерфейса необходимо предусмотреть наиболее удачное расположение элементов, чтобы они были удобны в использовании на любом устройстве с любым размером экрана.

Проанализировав аналоги, представленные в главе 1.2, с учетом выделенных функциональных блоков в главе 2.1, а также опираясь на разработанную диаграмму вариантов использования (приложение Б) было принято решение создать по одному отдельному экрану интерфейса для каждого функционального блока. Предполагаемый вид каждого экрана будет описан ниже.

Для блока сюжетных анимированных историй необходимо разработать минималистичный экран, на котором будут отображаться только кнопки для выбора развития сюжета.

В блоке декорирования комнаты необходимо разработать элементы, при нажатии на которые будет происходить уборка мусора, а также элементы для выбора декора.

В блоке изменения внешности персонажа необходимо разработать список, отображающий элементы выбора модификаторов внешности, а также элементы для переключения отображаемых категорий.

В блоке «Merge-2» необходимо реализовать игровое поле размерностью 7х9 ячеек, в котором посредством перетаскивания у пользователя будет возможность объединять предметы. Также на этом экране необходимо отобразить список заданий, доступных для выполнения игроку.

Для блока меню необходимо разработать экран, на котором будут расположены элементы, с помощью которых у игрока будет доступ ко всем функциям в игре.

Выход на экран меню будет доступен со всех экранов, разработанных для других блоков.

В основу интерфейса легли ресурсы, разработанные командами UI/UX дизайнеров и 2D художников.

## Безопасность и защита данных

Разработка игр представляет особые требования по безопасности. Необходимо обеспечить максимальную безопасность игр от несанкционированного доступа к данным, взломов, поддельных/небезопасных протоколов и так далее.

Защита игр от несанкционированного доступа - важный аспект защиты от нарушений и неправильного использования любого интеллектуального права. Для обеспечения данного рода защиты необходимо будет разработать инструменты для шифрования пользовательских данных, хранящихся в оперативной памяти, а также на накопителе постоянного хранения.

Также, при разработке приложения будут проведены различные тестирования на отказоустойчивость и безопасность данных. При возникновении ошибок в установленных при тестировании критических местах предусмотрены обработчики ошибок с их описанием. Это сделано для обратной связи с пользователем, чтобы он понимал, что произошло и почему так делать не стоит.

Помимо тестирования, проводимого самим разработчиком в рамках данного дипломного проекта, разработанное игровое приложение будет тестироваться профессиональной группой тестировщиков.

# РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ

## Результаты реализации серверной части

Приложение было разработано с помощью платформы разработки игр Unity, с использованием языка программирования C#. Для каждого функционального блока были разработаны различные модули, которые будут описаны ниже.

Для блока интерактивных сюжетных сценок был разработан класс StageController, в котором реализован паттерн «стратегия» и который необходим для воспроизведения данных сценок. Данный класс представлен в листинге 3.1.

Листинг 3.1– Класс StageController

using System;

using System.Collections.Generic;

using Cysharp.Threading.Tasks;

namespace Makeup.Story

{

    public class StageController

    {

        private int \_currentStageIndex;

        private List<IStage> \_stages;

        public IStage CurrentStage => \_stages[\_currentStageIndex];

        public bool IsLastStage => \_currentStageIndex >= \_stages.Count - 1;

        public bool IsFirstStage => \_currentStageIndex == 0;

        public int CurrentStageIndex => \_currentStageIndex;

        public event Action Complete;

        public event Action Fail;

        public void Init(List<IStage> stages)

        {

            \_stages = stages;

            \_currentStageIndex = -1;

            foreach (var choicesStage in \_stages)

                choicesStage.Init();

        }

        public void NextStage()

        {

            \_currentStageIndex++;

            if (\_currentStageIndex == \_stages.Count)

            {

                AllComplete();

Продолжение листинга 3.1

                return;

            }

            CurrentStage.Complete += OnStageComplete;

            CurrentStage.Fail += OnStageFail;

            CurrentStage.Start();

        }

        public async UniTask Restart(bool save)

        {

            CurrentStage.Complete += OnStageComplete;

            CurrentStage.Fail += OnStageFail;

            await CurrentStage.Restart(save);

        }

        public void PreRestartAll()

        {

            for (; \_currentStageIndex > 0; \_currentStageIndex--)

            {

                if (CurrentStage is CheckPointStage)

                {

                    \_currentStageIndex++;

                    CurrentStage.PreRestart(false);

                    break;

                }

                CurrentStage.Rollback();

            }

            if (\_currentStageIndex == 0)

                CurrentStage.PreRestart(false);

        }

        private void OnStageComplete()

        {

            Unsubscribe();

            NextStage();

        }

        private void Unsubscribe()

        {

            CurrentStage.Complete -= OnStageComplete;

            CurrentStage.Fail -= OnStageFail;

        }

        private void OnStageFail()

        {

            Unsubscribe();

Продолжение листинга 3.1

            Fail?.Invoke();

        }

        private void AllComplete()

        {

            Complete?.Invoke();

        }

        public void UnloadAll(bool byUser = false)

        {

            if (\_stages == null)

                return;

            if (byUser)

                CurrentStage.Exit();

            foreach (var choicesStage in \_stages)

                choicesStage.Unload();

        }

    }

}

Также для этого блока был разработан класс BaseStage, представленный в листинге 3.2, который является базовой реализацией паттерна «стратегия» для класса StageController.

Листинг 3.2 – Класс BaseStage

using System;

using Cysharp.Threading.Tasks;

using UnityEngine;

namespace Makeup.Story

{

public abstract class BaseStage<TEpisode, TData> : IStage

where TEpisode : EpisodeConfig

{

public BaseStage(TEpisode episode, TData data)

{

Data = data;

Episode = episode;

}

protected TData Data { get; }

protected TEpisode Episode { get; }

public event Action Complete;

public event Action Fail;

public virtual bool IsInProgressBar => false;

public virtual void Init()

{

Продолжение листинга 3.2

#if !PRODUCTION\_BUILD

Debug.Log($"Init {GetType()}");

#endif

}

public virtual void Start()

{

#if !PRODUCTION\_BUILD

Debug.Log($"Start {GetType()}");

#endif

}

public virtual void Release()

{

#if !PRODUCTION\_BUILD

Debug.Log($"ReleaseAndHide {GetType()}");

#endif

}

public virtual void PreRestart(bool save)

{

#if !PRODUCTION\_BUILD

Debug.Log($"PreRestart {GetType()}");

#endif

}

public virtual async UniTask Restart(bool save)

{

#if !PRODUCTION\_BUILD

Debug.Log($"Restart {GetType()}");

#endif

await UniTask.CompletedTask;

}

public virtual void Unload()

{

#if !PRODUCTION\_BUILD

Debug.Log($"Unload {GetType()}");

#endif

}

public virtual void Rollback()

{

#if !PRODUCTION\_BUILD

Debug.Log($"Rollback {GetType()}");

#endif

}

public virtual void Exit()

Продолжение листинга 3.2

{

#if !PRODUCTION\_BUILD

Debug.Log($"Exit {GetType()}");

#endif

}

public virtual void OnComplete()

{

#if !PRODUCTION\_BUILD

Debug.Log($"Complete {GetType()}");

#endif

Complete?.Invoke();

}

public virtual void OnFailed()

{

#if !PRODUCTION\_BUILD

Debug.Log($"Fail {GetType()}");

#endif

Fail?.Invoke();

}

}

}

Также для данного блока был разработан класс BaseChoiseStage, который реализует непосредственно запуск анимированных сценок. Класс BaseChoiseStage представлен в листинге 3.3.

Листинг 3.3 – Класс BaseChoiseStage

using System;

using System.Linq;

using Cysharp.Threading.Tasks;

using FuryLion.UI;

using Makeup.Data;

using Makeup.Shop;

using Makeup.Story.Scene;

using Spine.Unity;

using UI.Elements.TopBar;

using UnityEngine;

namespace Makeup.Story

{

public class BaseChoiceStage<TEpisode, TScene, TData> : BaseStage<TEpisode, TData>

where TEpisode : EpisodeConfig

where TScene : BaseEpisodeScene

where TData : ChoiceStageViewModel

{

private readonly EpisodeChoiceAnalyticsHelper \_analytics;

Продолжение листинга 3.3

protected readonly int Index;

private ChoiceStageItemViewModel[] \_items;

protected BaseChoiceStage(TEpisode episode, TData data, int index) : base(episode, data)

{

Index = index;

\_analytics = new EpisodeChoiceAnalyticsHelper(episode, data, index);

\_items = Data.Items.ToArray();

for (var i = 0; i < \_items.Length; i++)

\_items[i].SetAds(i == 2, RewardAdsPlacement.epic\_choise\_click, i);

\_items[0].Select += OnSelect1;

\_items[1].Select += OnSelect2;

if (\_items.Length == 3)

\_items[2].Select += OnSelect3;

}

private ChoicesPage ChoicesPage { get; set; }

protected TScene Scene => Episode.SceneLoader.Scene as TScene;

public override bool IsInProgressBar => true;

protected virtual float HidePanelDelay => 1f;

protected virtual async UniTask LoadSceneAndOpenPage()

{

await Episode.SceneLoader.Load();

if (!BasePage.IsOpened<ChoicesPage>())

await Page.OpenAsync<ChoicesPage>(new ChoicesPage.ChoicesPageParam());

ChoicesPage = BasePage.GetPage<ChoicesPage>();

Fade.Hide();

}

protected virtual async void OnSelect1()

{

await OnSelect(0);

}

protected virtual async void OnSelect2()

{

await OnSelect(1);

Продолжение листинга 3.3

}

protected virtual async void OnSelect3()

{

await OnSelect(2);

}

protected virtual async UniTask OnSelect(int index)

{

var story = StoryManager.Story;

if (HubManager.IsEpisodesReplayEnabled && story.IsFinishedOnce && !Episode.IsLast())

{

var isRewardCollected = NarrativeDataManager.IsChoiceRewardCollected(story, story.StageController.CurrentStageIndex, index);

if (!isRewardCollected)

{

const int starsCount = 1;

PlayerDataManager.Adjust(starsCount, Currency.Stars, GamePlacement.EpisodeChoiceReplayReward, CurrencyBarUpdateType.None);

NarrativeDataManager.OnChoiceRewardCollected(story, story.StageController.CurrentStageIndex, index);

StoryManager.OnStarCollected();

var choicesPanelItem = BasePage.GetPage<ChoicesPage>().ChoicesPanel.Items.First(item => item.Data.Index == index);

var starIconTransform = choicesPanelItem.StarIconTransform;

var starsBar = MainTopBar.StarsBar;

using var \_ = new BlockerScope(LayerHelper.MessageBox);

starsBar.DoShow();

choicesPanelItem.OnStarCollected();

await starsBar.ApplyWithEffects(starIconTransform.position, LayerHelper.MessageBox, starsCount);

await starsBar.DoHide();

}

}

Продолжение листинга 3.3

\_analytics.OnChoiceStart(index);

Scene.SetTimeScaleAllowed(true);

EpisodeAnalytics.ChoiceSelected(Episode, \_analytics.StageIndex, index, Data.Items[index]);

PreAnimation(index);

using (var \_ = new BlockerScope(LayerHelper.MessageBox))

{

await Animation(index, () => MidAnimation(index));

}

\_analytics.OnChoiceComplete(index);

var viewModel = Data.Items[index];

if (ConstantsConfig.IsEpisodesAdsRewardsFreeReuseEnabled && viewModel.IsAds)

viewModel.SetAds(false, RewardAdsPlacement.epic\_choise\_click, index);

PostAnimation(index);

}

protected virtual void PreAnimation(int index)

{

ChoicesPage.LockInputAnimated();

}

protected virtual async UniTask Animation(int index, Action reaction)

{

await UniTask.Delay(1000);

reaction?.Invoke();

await UniTask.Delay(1000);

}

protected virtual void MidAnimation(int index)

{

var data = Data.Items[index];

var iconIndex = \_items.IndexOf(data);

ChoicesPage.ShowMark(iconIndex);

if (data.IsCorrect)

ChoicesPage.MakeProgressGreen();

else

ChoicesPage.MakeProgressRed();

Продолжение листинга 3.3

if (HidePanelDelay > 0)

ChoicesPage.DoHideChoices(HidePanelDelay);

}

protected virtual void PostAnimation(int index)

{

if (Data.Items[index].IsCorrect)

{

OnComplete();

}

else

{

if (GameConfig.GameSettings.RemoveWrongChoices)

{

var itemToRemove = Data.Items[index];

\_items = \_items.Where(t => t != itemToRemove).ToArray();

}

OnFailed();

}

}

public sealed override async void Start()

{

base.Start();

EpisodeAnalytics.ChoiceStarted(Episode, \_analytics.StageIndex);

using var \_ = new BlockerScope(LayerHelper.MessageBox);

await LoadSceneAndOpenPage();

\_analytics.OnTransitionToStage();

Scene.SetTimeScaleAllowed(true);

await IntroAnimation();

\_analytics.OnTransitionCompleted();

Scene.SetTimeScaleAllowed(false);

await ChoicesPage.ShowChoicesAndBar(\_items);

\_analytics.OnChoiceShowed();

}

public override void OnComplete()

{

EpisodeAnalytics.ChoiceCompleted();

base.OnComplete();

}

protected virtual UniTask IntroAnimation()

{

Продолжение листинга 3.3

return UniTask.CompletedTask;

}

public override void PreRestart(bool save)

{

base.PreRestart(save);

ChoicesPage.ResetCurrentStage();

}

public override async UniTask Restart(bool save)

{

using var \_ = new BlockerScope(LayerHelper.MessageBox);

EpisodeAnalytics.ChoiceStarted(Episode, \_analytics.StageIndex);

await ChoicesPage.ShowChoices(\_items);

await base.Restart(save);

}

public override async void OnFailed()

{

EpisodeAnalytics.ChoiceFailed();

Scene.SetTimeScaleAllowed(false);

using var \_ = new BlockerScope(LayerHelper.MessageBox);

await UniTask.Delay(TimeSpan.FromSeconds(1));

base.OnFailed();

}

public override void Rollback()

{

base.Rollback();

ChoicesPage.ResetCurrentStage();

}

public override void Exit()

{

base.Exit();

EpisodeAnalytics.ChoiceFailed();

}

public override void Unload()

{

\_analytics.OnUnload();

base.Unload();

}

Продолжение листинга 3.3

}

public class BaseChoiceStage<TEpisode, TScene> : BaseChoiceStage<TEpisode, TScene, ChoiceStageViewModel>

where TEpisode : EpisodeConfig

where TScene : BaseEpisodeScene

{

protected BaseChoiceStage(TEpisode episode, ChoiceStageViewModel data, int index) : base(episode, data, index)

{

}

}

public class ChoiceStageViewModel

{

public ChoiceStageItemViewModel[] Items;

public static implicit operator ChoiceStageViewModel(ChoicesStageData data)

{

var items = new ChoiceStageItemViewModel[data.Choices.Length];

for (var i = 0; i < items.Length; i++)

items[i] = data.Choices[i];

return new ChoiceStageViewModel

{

Items = items

};

}

}

public class ChoiceStageItemViewModel

{

public string Animation;

public bool IsAds;

public RewardAdsPlacement AdPlacement;

public bool IsCorrect;

public SkeletonDataAsset SpineAsset;

public Sprite Sprite;

public int Index;

public event Action Select;

public void SelectData()

{

Select?.Invoke();

}

Продолжение листинга 3.3

public void SetAds(bool isAds, RewardAdsPlacement placement, int index)

{

IsAds = isAds;

AdPlacement = placement;

Index = index;

}

public static implicit operator ChoiceStageItemViewModel(ChoiceData data)

{

return new ChoiceStageItemViewModel

{

Sprite = data.Sprite,

IsCorrect = data.IsCorrect,

SpineAsset = data.SpineAsset,

Animation = data.Animation

};

}

}

}

Для блока декорирования комнат был разработан класс RoomView, представленный в листинге 3.4. Данный класс отвечает за отображение комнаты и элементов фурнитуры в ней.

Листинг 3.4 – Класс RoomView

using System;

using System.Collections.Generic;

using FuryLion;

using FuryLion.UI;

using Makeup.UI.Home;

using NaughtyAttributes;

using UnityEngine;

namespace Makeup.House

{

    public class RoomView : Element

    {

        [SerializeField] private Room \_room;

        [SerializeField] private Vector2 \_boundary;

        [SerializeField, Range(2, 3)] private int \_screenCount = 2;

        [SerializeField] private List<RoomObject> \_roomObjects;

        [SerializeField] private TaskPack[] \_taskPacks;

        private RoomConfig Config => RoomConfigs.Configs[\_room];

        [field: SerializeField] public float IconScaleModifier { private set; get; } = 1.0f;

Продолжение листинга 3.4

        public bool IsLowRes { private set; get; }

        public IEnumerable<BrokenRoomObject> BrokenRoomObjects => GetObjects<BrokenRoomObject>();

        public IEnumerable<TemporaryRoomObject> TemporaryRoomObjects => GetObjects<TemporaryRoomObject>();

        public IEnumerable<SelectableRoomObject> SelectableRoomObjects => GetObjects<SelectableRoomObject>();

        public int ScreenCount => \_screenCount;

        public EditTasks EditTasks { get; } = new();

        private IEnumerable<T> GetObjects<T>() where T : RoomObject

        {

            foreach (var roomObject in \_roomObjects)

            {

                if (roomObject is T target)

                    yield return target;

            }

        }

        public void Init(bool isLowRes)

        {

            IsLowRes = isLowRes;

            for (var i = 0; i < \_taskPacks.Length; i++)

            {

                var taskPack = \_taskPacks[i];

                taskPack.Init(this, \_room, i, \_taskPacks.Length);

                foreach (var task in taskPack.Tasks)

                {

                    if (task is BaseDecorationTask decorationTask)

                        EditTasks.Add(decorationTask);

                }

            }

        }

        public RoomTask GetRoomTask(string id)

        {

            foreach (var taskPack in \_taskPacks)

            {

                var task = taskPack.GetTask(id);

                if (task != null)

                    return task;

            }

Продолжение листинга 3.4

            return null;

        }

        public RoomObjectPin GetPin(string taskId)

        {

            foreach (var taskPack in \_taskPacks)

            {

                foreach (var task in taskPack.Tasks)

                {

                    if (string.Equals(task.Id, taskId))

                        return task.Pin;

                }

            }

            return null;

        }

        public TaskPack GetCurrentPack()

        {

            foreach (var taskPack in \_taskPacks)

            {

                if (!taskPack.IsCompleted)

                    return taskPack;

            }

            return null;

        }

        public Bounds GetWorkAreaBounds(int roomPart)

        {

            var positionX = transform.position.x;

            var startPos = positionX - RoomPartWidth \* 0.5f \* \_screenCount;

            var bounds = new Bounds();

            bounds.SetMinMax(new Vector2

            {

                x = startPos + RoomPartWidth \* roomPart,

                y = RoomPartHeight \* -0.5f + BottomIndentWorkArea

            }, new Vector2

            {

                x = startPos + RoomPartWidth \* roomPart + RoomPartWidth,

                y = RoomPartHeight \* 0.5f - TopIndentWorkArea

            });

            return bounds;

        }

        private float RoomPartWidth => 8.0f \* 2.0f;

Продолжение листинга 3.4

        private float RoomPartHeight => 17.4f \* 2.0f;

        private float BottomIndentWorkArea => 7.0f;

        private float TopIndentWorkArea => 6.0f;

        private Vector2 SizeWorkArea => new Vector2(RoomPartWidth, RoomPartHeight);

#if UNITY\_EDITOR

        private void OnDrawGizmos()

        {

            var position = transform.position;

            Gizmos.DrawWireCube(position, \_boundary);

            var cameraWidth = 8.0f \* 2;

            var cameraHeight = 17.4f \* 2;

            var cameraSize = new Vector2(cameraWidth, cameraHeight);

            var center = /\*\_boundary \* 0.5f +\*/ (Vector2)position;

            var left = new Vector2

            {

                x = center.x - cameraWidth \* 0.5f \* (\_screenCount - 1),

                y = center.y

            };

            var right = new Vector2

            {

                x = center.x + cameraWidth \* 0.5f \* (\_screenCount - 1),

                y = center.y

            };

            void DrawDottedRect(Vector2 size, Vector2 center)

            {

                using (new HandlesColor(Color.green))

                {

                    UnityEditor.Handles.DrawDottedLine(new Vector3(center.x - size.x \* 0.5f, center.y - size.y \* 0.5f), new Vector3(center.x - size.x \* 0.5f, center.y + size.y \* 0.5f), 6);

                    UnityEditor.Handles.DrawDottedLine(new Vector3(center.x - size.x \* 0.5f, center.y + size.y \* 0.5f), new Vector3(center.x + size.x \* 0.5f, center.y + size.y \* 0.5f), 6);

                    // top indent

                    UnityEditor.Handles.DrawDottedLine(new Vector3(center.x - size.x \* 0.5f, center.y + size.y \* 0.5f - 8.0f), new Vector3(center.x + size.x \* 0.5f, center.y + size.y \* 0.5f - 8.0f), 6);

Продолжение листинга 3.4

                    UnityEditor.Handles.DrawDottedLine(new Vector3(center.x + size.x \* 0.5f, center.y + size.y \* 0.5f), new Vector3(center.x + size.x \* 0.5f, center.y - size.y \* 0.5f), 6);

                    UnityEditor.Handles.DrawDottedLine(new Vector3(center.x + size.x \* 0.5f, center.y - size.y \* 0.5f), new Vector3(center.x - size.x \* 0.5f, center.y - size.y \* 0.5f), 6);

                    // bottom indent

                    UnityEditor.Handles.DrawDottedLine(new Vector3(center.x + size.x \* 0.5f, center.y - size.y \* 0.5f + 7.0f), new Vector3(center.x - size.x \* 0.5f, center.y - size.y \* 0.5f + 7.0f), 6);

                }

            }

            DrawDottedRect(cameraSize, left);

            DrawDottedRect(cameraSize, right);

            for (var i = 0 ; i < \_screenCount; i++)

                DrawRect(GetWorkAreaBounds(i), Color.magenta);

        }

        private void DrawRect(Bounds bounds, Color color)

        {

            DrawRect(bounds.center, bounds.size, color);

        }

        private void DrawRect(Vector2 center, Vector2 size, Color color)

        {

            using var \_ = new GizmosScopes.Color(color);

            Gizmos.DrawWireCube(center, size);

        }

        [Button]

        private void Refresh()

        {

            \_roomObjects.Clear();

            \_roomObjects.AddRange(transform.GetComponentsInChildren<RoomObject>(true));

        }

        [Button]

        private void SetDefault()

        {

            SetActiveBrokenRoomObjects(true);

            SetActiveTemporaryRoomObjects(false);

            SetActiveSkin(-1);

Продолжение листинга 3.4

        }

        [Button]

        private void SetClearAndTools()

        {

            SetActiveBrokenRoomObjects(false);

            SetActiveTemporaryRoomObjects(true);

            SetActiveSkin(-1);

        }

        [Button]

        private void SetClear()

        {

            SetActiveBrokenRoomObjects(false);

            SetActiveTemporaryRoomObjects(false);

            SetActiveSkin(-1);

        }

        [Button]

        private void SetSkin1()

        {

            SetActiveBrokenRoomObjects(false);

            SetActiveTemporaryRoomObjects(false);

            SetActiveSkin(0);

        }

        [Button]

        private void SetSkin2()

        {

            SetActiveBrokenRoomObjects(false);

            SetActiveTemporaryRoomObjects(false);

            SetActiveSkin(1);

        }

        [Button]

        private void SetSkin3()

        {

            SetActiveBrokenRoomObjects(false);

            SetActiveTemporaryRoomObjects(false);

            SetActiveSkin(2);

        }

        [Button]

        private void HideAllPins()

        {

            foreach (var roomObject in \_roomObjects)

            {

                var pin = roomObject.Pin;

                if (pin == null)

                    continue;

Продолжение листинга 3.4

                pin.gameObject.SetActive(false);

            }

        }

        [Button]

        private void ShowDecorationPins()

        {

            foreach (var taskPack in \_taskPacks)

            {

                foreach (var taskPackTask in taskPack.Tasks)

                {

                    if (taskPackTask is BaseDecorationTask)

                        taskPackTask.ShowPin\_Editor();

                }

            }

        }

        [Button]

        private void HideDecorationPins()

        {

            foreach (var taskPack in \_taskPacks)

            {

                foreach (var taskPackTask in taskPack.Tasks)

                {

                    if (taskPackTask is BaseDecorationTask)

                        taskPackTask.HidePin\_Editor();

                }

            }

        }

        [Button]

        private void FixPins()

        {

            var pins = GetComponentsInChildren<RoomObjectPin>(true);

            foreach (var pin in pins)

                pin.transform.parent = transform;

        }

        private void SetActiveBrokenRoomObjects(bool value)

        {

            // foreach (var roomBrokenRoomObject in BrokenRoomObjects)

                // roomBrokenRoomObject.SetActive(value, true);

        }

        private void SetActiveTemporaryRoomObjects(bool value)

        {

            // foreach (var temporaryRoomObject in TemporaryRoomObjects)

Продолжение листинга 3.4

                // temporaryRoomObject.SetActive(value, true);

        }

        private void SetActiveSkin(int index)

        {

            foreach (var roomSelectableRoomObject in SelectableRoomObjects)

                roomSelectableRoomObject.SetSkin(index);

        }

#endif

    }

}

Также для данного блока был разработан класс RoomObject, который используется для отображения элементов в комнате. Класс RoomObject представлен в листинге 3.5.

Листинг 3.5 – Класс RoomObject

using System;

using Cysharp.Threading.Tasks;

using DG.Tweening;

using FuryLion.UI;

using NaughtyAttributes;

using UnityEngine;

using UnityEngine.Serialization;

namespace Makeup.House

{

    using UI.Home;

    public abstract class RoomObject : MonoBehaviour

    {

        [FormerlySerializedAs("\_roomObjectPin")]

        [SerializeField] private RoomObjectPin \_pin;

        [field: SerializeField] public RoomObjectType Type { private set; get; }

        private Tween \_setActiveTween;

        private bool \_isHidden;

        private RoomObjectAnimator \_roomObjectAnimator;

        public RoomObjectPin Pin => \_pin;

        public virtual bool HasBase => false;

        public void SetActive(bool value)

        {

Продолжение листинга 3.5

            gameObject.SetActive(value);

        }

        public virtual void Install()

        {

            PlayAnimation(RoomObjectActionType.Install);

        }

        public virtual async UniTask InstallAsync()

        {

            await PlayAnimation(RoomObjectActionType.Install);

        }

        public void Change()

        {

            PlayAnimation(RoomObjectActionType.Change);

            DOFade(1.0f, 0.0f, 1.0f);

        }

        public async UniTask ChangeAsync()

        {

            await UniTask.WhenAll(PlayAnimation(RoomObjectActionType.Change), DOFade(1.0f, 0.0f, 1.0f).ToUniTask(TweenCancelBehaviour.Complete));

        }

        public abstract Tween DOFade(float value, float duration, float from);

        public virtual void Remove()

        {

        }

        public virtual UniTask RemoveAsync()

        {

            return UniTask.CompletedTask;

        }

        protected async UniTask PlayAnimation(RoomObjectActionType type)

        {

            if (\_roomObjectAnimator != null)

            {

                \_roomObjectAnimator.Cancel();

                Recycler.Release(\_roomObjectAnimator);

            }

            \_roomObjectAnimator = Recycler.Get<RoomObjectAnimator>();

Продолжение листинга 3.5

            try

            {

                await \_roomObjectAnimator.PlayAsync(this, type);

            }

            catch (OperationCanceledException)

            {

            }

            Recycler.Release(\_roomObjectAnimator);

            \_roomObjectAnimator = null;

        }

        public void InstallInBackground()

        {

            SetActive(true);

        }

        public void RemoveInBackground()

        {

            SetActive(false);

        }

#if UNITY\_EDITOR

        [Button]

        private void ShowPin() => ShowPin(true);

        [Button]

        private void HidePin() => ShowPin(false);

        private void ShowPin(bool value)

        {

            \_pin.gameObject.SetActive(value);

        }

        [Button]

        private void CreatePinWithPrice()

        {

            var localPosition = Vector2.zero;

            if (\_pin != null)

            {

                localPosition = \_pin.transform.localPosition;

                DestroyImmediate(\_pin.gameObject);

            }

            var pin = UnityEditor.AssetDatabase.LoadAssetAtPath<RoomObjectPin>("Assets/House/Prefabes/Elements/RoomTaskPin with price.prefab");

Продолжение листинга 3.5

            \_pin = UnityEditor.PrefabUtility.InstantiatePrefab(pin, transform) as RoomObjectPin;

            \_pin.SetLocalPositionXY(localPosition.x, localPosition.y);

        }

        [Button]

        private void CreatePin()

        {

            var localPosition = Vector2.zero;

            if (\_pin != null)

            {

                localPosition = \_pin.transform.localPosition;

                DestroyImmediate(\_pin.gameObject);

            }

            var pin = UnityEditor.AssetDatabase.LoadAssetAtPath<RoomObjectPin>("Assets/House/Prefabes/Elements/RoomTaskPin.prefab");

            \_pin = UnityEditor.PrefabUtility.InstantiatePrefab(pin, transform) as RoomObjectPin;

            \_pin.SetLocalPositionXY(localPosition.x, localPosition.y);

        }

#endif

    }

}

Для блока «Merge-2» был реализован класс TokenWrapper, который используется как контейнер для объектов на игровом поле. Данный класс представлен в листинге 3.6.

Листинг 3.6 – Класс TokenWrapper

using System;

using Cysharp.Threading.Tasks;

using DG.Tweening;

using FuryLion.ObjectPool;

using FuryLion.UI;

using FuryLion.UI.Extensions;

using FuryLion.UI.Vfx;

using Makeup;

using Merge.Config;

using Merge.Core.Grid;

using Merge.Core.Tasks;

using Merge.Core.UI;

using Merge.Data;

using Merge.Scripts.Core.Mechanics;

using UnityEngine;

Продолжение листинга 3.6

using DragElement = Merge.Core.Grid.DragElement;

using IReceivable = FuryLion.ObjectPool.IReceivable;

using IReleasable = FuryLion.ObjectPool.IReleasable;

namespace Merge.Core

{

    public class TokenWrapper : Element, IReleasable, IReceivable

    {

        [SerializeField] private DragElement \_dragElement;

        [SerializeField] private Transform \_container;

        [SerializeField] private Transform \_tokenLightContainer;

        [SerializeField] private Image \_taskHighlight;

        [SerializeField] private Image \_webBackground;

        [SerializeField] private AnimationCurve \_spawnMoveEase;

        [SerializeField] private Image \_checkImage;

        [SerializeField] private Transform \_checkTransform;

        public static event Action<TokenWrapper, Cell> DroppedOver;

        public static event Action<TokenWrapper, Cell> HoverEnter;

        public static event Action<TokenWrapper, Cell> HoverExit;

        public static event Action<TokenWrapper> Click;

        public static event Action<TokenWrapper> Down;

        public static event Action<TokenWrapper> Up;

        public static event Action<TokenWrapper> Inited;

        public static event Action<TokenWrapper> Upgraded;

        public static event Action<TokenWrapper, bool> DragStateChanged;

        public static event Action<TokenWrapper> DroppedIntoStorage;

        public static event Action<TokenWrapper> DroppedIntoSeller;

        public static event Action<TokenWrapper> DroppedIntoEquipment;

        public static event Action<TokenWrapper> Reloaded;

        public static event Action<TokenWrapper> CheckStateUpdated;

        private static int \_defaultDragZPos = -10;

        public static int DragZPosition { get; private set; } = -25;

        private static readonly Color32 TaskColor = Color.white;

        private static readonly Color32 TaskHiddenColor = TaskColor.A(0);

        public GameController GameController { get; private set; }

        private bool \_isDragging;

        private bool \_wasMovedByDrag;

        private TokenSelector \_selector;

        private Tween \_taskTween;

        private Tween \_hintTween;

        private Tween \_selectTween;

Продолжение листинга 3.6

        private Tween \_upgradeHintTween;

        private Tween \_flyTween;

        private Tween \_checkTween;

        private Sequence \_spawnSequence;

        private Sequence \_spawnSequenceScale;

        private PoolableVfxWithGlow \_upgradeHintVfx;

        public bool IsDisabled { get; private set; }

        public BaseToken Token { get; private set; }

        public int I => Token.I;

        public int J => Token.J;

        public (int i, int j) Index => (I, J);

        public TokenData TokenData => Token.TokenData;

        public DragElement DragElement => \_dragElement;

        public Transform Container => \_container;

        public bool IsBoxSpawn { get; set; }

        public bool IsBubbleSpawn { get; set; }

        public bool DragEnabled

        {

            get => \_dragElement.gameObject.activeSelf;

            set

            {

                if (\_isDragging)

                    \_dragElement.OnRelease(new InputPoint {WorldPosition = transform.position});

                \_dragElement.gameObject.SetActive(value);

            }

        }

        public bool IsQuestItem { get; private set; }

        public bool HintAvailable { get; private set; }

        public void OnReceived()

        {

            \_dragElement.DroppedOver += OnDroppedOver;

            \_dragElement.DragStateChanged += OnDragStateChanged;

            \_dragElement.Click += OnClick;

            \_dragElement.Down += OnDown;

            \_dragElement.Up += OnUp;

            \_dragElement.HoverEnter += OnHoverEnter;

            \_dragElement.HoverExit += OnHoverExit;

            \_dragElement.DroppedIntoStorage += OnDroppedIntoStorage;

            \_dragElement.DroppedIntoSeller += OnDroppedIntoSeller;

Продолжение листинга 3.6

            \_dragElement.DroppedIntoEquipment += OnDroppedIntoEquipment;

            DragEnabled = true;

        }

        public void Init(TokenData tokenData, GameController gameController)

        {

            GameController = gameController;

            if (Token != null)

                MergePool.Release(Token);

            Token = GetToken(tokenData);

            \_taskHighlight.gameObject.SetActive(false);

            \_taskHighlight.Color = TaskHiddenColor;

            \_checkImage.gameObject.SetActive(false);

            \_checkImage.SetAlpha(1f);

            gameObject.name = $"{Token.gameObject.name} [Wrapper]";

            SetChecked(false, false);

            \_dragElement.transform.SetLocalPositionXY(Vector2.zero);

            Inited?.Invoke(this);

        }

        public virtual void OnSetupOnGrid()

        {

            Token.OnSetupOnGrid();

        }

        public bool DataIsEquals(TokenData tokenData)

        {

            if (Token == null)

                return false;

            return TokenData.Type == tokenData.Type && TokenData.Level == tokenData.Level;

        }

        public bool IsCheckEnabled

        {

            set => \_checkTransform.gameObject.SetActive(value);

        }

Продолжение листинга 3.6

        public void SetChecked(bool value, bool withAnimation)

        {

            if (value && \_isDragging)

                return;

            var targetScale = value ? Vector3.one : Vector3.zero;

            var targetColor = value ? TaskColor : TaskHiddenColor;

            \_checkTween?.Kill();

            if (withAnimation)

            {

                if (value)

                {

                    \_checkImage.gameObject.SetActive(true);

                    \_taskHighlight.gameObject.SetActive(true);

                }

                var ease = value ? Ease.OutBack : Ease.InBack;

                \_checkTween = DOTween.Sequence()

                    .Join(\_checkTransform.DOScale(targetScale, 0.2f)

                        .SetEase(ease)

                        .OnComplete(() =>

                        {

                            if (!value)

                                \_checkImage.gameObject.SetActive(false);

                        }))

                    .Join(\_taskHighlight.DOColor(targetColor, 0.2f)

                        .OnComplete(() =>

                            {

                                if (!value)

                                    \_taskHighlight.gameObject.SetActive(false);

                            }

                        ));

                if (value && !IsQuestItem)

                    Sounds.Merge.m\_obj\_collect\_true.PlaySound();

            }

            else

            {

                \_checkImage.gameObject.SetActive(value);

                \_checkTransform.SetLocalScale(targetScale);

                \_taskHighlight.Color = targetColor;

                \_taskHighlight.gameObject.SetActive(value);

            }

            IsQuestItem = value;

Продолжение листинга 3.6

            CheckStateUpdated?.Invoke(this);

        }

        public void Disable()

        {

            IsDisabled = true;

        }

        public void Enable()

        {

            IsDisabled = false;

        }

        [ContextMenu("Upgrade")]

        public void Upgrade()

        {

            \_flyTween?.Kill();

            \_selectTween?.Kill();

            var data = Token.TokenData;

            var config = TokensConfig.GetConfig(data.Type, data.Level + 1);

            var newData = config.CreateData();

            newData.I = data.I;

            newData.J = data.J;

            MergePool.Release(Token);

            Token = GetToken(newData);

            Token.OnSetupOnGrid();

            Token.ShowUpgrade();

            gameObject.name = $"{Token.gameObject.name} [Wrapper]";

            Upgraded?.Invoke(this);

        }

        private BaseToken GetToken(TokenData tokenData)

        {

            IsDisabled = false;

            var modifier = tokenData.Modifier;

            var token = ConstantsConfig.IsLifeSimFlow

                ? MergePool.Get<Token>("Token")

                : GameController.GetToken(modifier);

            token.gameObject.SetupToParent(\_container.gameObject);

            token.SetLocalPosition(Vector3.zero);

Продолжение листинга 3.6

            token.Init(tokenData, GameController);

            token.OnSetupOnWrapper(this);

            IsDisabled = !token.IsActive;

            HintAvailable = token.IsHintAvailable;

            \_dragElement.CanDrag = token.CanDrag;

            \_dragElement.BoxCollider.enabled = token.CanClick;

            \_webBackground.gameObject.SetActive(!ConstantsConfig.IsLifeSimFlow &&

                                                GameController.GameId == GameService.MainContainerId &&

                                                ConstantsConfig.MergeArtConfig.UseNewWebBackground &&

                                                tokenData.Modifier is LockedTokenModifier);

            if (!IsBoxSpawn && !IsBubbleSpawn)

            {

                var vfxPosition = token.transform.position;

                vfxPosition.z -= 10;

                FuryLion.ObjectPool.VFX.Get<BasePoolableVfxView>("CloudsDisappearVfx").Play(vfxPosition);

                return token;

            }

            if (IsBoxSpawn)

            {

                token.SetLocalScaleXY(0, 0);

                token.transform.DOScaleXY(1f, 0.35f)

                    .SetEase(Ease.OutBack);

                IsBoxSpawn = false;

            }

            if (IsBubbleSpawn)

            {

                SpawnedFrom(transform.position);

                IsBubbleSpawn = false;

            }

            return token;

        }

        public void SetSize(float size)

        {

            \_dragElement.BoxCollider.size = new Vector2(size, size);

        }

        private void OnDragStateChanged(bool isDragging)

Продолжение листинга 3.6

        {

            Token.transform.DOKill();

            \_isDragging = isDragging;

            \_wasMovedByDrag |= isDragging;

            if (\_selector != null)

                \_selector.ChangeVisibility(!isDragging);

            if (HintAvailable)

            {

            }

            DragStateChanged?.Invoke(this, isDragging);

            if (isDragging)

                SetChecked(false, false);

            if (Token is ICallToActionBubble eater)

                eater.SetBubbleLoopEnabled(!isDragging);

            if (Token is not TokenBooster booster)

                return;

            if (isDragging)

                booster.OnDragStarted();

            else

                booster.OnDragEnded();

        }

        private void OnDroppedOver(Cell cell)

        {

            DroppedOver?.Invoke(this, cell);

        }

        private void OnHoverEnter(Cell cell)

        {

            HoverEnter?.Invoke(this, cell);

        }

        private void OnHoverExit(Cell cell)

        {

            HoverExit?.Invoke(this, cell);

        }

        private void OnDroppedIntoStorage()

        {

            DroppedIntoStorage?.Invoke(this);

        }

        private void OnDroppedIntoSeller()

Продолжение листинга 3.6

        {

            DroppedIntoSeller?.Invoke(this);

        }

        private void OnDroppedIntoEquipment()

        {

            DroppedIntoEquipment?.Invoke(this);

        }

        private void OnClick()

        {

            Token.OnClick();

            Click?.Invoke(this);

        }

        private void OnDown()

        {

            \_selectTween?.Kill();

            \_container.transform.SetLocalScale(Vector3.one);

            \_selectTween = DOTween.Sequence()

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one \* 1.1f, 0.133f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one \* 0.94f, 0.233f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one \* 1.055f, 0.266f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one \* 0.94f, 0.266f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one \* 1.02f, 0.33f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one \* 0.98f, 0.433f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one, 0.33f)

                    .SetEase(Ease.InOutSine)

                )

Продолжение листинга 3.6

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one, 1.33f)

                    .SetEase(Ease.InOutSine)

                );

            Down?.Invoke(this);

        }

        private void OnUp()

        {

            \_wasMovedByDrag = false;

            Up?.Invoke(this);

        }

        private void Update()

        {

            if (\_isDragging)

            {

                if (Token == null)

                {

                    \_isDragging = false;

                    return;

                }

                var lerpPos =

                    Vector3.Lerp(

                        Token.transform.position,

                        \_dragElement.transform.position,

                        0.5f

                    );

                Token.transform.SetLocalPositionZ(DragZPosition);

                Token.SetPositionXY(lerpPos.x, lerpPos.y);

            }

        }

        public void Return(bool wrongMerge)

        {

            if (\_wasMovedByDrag)

            {

                if (wrongMerge)

                    Sounds.Merge.m\_obj\_false.PlaySound();

                else

                    Sounds.Merge.m\_obj\_drop.PlaySound();

            }

            \_dragElement.transform.SetLocalPositionXY(Vector2.zero);

            Token.transform.SetLocalPositionZ(0);

Продолжение листинга 3.6

            Token.transform.DOLocalMoveXY(Vector2.zero, 0.2f).SetEase(Ease.OutBack);

        }

        public void Place(int i, int j)

        {

            var tokenViewPosition = Token.transform.position;

            \_dragElement.transform.SetLocalPositionXY(Vector2.zero);

            transform.SetLocalPositionXY(0, 0);

            Token.transform.position = tokenViewPosition;

            Token.transform.SetLocalPositionZ(0);

            Token.transform.DOLocalMoveXY(Vector2.zero, 0.2f).SetEase(Ease.OutBack);

            Token.Place(i, j);

        }

        public bool CanUpgrade(TokenWrapper other)

        {

            if (ConstantsConfig.IsLifeSimFlow)

                return Token.CanMerge(other.Token);

            return Token.CanMerge(other.Token) && GameController.GetMechanics<OverMergeLockMechanics>().IsMergeAvailable(this, other);

        }

        public void Select()

        {

            if (\_selector == null)

            {

                \_selector = MergePool.Get<TokenSelector>("TokenSelector");

                \_selector.gameObject.SetupToParent(gameObject);

                \_selector.DoBreathe();

            }

        }

        public void Deselect()

        {

            \_selectTween?.Kill();

            ReleaseSelector();

            if (Token is ITokenTimerUpdatable token)

                token.TimerTitleText = null;

        }

        // public async UniTask FlyTo(TaskItemElement taskItemElement)

Продолжение листинга 3.6

        public async UniTask FlyTo(TaskCell taskItemElement)

        {

            \_dragElement.CanDrag = false;

            DragZPosition -= 1;

            var zPos = DragZPosition;

            // Token.transform.SetLocalPositionZ(zPos);

            Token.transform.SetLocalPositionZ(taskItemElement.transform.position.z - 2f);

            \_selectTween?.Kill();

            \_flyTween?.Kill();

            \_flyTween = Token.transform

                // .DOMoveXY(taskItemElement.transform.position, 0.5f)

                .DOMoveXY(taskItemElement.transform.position, 0.5f)

                .SetEase(Ease.InSine)

                .OnComplete(() =>

                {

                    if (zPos == DragZPosition)

                        DragZPosition = \_defaultDragZPos;

                });

            await \_flyTween.AwaitForComplete();

            taskItemElement.Bounce();

        }

        public void MoveTo(Vector3 spawnPosition, Vector3 startScale = default, Action callback = null)

        {

            IsCheckEnabled = true;

            DragZPosition -= 1;

            var zPos = DragZPosition;

            spawnPosition.z -= 3f;

            Token.transform.SetLocalPositionZ(zPos);

            Token.transform.SetLocalScaleXY(startScale != default ? startScale : new Vector2(0.65f, 0.65f));

            Token.transform.DOKill();

            \_spawnSequence?.Kill();

            var moveTween = DOTween.Sequence()

                .Append(Token.transform

                    .DOLocalPath(

                        FormPath(Token.transform.localPosition, spawnPosition), 0.676f,

Продолжение листинга 3.6

                        PathType.CatmullRom)

                    .SetEase(\_spawnMoveEase)

                    .OnComplete(() =>

                    {

                        Token.transform.SetLocalPositionZ(0);

                        if (zPos == DragZPosition)

                            DragZPosition = \_defaultDragZPos;

                    }));

            var scaleTween = DOTween.Sequence()

                .Append(Token.transform

                    .DOScaleXY(1.1f, 0.2f)

                    .SetEase(Ease.InOutSine))

                .Append(Token.transform

                    .DOScaleXY(0.3f, 0.2f)

                    .SetEase(Ease.InOutSine))

                .Append(Token.transform

                    .DOScaleXY(0.4f, 0.2f)

                    .SetEase(Ease.InOutSine))

                .Append(Token.transform

                    .DOScaleXY(0f, 0.1f)

                    .SetEase(Ease.InOutSine))

                .Play();

            \_spawnSequence = DOTween.Sequence()

                .Join(moveTween)

                .Join(scaleTween)

                .OnComplete(() => callback?.Invoke());

        }

        public void SpawnedFrom(Vector3 spawnPosition, Vector3 startScale = default, Action callback = null)

        {

            IsCheckEnabled = true;

            var tokenTransform = Token.transform;

            tokenTransform.SetLocalPosition(Vector3.zero);

            DelayedCall.Run(0.55f, PlayLandingVfx);

            DragZPosition -= 1;

            var zPos = DragZPosition;

            spawnPosition.z -= 3f;

            tokenTransform.SetLocalPositionZ(zPos);

            tokenTransform.SetPosition(spawnPosition);

            tokenTransform.SetLocalScaleXY(startScale != default ? startScale : new Vector2(0.65f, 0.65f));

Продолжение листинга 3.6

            tokenTransform.DOKill();

            \_spawnSequence?.Kill();

            var moveTween = DOTween.Sequence()

                .Append(tokenTransform

                    .DOLocalPath(

                        FormPath(tokenTransform.localPosition, new Vector3(0, 0, tokenTransform.localPosition.z)), 0.676f,

                        PathType.CatmullRom)

                    .SetEase(\_spawnMoveEase)

                    .OnComplete(() =>

                    {

                        tokenTransform.SetLocalPositionZ(0);

                        if (zPos == DragZPosition)

                            DragZPosition = \_defaultDragZPos;

                    }));

            var scaleTween = DOTween.Sequence()

                .Append(tokenTransform

                    .DOScaleXY(1.7f, 0.2f)

                    .SetEase(Ease.InOutSine))

                .Append(tokenTransform

                    .DOScaleXY(0.8f, 0.2f)

                    .SetEase(Ease.InOutSine))

                .Append(tokenTransform

                    .DOScaleXY(1.1f, 0.2f)

                    .SetEase(Ease.InOutSine))

                .Append(tokenTransform

                    .DOScaleXY(1f, 0.1f)

                    .SetEase(Ease.InOutSine))

                .Play();

            \_spawnSequence = DOTween.Sequence()

                .Join(moveTween)

                .Join(scaleTween)

                .OnComplete(() => callback?.Invoke());

        }

        private Vector3[] FormPath(Vector3 startPosition, Vector3 targetPosition)

        {

            var pos = new[]

            {

                startPosition,

                new Vector3(

                    startPosition.x / 2 + (startPosition.x > 0 ? -0.9f : 0.9f),

Продолжение листинга 3.6

                    startPosition.y / 2,

                    startPosition.z),

                targetPosition

            };

            return pos;

        }

        private void PlayLandingVfx()

        {

            var spawnVfx = FuryLion.ObjectPool.VFX.Get<BasePoolableVfxView>("SpawnDustVfx");

            spawnVfx.Play(transform.position, transform);

            spawnVfx.transform.SetLocalRotation(-90, 0, 0);

        }

        public void DoHint(Vector3 position, bool withSound = false)

        {

            DoHintToPosition(position, withSound);

        }

        private void DoScaleHint()

        {

            \_hintTween?.Kill();

            \_container.transform.localPosition = Vector3.zero;

            var doForward = \_container.transform

                .DOScaleXY(new Vector2(1.1f, 1.1f), 0.4f)

                .SetEase(Ease.InCubic);

            var doBackward = \_container.transform

                .DOScaleXY(Vector2.one, 0.6f)

                .SetEase(Ease.InOutSine);

            \_hintTween = DOTween.Sequence()

                .Append(doForward)

                .Append(doBackward)

                .SetLoops(-1);

        }

        private void DoHintToPosition(Vector3 position, bool withSound = false)

        {

            if (Token is TokenBooster booster)

                booster.StopAnimation();

            \_selectTween?.Kill();

            \_hintTween?.Kill();

Продолжение листинга 3.6

            \_container.transform.localPosition = new Vector3(0, 0, -5);

            var direction = (position - transform.position).normalized;

            var doScale = DOTween.Sequence()

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one, 0.166f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector3.one \* 1.1f, 0.3f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one, 0.5f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one \* 1.1f, 0.366f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOScaleXY(Vector2.one, 0.4f)

                    .SetEase(Ease.InOutSine)

                );

            var doForward = DOTween.Sequence()

                .Append(\_container.transform

                    .DOLocalMoveXY(direction \* 0.3f, 0.4f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOLocalMoveXY(direction \* 0.1f, 0.433f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOLocalMoveXY(direction \* 0.3f, 0.366f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOLocalMoveXY(Vector3.zero, 0.733f)

                    .SetEase(Ease.InOutSine)

                )

                .Append(\_container.transform

                    .DOLocalMoveXY(Vector3.zero, 0.733f)

                    .SetEase(Ease.InOutSine)

                );

Продолжение листинга 3.6

            if (withSound)

            {

                void PlaySound() => Sounds.Merge.m\_obj\_hint.PlaySound();

                doForward.OnStart(PlaySound);

                doForward.OnRewind(PlaySound);

            }

            // var doBackward = DOTween.Sequence()

            //     .Join(\_container.transform

            //         .DOLocalMoveXY(Vector3.zero, 0.6f)

            //         .SetEase(Ease.InOutSine)

            //     )

            //     .Join(\_container.transform

            //         .DOScaleXY(Vector2.one, 0.6f)

            //         .SetEase(Ease.InOutSine)

            //     );

            \_hintTween = DOTween.Sequence()

                .Join(doScale)

                .Join(doForward)

                .SetLoops(-1);

        }

        public void StopHint()

        {

            if (\_hintTween == null || !\_hintTween.IsPlaying())

                return;

            if (Token is TokenBooster booster)

                booster.LoopAnimation();

            \_hintTween?.Kill();

            \_container.transform.SetLocalPositionZ(0);

            \_hintTween = DOTween.Sequence()

                .Join(\_container.transform.DOLocalMoveXY(Vector3.zero, 0.2f))

                .Join(\_container.transform.DOScaleXY(Vector2.one, 0.2f));

        }

        public void UpgradeHint(bool isYellow)

        {

            if (\_upgradeHintVfx != null)

                return;

            transform.SetLocalPositionZ(-3f);

Продолжение листинга 3.6

            var vfxName = isYellow ? "MergeHintVfx" : "PurpleMergeHintVfx";

            \_upgradeHintVfx = FuryLion.ObjectPool.VFX.Get<PoolableVfxWithGlow>(vfxName);

            \_upgradeHintVfx.Play(\_tokenLightContainer, Vector3.zero);

            \_selectTween?.Kill();

            \_upgradeHintTween?.Kill();

            \_upgradeHintTween = \_container.transform

                .DOScaleXY(1.1f, 0.5f)

                .SetEase(Ease.InOutSine);

        }

        public void StopUpgradeHint(bool force = false)

        {

            if (\_upgradeHintVfx == null)

                return;

            transform.SetLocalPositionZ(-1f);

            var vfx = \_upgradeHintVfx;

            \_upgradeHintVfx = null;

            vfx.Stop();

            DelayedCall.Run(force ? 0f : 3f, () => vfx.Release());

            \_selectTween?.Kill();

            \_upgradeHintTween?.Kill();

            \_upgradeHintTween = \_container.transform

                .DOScaleXY(1f, 0.2f)

                .SetEase(Ease.InOutSine);

        }

        public void DoFade(float fade)

        {

            \_checkImage.DOFade(fade, 0.2f);

            Token.DoFade(fade);

        }

        public void OnReleased()

        {

            \_wasMovedByDrag = false;

            \_isDragging = false;

            \_selectTween?.Kill();

            \_flyTween?.Kill();

            \_spawnSequence?.Kill();

            \_taskTween?.Kill();

            \_hintTween?.Kill();

Продолжение листинга 3.6

            \_upgradeHintTween?.Kill();

            \_taskHighlight.Color = TaskHiddenColor;

            \_dragElement.transform.SetLocalPositionXY(Vector2.zero);

            \_container.transform.localPosition = Vector3.zero;

            \_container.transform.SetLocalScaleXY(1, 1);

            transform.rotation = Quaternion.identity;

            \_dragElement.DroppedOver -= OnDroppedOver;

            \_dragElement.DragStateChanged -= OnDragStateChanged;

            \_dragElement.Click -= OnClick;

            \_dragElement.Down -= OnDown;

            \_dragElement.Up -= OnUp;

            \_dragElement.HoverEnter -= OnHoverEnter;

            \_dragElement.HoverExit -= OnHoverExit;

            \_dragElement.DroppedIntoStorage -= OnDroppedIntoStorage;

            \_dragElement.DroppedIntoSeller -= OnDroppedIntoSeller;

            \_dragElement.DroppedIntoEquipment -= OnDroppedIntoEquipment;

            DragEnabled = true;

            StopUpgradeHint();

            ReleaseSelector();

            ReleaseToken();

        }

        public void ReleaseSelector()

        {

            if (\_selector != null)

            {

                MergePool.Release(\_selector);

                \_selector = null;

            }

        }

        private void ReleaseToken()

        {

            if (Token == null)

                return;

            MergePool.Release(Token);

            Token = null;

        }

        public void ReloadToken(TokenData data)

        {

            data.I = I;

Продолжение листинга 3.6

            data.J = J;

            Init(data, GameController);

            OnSetupOnGrid();

            Reloaded?.Invoke(this);

        }

        public void ReloadToken()

        {

            ReloadToken(TokenData);

        }

        [ContextMenu("Remove")]

        private void DebugRemove()

        {

            GameController.Grid.Remove(this);

        }

    }

}

В ходе реализации программного продукта была разработана мобильная гибридно-казуальная игра «Jess` Stories», при разработке которой было использовано множество стандартных для сферы разработки игр паттернов программирования. Данный программный продукт отвечает всем требованиям поставленного технического задания.

Представление результатов реализации программного продукта предоставлено в документе «Описание программы», приложение В.

## Результаты реализации клиенткой части

(описать точно также как и серверную часть).

## Разработка программной документации

В результате создания программного продукта была разработана следующая программная документация:

* техническое задание (приложение А);
* описание программы (приложение В);
* программа и методика испытаний (приложение Г);
* руководство пользователя (приложение Д).

## Тестирование

Тестирование игр – неотъемлемая часть процесса разработки видеоигр. Этап тестирования включает в себя выявление ошибок и предоставление игрокам возможных путей реакции на игру. При создании видеоигр процесс тестирования может присутствовать как во время прототипирования, разработки и мягкого запуска, так и после релиза. Роль тестирования игры заключается в анализе, выявлении, документировании и исправлении всех дефектов программного обеспечения, которые могут негативно повлиять на конечный игровой опыт. Это фактор, который определяет, сможет ли игра стать успешной, что напрямую зависит от опыта игроков.

Типы тестирования игр:

1. Комбинаторное тестирование. Этот тип тестирования игр помогает повысить эффективность приложений и выполнения тестов, предполагая, что степень обнаружение ошибок повышается посредством перебора комбинаций различных параметров. Этот метод тестирования выделяет уникальные атрибуты и старается охватить все возможные комбинации параметров.

2. Функциональное тестирование. Этот метод направлен на решение проблем, связанных с игровым контентом и графикой, и его цель – подтвердить, соответствует ли получившаяся игра исходным спецификациям. Функциональное тестирование – это сложный этап, требующий времени для выявления проблем с графикой, аудиовизуальными проблемами и так далее.

3. Произвольное (специальное) тестирование. Для этого метода тестирования игры не требуется какой-либо изначальной документации или процесса планирования – оно выполняется чисто интуитивным образом. Целью специального тестирования является обеспечение полноты тестирования и обнаружение неочевидных ошибок или багов.

4. Тестирование совместимости. Проводится с целью проверить, оптимизирован ли пользовательский интерфейс для различных размеров экранов различных устройств, таких как мобильные телефоны, ПК и консоли. Также с помощью данной методики проверяется совместимость игры на тех же самых разных игровых платформах. Своевременное обнаружение и устранение неполадок графики, программного обеспечения или оборудования делает пользователя удовлетворенным продуктом благодаря хорошему игровому процессу.

5. Регрессионное тестирование. Этот метод тестирования игр можно охарактеризовать как повторное тестирование функциональности игровых процессов. Регрессионным тестированием проверяется, не появились ли после новых изменений игры баги или ошибки в старых, уже проверенных компонентах. Таким образом, качество игры поддерживается на высоком уровне. [22]

В ходе создания программного продукта, в качестве метода тестирования было выбрано функциональное тестирование.

Функциональные тесты основываются на функциях, выполняемых системой, и могут проводиться на всех уровнях тестирования. Как правило, эти функции описываются в требованиях, функциональных спецификациях или в виде случаев использования системы (use cases).

Результаты функционального тестирования разработанного программного продукта продемонстрированы в таблице Г.1 (приложение Г).

В результате тестирования мобильная гибридно-казуальная игра «Jess’ Stories» продемонстрировала себя, как готовый к эксплуатации программный продукт, стабильный в использовании и с интуитивно понятным и приятным пользовательским интерфейсом.

Критических дефектов или ошибок, ограничивающих пользователя в его действиях во время игрового процесса и не только, в ходе функционального тестирования выявлено не было.

## ЗАКЛЮЧЕНИЕ

В результате реализации дипломного проекта была спроектирована и создана мобильная гибридно-казуальная игра «Jess’ Stories».

В ходе выполнения дипломного проекта были решены следующие заранее поставленные задачи:

* изучена предметная область;
* проведен анализ аналогов и прототипов с описанием преимуществ и недостатков рассматриваемых готовых продуктов и их устройства;
* выявлены требования к разрабатываемому программному продукту;
* разработано технического задание;
* спроектирован программный продукт с учетом технического задания;
* на основе проектирования разработан программный продукт;
* протестирован готовый программный продукт;
* проведен расчет цен и себестоимости программного продукта;
* разработана сопроводительная документация – «Описание программы», «Программа и методика испытаний», «Руководство пользователя».

При разработке программного продукта использовалась платформа для разработки игр Unity, высокоуровневый язык программирования C#, использующий в основе написания программного кода платформу .NET. В качестве среды разработки использовалась IDE – JetBrains Rider. При проектировании и реализации программного кода были использованы свойственные для сферы разработки игр порождающие, поведенческие и структурные паттерны программирования.

Реализованный программный продукт полностью соответствует требованиям технического задания и готов к эксплуатации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Игровая магия: технологии и инновации в мире мобильных игр ­– itWeek [Электронный ресурс]. – Режим доступа: <https://www.itweek.ru/mobile/article/detail.php?ID=227503> – Дата доступа: 21.03.2024;
2. Hybrid-Casual ­– APPLOVIN [Электронный ресурс]. – Режим доступа: <https://applovin.com/glossary/hybrid-casual/> – Дата доступа: 21.03.2024;
3. Что такое гиперказуальные игры и почему они популярны­– XYZ Media [Электронный ресурс]. – Режим доступа: <https://media-xyz.com/ru/articles/794-chto-takoe-giperkazualnye-igry-i-pochemu-oni-p> – Дата доступа: 21.03.2024;
4. Что такое казуальные игры – CQ.RU [Электронный ресурс]. – Режим доступа: <https://cq.ru/articles/gaming/chto-takoe-kazualnye-igry> – Дата доступа: 21.03.2024;
5. Comics Bob – Uptodown [Электронный ресурс]. – Режим доступа: <https://comics-bob.ru.uptodown.com/android> – Дата доступа: 23.03.2024;
6. Love & Pies - Merge – Uptodown [Электронный ресурс]. – Режим доступа: <https://love-and-pies.en.uptodown.com/android> – Дата доступа: 23.03.2024;
7. Makeover Merge Android – Malavida [Электронный ресурс]. – Режим доступа: <https://makeover-merge.ru.malavida.com/android/> – Дата доступа: 23.03.2024;
8. ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ ДЛЯ ВЕБ-РАЗРАБОТКИ: КАК ОПРЕДЕЛИТЬСЯ С ТЕХНОЛОГИЕЙ – Калининградский бизнес-колледж [Электронный ресурс]. – Режим доступа: https://student39.ru/blog/obrazovanie/vybor-yazyka-programmirovaniya-dlya-veb-razrabotki-kak-opredelitsya-s-tekhnologiey/ – Дата доступа: 24.03.2024;
9. Язык C# и платформа .NET – METANIT.COM [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/1.1.php – Дата доступа: 25.03.2024;
10. Язык программирования C++ – METANIT.COM [Электронный ресурс]. – Режим доступа: <https://metanit.com/cpp/tutorial/1.1.php> – Дата доступа: 25.03.2024;
11. C++ и искусство разработки игр: Секреты создания захватывающих игровых миров – Дзен [Электронный ресурс]. – Режим доступа: https://dzen.ru/a/ZXhDNRzWjhysuEeS – Дата доступа: 25.03.2024;
12. Разработка игр на JavaScript – VC.ru [Электронный ресурс]. – Режим доступа: <https://vc.ru/u/326339-dev-house/848499-razrabotka-igr-na-javascript> – Дата доступа: 25.03.2024;
13. Unity: что это за движок, для чего нужен и почему так популярен – Skillbox Media [Электронный ресурс]. – Режим доступа: https://skillbox.ru/media/gamedev/unity-chto-eto-za-dvizhok-dlya-chego-nuzhen-i-pochemu-tak-populyaren/ – Дата доступа: 25.03.2024;
14. Unreal Engine – Skillfactory Media [Электронный ресурс]. – Режим доступа: <https://blog.skillfactory.ru/glossary/unreal-engine/> – Дата доступа: 25.03.2024;
15. Phaser – Wikipedia.org [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Phaser> – Дата доступа: 25.03.2024;
16. UML – диаграмма вариантов использования (use case diagram) – Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/47940/ – Дата доступа: 26.03.2024;
17. Одиночка – Refactoring guru [Электронный ресурс]. – Режим доступа: <https://refactoring.guru/ru/design-patterns/singleton> – Дата доступа: 27.03.2024;
18. Паттерн Object Pool (пул объектов) – CPP-REFERENCE [Электронный ресурс]. – Режим доступа: <http://cpp-reference.ru/patterns/creational-patterns/object-pool/> – Дата доступа: 27.03.2024;
19. Адаптер – Refactoring guru [Электронный ресурс]. – Режим доступа: https://refactoring.guru/ru/design-patterns/adapter – Дата доступа: 27.03.2024;
20. Наблюдатель – Refactoring guru [Электронный ресурс]. – Режим доступа: <https://refactoring.guru/ru/design-patterns/observer> – Дата доступа: 27.03.2024;
21. Стратегия – Refactoring guru [Электронный ресурс]. – Режим доступа: <https://refactoring.guru/ru/design-patterns/strategy> – Дата доступа: 27.03.2024;
22. Тестирование игр: лучшие практики и новые тренды – Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/otus/articles/556784/ – Дата доступа: 27.03.2024;
23. The Mobile eCPM Report: In-App Ad Monetization Worldwide [Updated in April 2024] – appodeal [Электронный ресурс]. – Режим доступа: https://appodeal.com/blog/mobile-ecpm-report-app-ad-monetization-worldwide-performance/ – Дата доступа: 20.05.2024;

# ПРИЛОЖЕНИЕ А

# (обязательное)

**Техническое задание**

## Введение

Наименование продукта: мобильная гибридно-казуальная игра «Jess’ Stories».

Данный продукт будет использоваться для развлечения пользователя. Он является представителем класса игры, находящихся на своем пике популярности в данный момент. Содержит в себе глубокие механики, замаскированные под простой и приятный интерфейс с легким управлением.

## А.1 Основание для разработки

Основанием для разработки является приказ №\_\_\_ от .05.2024. «О закреплении тем дипломных проектов студентов факультета информационных технологий специальности 1-40-01-01 «Программное обеспечение информационных технологий» по кафедре технологий программирования УО «Полоцкий государственный университет имени Евфросинии Полоцкой».

Тема разработки – «Игровое приложение «Jess’ Stories» в жанре гибридказуал» выданная студенту группы 19-ИТзд Соловьёву Илье Александровичу, руководителем назначен старший преподаватель кафедры технологий программирования Михнович Ольга Владимировна.

## А.2 Назначение разработки

Основной целью разработки игрового приложения является разнообразие досуга пользователя. Приложение предоставляет следующие возможности:

* отображение анимированных сюжетных сценок;
* возможность выбора развития сюжета в сценках;
* возможность зарабатывать внутриигровую валюту посредством игры в жанре «Merge-2»;
* возможность использовать внутриигровую валюту для изменения внешности главного игрового персонажа;
* возможность использовать внутриигровую валюту для проведения уборки в доме с последующим выбором новой мебели;
* возможность приобретать внутриигровую валюту за реальные деньги;
* опциональный просмотр рекламных роликов для получения внутриигровых бонусов;
* установка параметров настройки игры;
* сохранение игрового прогресса.

## А.3 Требования к программе или программному изделию

**А.3.1 Требования к функциональным характеристикам**

К функциональным характеристикам выдвигаются следующие требования:

* пользователь может самостоятельно менять базовые настройки игры, как например – отключить звук;
* пользователь может проходить сюжетные сценарии один за другим, если у него достаточно внутриигровой валюты;
* пользователь может зарабатывать внутриигровую валюту посредством игры в «Merge-2» режиме для совершенствования персонажа и его окружения;
* комнаты в доме будут открываться постепенно: после полного обустройства в комнате будет открываться следующая, в которой необходимо будет убрать мусор и расставить новую мебель;
* по мере прохождения игры в «Merge-2» режиме должны открываться предметы с новыми игровыми механиками;
* пользовательский интерфейс должен быть интуитивно понятным и должен обеспечивать возможность взаимодействия со всем функционалом игры.

**А.3.2 Требования к надежности**

Программное средство должно обеспечивать контроль корректности входных данных. В случае обнаружения ошибок должны срабатывать обработчики ошибок, которые будут сообщать пользователю основную информацию об ошибке без отказа приложения с соответствующим продолжением пользования игрой.

**А.3.3 Условия эксплуатации**

Данное программное средство не требует предварительной подготовки.

**А.3.4 Требования к составу и параметрам технических средств**

Минимальными и достаточными требованиями по конфигурации оборудованиям клиента для функционирования системы являются:

* версия Android 8.0 и выше или iOS 12 и выше;
* объем оперативной памяти не менее 1024 MB;
* свободное пространство в памяти устройства не менее 500 MB.

**А.3.5 Требования к информационной и программной совместимости**

Для программного обеспечения клиента требуется операционная система: Android 8.0 и выше или iOS 12 и выше.

**А.3.6 Требования к маркировке и упаковке**

Требования к маркировке и упаковке отсутствуют.

**А.3.7 Требования к транспортированию и хранению**

Требования к транспортировке и хранению отсутствуют.

## А.4 Требования к программной документации

Программная документация должна состоять из следующих документов:

* техническое задание в соответствии с ГОСТ 19.201-78 ЕСПД;
* описание программы в соответствии с ГОСТ 19.402-2000 ЕСПД;
* руководство оператора в соответствии с ГОСТ 19.505-79 ЕСПД;
* программа и методика испытаний в соответствии с ГОСТ 19.301-2000 ЕСПД.

Содержание и структура программной документации соответствует требованиям ЕСПД.

## А.5 Технико-экономические показатели

Эффективность данного программного продукта обуславливается совокупностью следующих факторов:

* популярность гибридно-казуального жанра игр;
* популярность игр в жанре «Merge-2»;
* облегчение задачи отдела маркетинга за счет использования в игре интерактивных анимированных сценок.

А также большим бонусом является тот факт, что мобильный телефон постоянно присутствует рядом с пользователем, таким образом у игрока всегда имеется доступ к игре.

## А.6 Этапы разработки

Этапы разработки:

* постановка задачи;
* анализ исходных данных;
* разработка программного средства;
* тестирование программного средства;
* разработка технической документации к данному программному средству.

## А.7 Порядок контроля и приемки

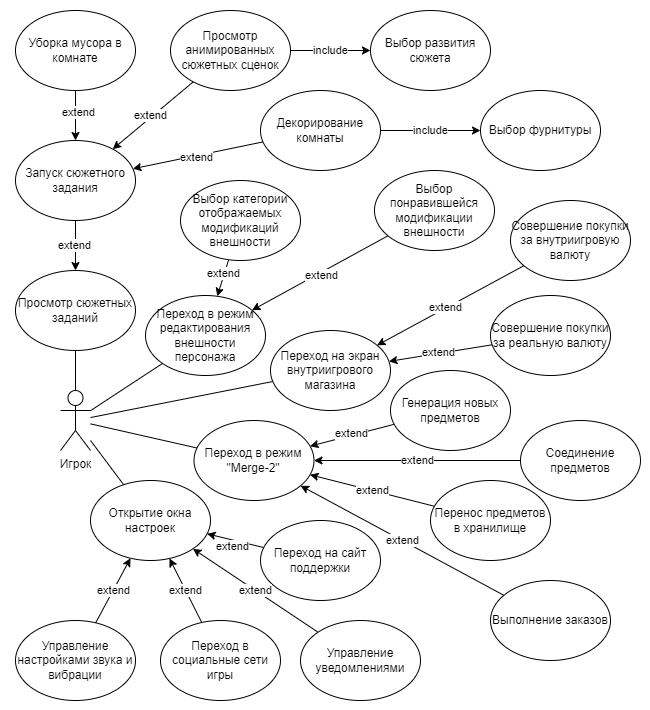
Для контроля выполнения требования технического задания необходимо провести испытания. Порядок и состав испытания определяются программой и методикой испытаний.

Приемка готового программного средства осуществляется комиссией по защите дипломного проекта в соответствии с программой и методикой испытаний.

# ПРИЛОЖЕНИЕ Б

(справочное)

**Диаграмма вариантов использования**



# ПРИЛОЖЕНИЕ Г

(обязательное)

**Программа и методика испытаний**

**Г.1 Объект испытаний**

Объектом испытаний является мобильная гиперказуальная игра «Destroyer».

**Г.2 Цель испытаний**

Проверка программного продукта, мобильной гиперказуальной игры «Destroyer», на предмет работоспособности в соответствии с техническим заданием и другое программной документации, а также отказоустойчивости ошибкам в процессе работы приложения.

**Г.3 Требования к программе**

Разработанный программный продукт должен включать в себя функциональный модуль пользователя, который отвечает за:

* настройку элементов игрового процесса;
* прохождение игрового уровня;
* зарабатывание игровой валюты;
* обмен валюты на помощников и прокачку;
* сохранение прогресса.

**Г.4 Требования к программной документации**

Дипломный проект должен содержать следующие элементы программное документации:

* техническое задание;
* описание программы;
* руководство пользователя.

**Г.5 Средства и порядок испытаний**

**Г.5.1 Средства для испытаний**

Для проведения испытания над мобильным приложением необходимо наличие персонального компьютера, с установленным на него игровым движком Unity C#. Для проверки корректности функционирования разработанного программного продукта должны применяться следующие программные средства:

* процессор с тактовой частотой 1440 МГц+;
* оперативная память 4 Гб+;
* наличие игрового движка Unity версии, на которой разрабатывался программный продукт;
* шнур для соединения мобильного устройства и персонального компьютера;
* разработанное Unity ПО для тестирования приложения на мобильном устройстве;
* монитор;
* клавиатура;
* мышь.

Также проводить тестирование можно непосредственно на самом мобильном устройстве. Для проверки корректности функционирования разработанного программного продукта должны применяться следующие программные средства:

* операционная система Android 7.0+ или iOS;
* оперативная память 1 Гб+;
* наличие свободного места на мобильном устройстве в размере 230 Мб+;

**Г.5.2 Порядок испытаний**

Проверка приложения выполнялась на мобильном устройстве с ОС Android в соответствии с тест-планом, представленным в таблицу Г.1.

Таблица Г.1 – Тест-план проверки мобильного приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовый вариант | Входные данные | Ожидаемый результат | Результат тестирования |
| Запуск приложения | Вход в мобильное приложение | Открытие основного игрового поля | Тест пройден успешно |
|  |  |  |  |
| Продолжение таблицы Г.1 | | | |
| Тестовый вариант | Входные данные | Ожидаемый результат | Результат тестирования |
| Управление игровым персонажем | Перемещение пальцем по экрану устройства или через эмулятор на персональном компьютере | Персонаж двигается вслед за указанием пользователя | Тест пройден успешно |
| Получение игровой валюты | Добытые кристаллы | Обмен кристаллов на игровую валюту | Тест успешно пройден |
| Найм помощников | Внутриигровая валюта | Покупка помощников за внутриигровую валюту | Тест успешно пройден |
| Переход на панель прокачки | Нажатие на кнопку «Улучшение» | Открытие панели прокачки персонажа | Тест успешно пройден |
| Прокачка персонажа | Внутриигровая валюта | Улучшение персонажа за внутриигровую валюту | Тест успешно пройден |
| Переход на панель настроек | Нажатие на кнопку «Настройки» | Открытие панели с настройками | Тест успешно пройден |
| Изменение настроек звука | Изменение параметра звука с помощью пальца | Изменение звука | Тест успешно пройден |
| Открытие панели смены уровня | Нажатие на кнопку «Карта» | Открытие панели смены уровня | Тест успешно пройден |
| Выбор нового игрового уровня | Нажатие на новый игровой уровень на карте | Переход на новый игровой уровень, запуск игрового процесса | Тест успешно пройден |
| Смерть персонажа | Смерть персонажа от AI | Возобновление игрового процесса на уровне с начала | Тест успешно пройден |
| Смерть помощника | Смерть помощника от AI | Исчезновение помощника | Тест успешно пройден |
| Получение помощников игровой валюты | Заработанные кристалла | Обмен кристаллов на валюту | Тест успешно пройден |

**Г.6 Методы испытаний**

Проверка программного продукта должна осуществляться методом «Черного ящика», который предназначен для проверки функционального назначения программы.

Корректность выполнения определяется сравнением фактического и ожидаемого результата. Если результаты совпали, то тестирование пройдено успешно. Обратные результаты будут говорить о проблемах и сбоях в работе программного продукта и необходимости устранения неполадок на определенных участка, соответственно. Исходя из этого можно будет сделать вывод, что тестирование программы не было пройдено успешно.