

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

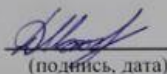
БАКАЛАВРСКАЯ РАБОТА

на тему

ОБРАЗОВАТЕЛЬНОЕ ПРИЛОЖЕНИЕ

«МАГИЯ ЗНАНИЙ»

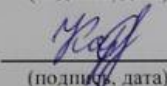
Студент


(подпись, дата)

31.05.2023

Д.Д. Макаров

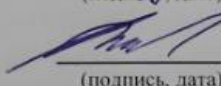
Руководитель


(подпись, дата)

31.05.2023

Н.С. Карамышева

Нормоконтролёр


(подпись, дата)

31.05.2023

А.С. Бычков

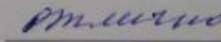
Работа допущена к защите (протокол зас. кафедры от 02.06.23 г. № 14)

Заведующий кафедрой

(подпись)


(фамилия, инициалы)

М.А. Митрохин

Работа защищена с отметкой  (протокол заседания ГЭК от

06.06.2023г.

№

2)

Секретарь ГЭК

(подпись)

Е.И. Гурин

(фамилия, инициалы)

Пенза 2023

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 26 » апреля 20 23 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

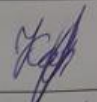
1. Студент Макаров Дмитрий Дмитриевич гр 19ВВП1 факультета вычислительной техники.
2. Тема работы Образовательное приложение «Магия знаний»
Тема утверждена приказом ПГУ № 458/о от "26" апреля 2023 г.
3. Руководитель бакалаврской работы к.т.н., доцент, каф. ВТ ПГУ Карамышева Н.С.
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение «Образовательное приложение «Магия знаний»» в жанре визуальной новеллы, рассказывающее об основных этапах и событиях истории России, реализовать систему диалогов. Разработать внутриигровую энциклопедию, сохраняющую информацию, полученную пользователем. Предусмотреть возможность выбора пользователем удобной для него платформы. В игровой форме донести информацию об истории страны. Провести тестирование и отладку каждого элемента приложения.

5. Перечень подлежащих разработке вопросов

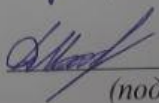
Требуется произвести анализ предметных областей: изучить существующие образовательные приложения, выделить их достоинства и недостатки. Выбрать язык программирования, который подойдет для написания логики приложения. Определить архитектуру приложения и вспомогательные средства разработки.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2023-30.04.2023	
2.	Анализ требований	10	01.05.2023-02.05.2023	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2023-07.05.2023	
4.	Описание алгоритмов	40	08.05.2023-11.05.2023	
5.	Разработка приложения «Образовательное приложение «Магия знаний»»	150	12.05.2023-29.05.2023	
6.	Оформление пояснительной записки	24	30.05.2023-31.05.2023	

Дата выдачи задания "26" апреля 2023 г.

Руководитель бакалаврской работы  26.04.23 Карамышева Н.С.
(подпись, дата)

Задание к исполнению принял студент  26.04.23 Макаров Д.Д.
(подпись, дата)

Работу к защите допустить
Декан факультета  26.05.2023 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 111 листов, 60 рисунков, 2 таблицы, 5 источников и 3 приложения.

ОБРАЗОВАНИЕ, ОБРАЗОВАТЕЛЬНЫЕ ПРИЛОЖЕНИЯ, ИСТОРИЯ, ВИЗУАЛЬНАЯ НОВЕЛЛА, *UNITY, C#, SOLID, SINGLETON, FUNGUS*.

Цель работы – разработать образовательное приложение.

Объектом разработки является образовательное приложение в жанре визуальной новеллы, рассказывающее об истории России.

В результате проделанной работы было разработано приложение в жанре визуальной новеллы, рассказывающее об основных этапах и событиях истории России, реализована система диалогов. Разработана внутриигровая энциклопедия, сохраняющую информацию, полученную пользователем. Предусмотрена возможность выбора пользователем удобной для него платформы. В игровой форме донесена информация об истории страны. Проведено тестирование и отладка каждого элемента приложения.

					ПГУ1.090301.13.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Макаров Д.Д.			Образовательное приложение «Магия знаний» Пояснительная записка	Лит.	Лист	Листов
Провер.		Карамышева Н.С.					2	
Н. Контр		Бычков А.С.				ФВТ, каф. ВТ гр. 19ВВП1		
Утверд		Митрохин М.А.						

Содержание

Перечень сокращений и обозначений	7
Введение	8
1 Образовательное приложение «Магия знаний»	9
1.1 Анализ существующих аналогов	9
1.1.1 «История России, даты»	9
1.1.2 «История России и Мира. ЕГЭ»	13
1.1.3 «История для всех! Древний мир»	16
1.1.4 «ЕГЭ История 2023»	20
1.2 Постановка задачи	22
1.3 Выбор игрового движка	23
1.3.1 <i>Unreal Engine</i>	23
1.3.2 <i>Unity</i>	24
1.3.3 <i>CryEngine</i>	25
1.3.4 <i>Godot</i>	26
1.4 Выбор языка программирования	27
1.4.1 <i>UnityScript</i>	27
1.4.2 <i>Boo</i>	28
1.4.3 <i>C#</i>	29
1.5 Выводы	30
1.6 Выбор инструментов разработки	30
1.6.1 Выбор среды разработки	30
1.7 Архитектура приложения	33
1.7.1 Принципы <i>SOLID</i>	34
1.7.2 Паттерн <i>Singleton</i>	34
1.7.3 Паттерн <i>Publish-Subscribe</i>	36
1.8 Разработка приложения	37
1.8.1 Разработка системы диалогов	39

1.8.2 Разработка внутриигровой энциклопедии	47
1.8.2.1 Класс <i>EncyclopediaManager</i>	49
1.8.2.2 Класс <i>EncyclopediaUI</i>	50
1.8.2.3 Класс <i>EncyclopediaArticle</i>	51
1.9 Тестирование и отладка	51
1.10 Описание приложения	54
1.10.1 Руководство пользователя	54
2 Охрана труда	61
2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения	61
2.1.1 Оборудование рабочего места	62
2.1.2 Освещенность рабочего места	64
2.1.3 Электробезопасность	65
2.1.4 Пожарная безопасность	65
2.1.5 Режим труда и отдыха	66
Заключение.....	68
Список используемых источников	69
Приложение А – <i>UML</i> – диаграммы	70
Приложение Б – Листинг программы	75
Приложение В – Презентация	95

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

Акроним – разновидность аббревиатуры, которая образуется по первым буквам слов, входящих в зашифрованное понятие

ГОСТ – региональный стандарт, принятый Межгосударственным советом по стандартизации, метрологии и сертификации Содружества Независимых Государств

ООП – объектно-ориентированное программирование

Паттерн проектирования – описание типичного способа решения проблем при проектировании программного обеспечения

ПК – персональный компьютер

ПЭВМ – персональная электронно-вычислительная машина

AR (*Augmented Reality*) – дополненная реальность

Framework – программная платформа, которая упрощает разработку программного продукта, определяет структуру проекта

SOLID (*Single responsibility Open–closed Liskov substitution Interface Segregation Dependency inversion*) - 5 основных принципов объектно-ориентированных проектирования и программирования

UI (*User Interface*) – пользовательский интерфейс

UML (*Unified Modeling Language*) – унифицированный язык моделирования

VR (*Virtual Reality*) – виртуальная реальность

.NET – кроссплатформенная платформа для создания различных типов приложений

					ПГУ1.090301.13.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В настоящее время разработка компьютерных игр является одной из самых быстроразвивающихся и перспективных отраслей техники. В связи с этим, все больше людей становятся заинтересованными в создании своих собственных игр, используя современные технологии и программное обеспечение.

Например, отчет от *App Annie* за 2020 год говорит, что время, проводимое пользователями в мобильных играх, увеличилось на четверть по сравнению с предыдущим годом.

Исследование от *PwC* 2021 года показывает, что глобальный рынок видеоигр увеличился на 9,3% в 2020 году.

Благодаря различным игровым механикам и интерактивности, обучение в играх становится более эффективным и запоминающимся. Кроме того, игры могут увлечь даже тех, кто не любит учиться, так как они делают процесс обучения более интересным и занимательным.

Целью выпускной работы является разработка образовательного игрового приложения в жанре визуальной новеллы, в котором будут представлены основные события истории России с древнейших времен по настоящее время.

					ПГУ1.090301.13.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

1 Образовательное приложение «Магия знаний»

За последние несколько десятков лет образование в школах не сильно изменилось и состоит из чтения учебников и запоминания информации оттуда, но технологии не стоят на месте и цифровизация проникает в различные сферы общества и образование не исключение. Для школьников это стали электронные дневники, электронное расписание и т.д. Следующий шаг - это изменение самого подхода к обучению, его цифровизация и геймификация.

Классический подход можно и нужно совмещать с более современным. Помимо прочтения учебников создавать интерактивные обучающие приложения.

Данная тема актуальна благодаря своему сочетанию игры и обучения. В ходе игры пользователям предстоит решать различные задачи, проходить тесты и отвечать на вопросы, которые помогут преодолеть препятствия.

В отличие от многих других приложений, "Магия знаний" предлагает обширный и интерактивный рассказ об истории России, что позволяет игрокам лучше понять и увлечься изучением предмета. Более того, в отличие от других образовательных приложений, которые представляют собой справочник или набор билетов для экзаменов, "Магия знаний" создает привлекательную игровую среду для изучения истории России, что может привлечь и мотивировать школьников к более глубокому изучению данной темы.

1.1 Анализ существующих аналогов

Среди аналогов образовательных исторических приложений можно выделить:

- «История России, даты»;
- «История России и Мира. ЕГЭ»;
- «История для всех! Древний мир»;
- «ЕГЭ История 2023».

1.1.1 «История России, даты»

Пользовательский графический интерфейс представлен на рисунках 1, 2, 3, 4.

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

Ссылка на скачивание - <https://apps.rustore.ru/app/studycards.school.history>.

Проанализировав описание, прикрепленное на странице приложения в *Rustore*, можно сделать вывод, что это приложение предоставляет возможность изучать школьный курс истории в виде флеш-карточек. Приложение имеет три режима: свободный режим, режим прохождения и режим конспекта.

В свободном режиме пользователь может свободно перелистывать карточки, изучая материал в удобном для себя порядке.

В режиме прохождения пользователю будет предложено лёгкое задание после каждой карточки, чтобы закрепить полученные знания. Этот режим является более структурированным, чем свободный режим.

В режиме конспекта весь материал будет представлен в виде лаконичного конспекта, который может быть полезен для повторения изученного материала.

Общая идея состоит в том, чтобы предоставить ученикам удобный и простой способ изучения большого объёма информации, представленной в виде флеш-карточек, которые являются эффективным методом запоминания информации. Режимы приложения позволяют пользователям настроить его под свои потребности, в зависимости от их уровня знаний и предпочтений в изучении материала.

					ПГУ1.090301.13.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		



Рисунок 1 – Логотип приложения в *Rustore*

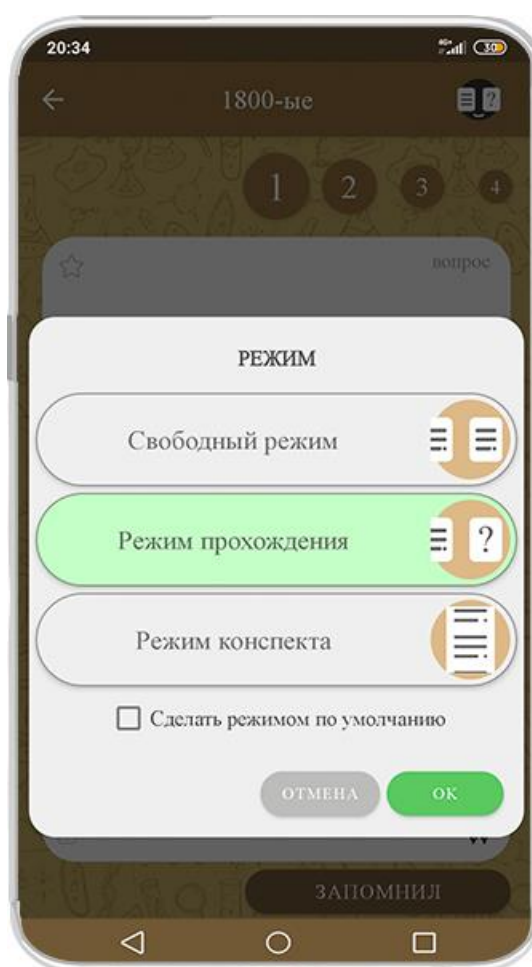


Рисунок 2 – Выбор режима



Рисунок 3 – Сборник дат



Рисунок 4 – Меню выбора эпохи для изучения

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

Используя приложение в течении нескольких дней можно выделить следующие плюсы:

- возможность добавлять эпохи в избранное, чтобы позже быстро вернуться к ним, не просматривая все остальные;
- разделение информации по карточкам является эффективным методом запоминания информации, особенно больших объёмов, так как они обеспечивают активное повторение материала и развитие визуальной памяти;
- каждый из режимов может быть полезен в изучении предмета. Режим прохождения помогает закреплять знания лёгкими заданиями после каждой карточки, что способствует более глубокому запоминанию материала. Режим конспекта позволяет получить краткое и лаконичное представление о материале, что может быть полезно для повторения уже изученного материала;
- мобильное приложение доступно в любое время в любом месте;
- приятное визуальное помогает улучшить восприятие информации и позволяет лучше сконцентрироваться на изучении материала. Использование ярких и приятных цветов, анимаций и графических элементов может сделать приложение более привлекательным и интересным для пользователей.

Из минусов можно выделить следующее:

- отсутствие интерактивности, обучение предусматривает обычное чтение и запоминание текстов, однако, для школьников начальных классов, отсутствие интересного повествования может вызвать у них скуку и неудовлетворенность процессом обучения;
- отсутствие кроссплатформенности. Данное приложение представлено только для мобильных устройств, что ограничивает пользователя, отсутствие выбора альтернативных платформ может оставить негативное впечатление.

1.1.2 «История России и Мира. ЕГЭ»

Графический интерфейс представлен на рисунках 5, 6, 7, 8.

Ссылка на скачивание - <https://apps.rustore.ru/app/com.nagimov.din.dirhisfree>.

Проанализировав описание, прикрепленное на странице приложения в

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

Rustore, можно сделать вывод, что приложение охватывает историю России, начиная от правителей и их политики до важных научных открытий, а также предлагает исторический словарь основных понятий и терминов.

История распределена на темы и статьи, каждая тема содержит даты важных событий.

Приведены статьи не только по истории России, но и мира.

Пользуясь данным приложением на протяжении нескольких часов удалось выявить несколько ошибок, связанных с отображением контента.



Рисунок 5 – Логотип приложения в *Rustore*



Рисунок 6 – Хронология войны 1812 г.

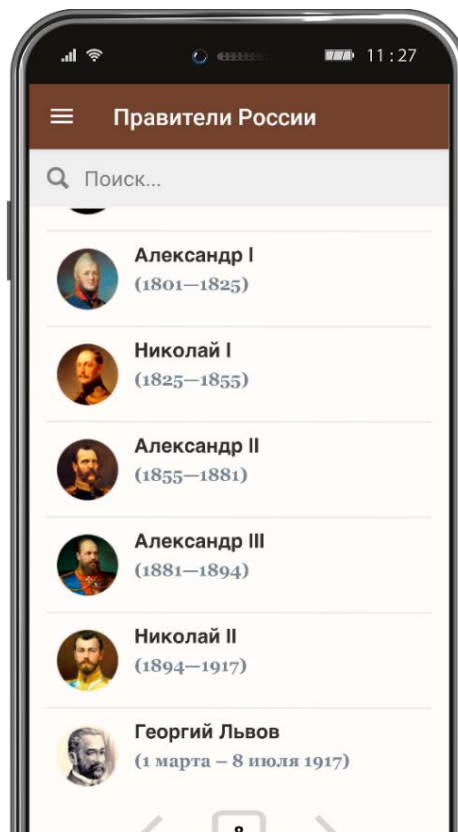


Рисунок 7 – Правители России по порядку правления

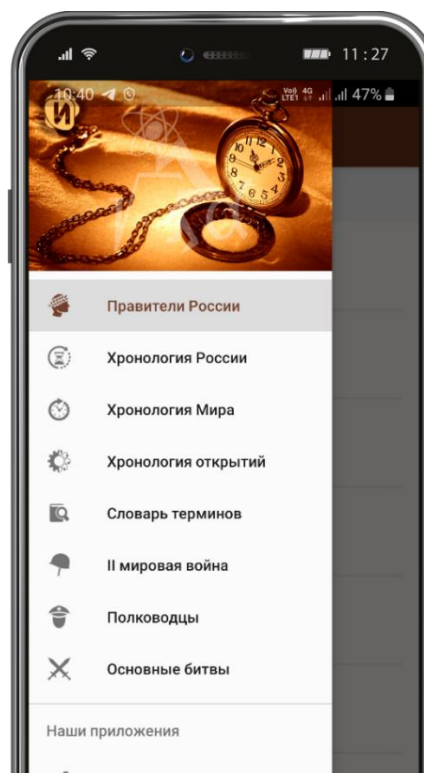


Рисунок 8 – Главное меню приложения

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		15

Из плюсов данного приложения можно выделить следующее:

- понятный графический интерфейс позволяет быстро найти нужную информацию в конкретном периоде истории, что существенно упрощает процесс обучения и снижает время, затрачиваемое на поиск нужных сведений;
- небольшой размер приложения составляет всего 9,5 МБ, оно может быть запущено даже на самых слабых мобильных устройствах;
- освещены события не только России, но и мира, это позволяет пользователям приложения получить более широкий обзор истории мировой цивилизации и узнать о том, какие важные события происходили за пределами своей страны;
- мобильное приложение доступно в любое время и месте.

Из минусов можно выделить следующее:

- отсутствие интерактивности, обучение предусматривает обычное чтение и запоминание текстов, однако, для школьников начальных классов, отсутствие интересного повествования может вызвать у них скуку и неудовлетворенность процессом обучения;
- отсутствие кроссплатформенности. Данное приложение представлено только для мобильных устройств, что ограничивает пользователя, отсутствие выбора альтернативных платформ может оставить негативное впечатление;
- отсутствие возможности сохранять статьи в избранное, для быстрого доступа к выбранным статьям;
- жалобы пользователей на ошибки, связанные с оплатой Premium доступа.

1.1.3 «История для всех! Древний мир»

Графический интерфейс представлен на рисунках 9, 10, 11, 12, 13.

Ссылка на скачивание - <https://apps.rustore.ru/app/ru.vda.ancienthistory>.

Описание приложения "История для всех! Древний мир" в Rustore говорит о том, что оно предназначено для погружения пользователей в античную историю и описывает исторические события и цитаты из Древней Персии, Древней Греции, Древнего Рима и Древнего Китая.

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16

Игра содержит пять режимов, включая эрудит, режим правда или ложь, кроссворды, найди слово и угадай цитату.

Пользователи могут выбирать наиболее интересный для них режим и использовать свои знания, логику и интуицию для ответа на вопросы и открытия новой информации.



Рисунок 9 – логотип приложения в Rustore



Рисунок 10 – Режим кроссворда



Рисунок 11 – Режим поиска слов

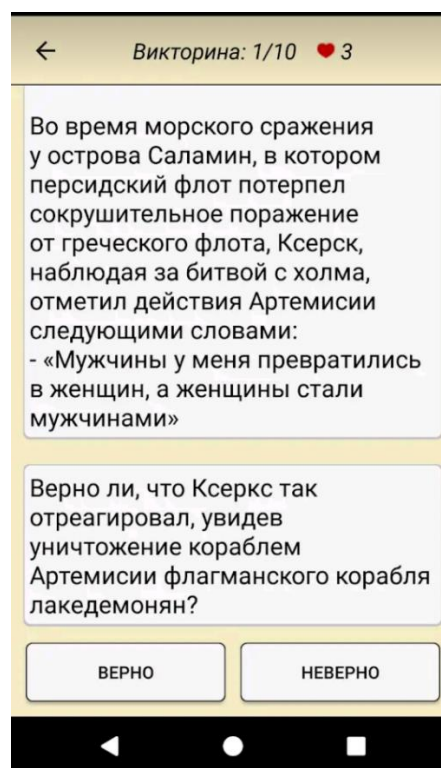


Рисунок 12 – Режим викторины



Рисунок 13 – Меню выбора эпохи

Из плюсов можно выделить, что приложение предлагает большое разнообразие режимов игры, каждый из которых проверяет различные навыки пользователя, и позволяет игроку выбирать наиболее интересный и подходящий режим для него, повышая мотивацию продолжать играть.

Данное приложение имеет следующие минусы:

- скудный интерфейс, данное приложение выглядит устаревшим;
- сосредоточенность только на античности, приложение не покрывает другие периоды истории;
- поверхностное содержание не обеспечивает достаточно глубокого;
- исследования исторических периодов и событий, а только описывать их в общих чертах;
- технические ошибки.

1.1.4 «ЕГЭ История 2023»

Графический интерфейс представлен на рисунках 14, 15, 16, 17.

Ссылка на скачивание - <https://apps.rustore.ru/app/istoryja.rossii.ege>.

Изучив описание приложения в *Rustore* можно понять, что приложение строго ориентировано на подготовку к ЕГЭ по истории. Материал готовился совместно с репетиторами и профессорами.

В игре присутствует множество режимов, направленных на различные способы изучения истории.

- тесты;
- блиц тест;
- даты;
- теория;
- таблицы;
- Великая Отечественная Война;
- архитектура и скульптура;
- личности;
- карты;
- исторические периоды;
- калькулятор баллов ЕГЭ.

Присутствует платная подписка, дающая доступ к платному чату.



Рисунок 14 – Логотип в *Rustore*

					ПГУ1.090301.13.001 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

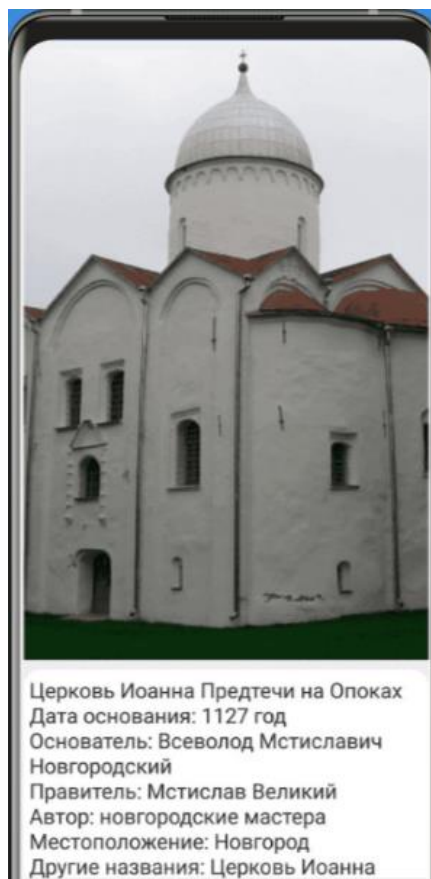


Рисунок 15 – Историческая справка

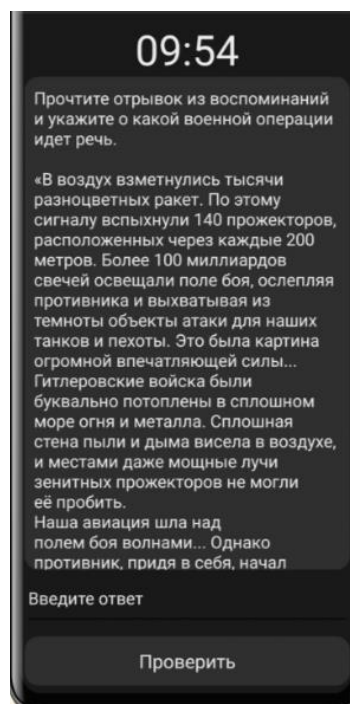


Рисунок 16 – Режим тестирования на время

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		21

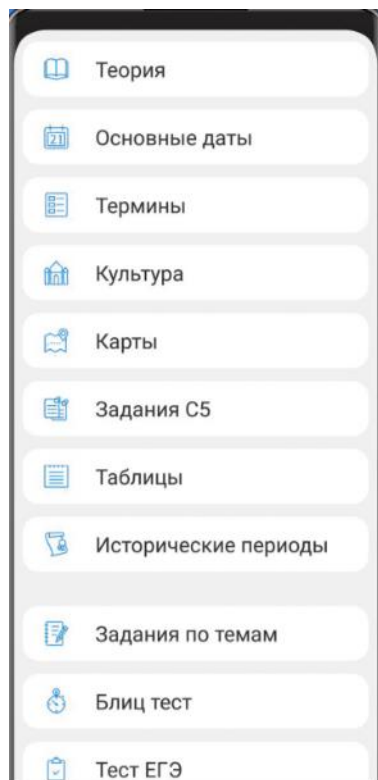


Рисунок 17 – Выбор режима подготовки

1.2 Постановка задачи

Требуется разработать приложение, которое будет соответствовать следующим целям:

- рассказать в доступной форме об основных этапах и событиях истории России;
- повествование должно вестись посредством диалогов между историческими личностями, для погружения игрока в процесс игры;
- сохранение полученных знаний в игре, для последующего обращения к ним;
- интуитивно понятный интерфейс для работы с приложением;
- расположение элементов пользовательского интерфейса должно позволять пользователю быстро ориентироваться в приложении;
- игрок должен иметь возможность выбирать удобную для него платформу;
- необходимо провести масштабную отладку и тестирование каждого

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

элемента в целях устранения ошибок и недочетов.

В процессе разработки необходимо решить следующие задачи:

- сочетать в себе элементы игры и обучения, чтобы повысить интерес и мотивацию игроков;
- предусмотреть автоматическое сохранение прогресса игрока;
- создать энциклопедию знаний, в которую будут записываться основная полученная информация, полученная игроком;
- создание версий игры для игры на основных платформах;
- создание системы диалогов;
- создание версий для мобильных устройств и устройств под управлением *Windows, Mac, IOS и Android*;
- корректная работа на каждой из перечисленных платформ.

1.3 Выбор игрового движка

В современном мире существует множество различных инструментов для создания игр - от простых библиотек до гигантских платформ с множеством функций и возможностей. Чтобы создать игру на высоком уровне, необходимо выбрать качественные инструменты. Поэтому для разработки игр лучше всего использовать уже готовый движок, который предоставляет возможность создавать игры без лишних трудностей и затрат времени. Ведь это является ключом к созданию интересных игр, которые будут пользоваться популярностью среди игроков.

1.3.1 *Unreal Engine*

Unreal Engine - это игровой движок, разработанный компанией *Epic Games* на языке программирования *C++*. Он был представлен в 1998 году как основа для первого шутера от первого лица *Unreal*, и в настоящее время используется для создания игр различных жанров.

Основные особенности движка *Unreal Engine*:

- графика высокого качества. *Unreal Engine* обладает мощными

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

возможностями по созданию визуальных эффектов, реалистичного освещения и динамической обработки изображений. Это позволяет создавать удивительные графические эффекты и улучшать общую качественную составляющую игры;

- большое сообщество и широкий инструментарий. *Unreal Engine* имеет огромное сообщество разработчиков, которые создают инструменты, плагины и дополнения для улучшения движка. Это позволяет разработчикам быстро и легко находить решения для любых задач;

- *blueprints* - система визуального программирования. *Unreal Engine* имеет систему *blueprints*, которая позволяет разработчикам создавать игровые объекты и логику без необходимости писать код. Это упрощает процесс разработки и позволяет не программистам быстрее создавать прототипы и реализовывать свои идеи;

- открытый исходный код. *Unreal Engine* имеет открытый исходный код, что позволяет разработчикам изменять и улучшать движок под свои нужды. Это означает, что любой желающий может получить доступ к исходному коду и использовать его для своих проектов.

1.3.2 Unity

Unity - это еще один популярный игровой движок, используемый для создания игр на различных платформах, включая ПК, мобильные устройства, игровые консоли и виртуальную реальность.

Основные особенности движка *Unity*:

- широкий выбор языков программирования. *Unity* позволяет использовать различные языки программирования, включая *C#*, *JavaScript(UnityScript)* и *Boo*. Это дает разработчикам возможность выбрать наиболее подходящий язык для их задач;

- поддержка *VR* и *AR*. *Unity* имеет широкую поддержку виртуальной и дополненной реальности, что позволяет разработчикам создавать игры и приложения для этих платформ;

- интеграция с другими инструментами. *Unity* имеет возможность интеграции

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		24

с другими инструментами разработки, включая *Visual Studio*, *Xcode* и другие. Это упрощает процесс разработки и позволяет использовать уже имеющиеся инструменты.

Среда имеет *Drag&Drop* - интерфейс, который состоит из различных окон, которые очень просто настраиваются по высоте, ширине и размерам.

Игровой движок *Unity3D* полностью интегрирован в среду разработки *Unity*. Тесная интеграция позволяет прямо в редакторе увидеть все то, на что будет способна выпущенная игра.

Так же в *Unity* имеется возможность сборки проекта для всех актуальных платформ, что сразу решает одну из поставленных задач.

Unity имеет большое количество ресурсов, включая учебные материалы, видеоуроки, документацию и форумы сообщества. Это позволяет разработчикам быстро и легко находить ответы на свои вопросы и учиться новым навыкам.

1.3.3 *CryEngine*

CryEngine - это игровой движок, разработанный компанией *Crytek* на языках *C++* и *Lua*. Используется для создания игр с высококачественной графикой и физикой, таких как шутеры от первого лица. Он также поддерживает виртуальную реальность и имеет встроенные инструменты для создания игровых объектов, анимации и эффектов.

Основные особенности движка *Cry Engine*:

- графика. *Cry Engine* имеет высокоэффективный движок рендеринга, который обеспечивает превосходную графику и позволяет создавать качественные визуальные эффекты. Он использует технологию *Global Illumination*, которая создает более реалистичные и детализированные сцены;

- ИИ. *Cry Engine* предоставляет набор инструментов для создания умных и реалистичных противников. Использование нейронных сетей позволяет создавать более сложные и разнообразные поведенческие сценарии;

- физика. *Cry Engine* имеет мощную физическую систему, которая позволяет создавать реалистичные эффекты физики, такие как взрывы, разрушения и т.д.

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		25

1.3.4 Godot

Godot - это бесплатный, открытый исходный код движок для разработки 2D и 3D игр. Он был создан в 2007 году, а затем был открыт для общественности в 2014 году. *Godot* поддерживает множество языков программирования, включая C++, C#, GDScript и Visual Scripting.

Однако *Godot* все еще является новым движком и не имеет такой широкой поддержки сообщества и рынка, как у движков *Unreal* и *Unity*. Кроме того, не все инструменты и функции могут быть доступны сразу при установке, как это происходит в более популярных движках. Тем не менее, *Godot* остается популярным выбором для начинающих и опытных разработчиков, благодаря своей гибкости и бесплатности.

В таблице 1.1 приведено сравнение всех выше описанных платформ для разработки игр.

Таблица 1.1 – Сравнение игровых движков

Движок	Достоинства	Недостатки
<i>Unreal Engine</i>	Кроссплатформенность, продвинутый инструментарий, графика высокого качества, крупное сообщество разработчиков.	Высокие требования к аппаратной части, сложность освоения, затратность разработки.
<i>Unity</i>	Кроссплатформенность, продвинутый инструментарий, более доступный графический интерфейс, относительно легок в освоении.	Проблемы с производительностью, сложность настройки параметров среды, закрытый исходный код.

<i>Cry Engine</i>	Кроссплатформенность, графика высокого уровня, готовые шаблоны.	Высокие требования к аппаратной части, сложен в освоении, необходимость лицензии.
<i>Godot</i>	Полностью бесплатен, кроссплатформенность, широкий выбор языков программирования.	Небольшая популярность, не такой мощный как другие движки.

Проанализировав плюсы и минусы каждого игрового движка, можно прийти к выводу что оптимальным выбором станет *Unity*.

1.4 Выбор языка программирования

В *Unity* можно писать на нескольких языках программирования, таких как *C#*, *UnityScript* (базируется на языке *JavaScript*) и *Boo* (базируется на языке *Python*).

1.4.1 *UnityScript*

UnityScript (также известный как *JavaScript for Unity*) - это язык программирования, который можно использовать для разработки игр в *Unity*. В отличие от *JavaScript*, который используется в веб-разработке, *UnityScript* был создан специально для работы с игровым движком *Unity*.

Особенности *UnityScript*:

- легкость изучения. *UnityScript* основан на языке программирования *JavaScript*, который является одним из самых популярных языков веб-разработки. Это делает его легче изучать для программистов, уже знакомых с *JavaScript*;
- интеграция с *Unity*. *UnityScript* был разработан специально для работы с игровым движком *Unity*. Это означает, что он легко интегрируется с остальными компонентами *Unity*, такими как визуальный редактор, компоненты игровых объектов и т.д;

- динамическая типизация. *UnityScript* - это язык с динамической типизацией, что означает, что типы данных могут быть изменены во время выполнения программы. Это позволяет более гибко управлять данными в игре;

- сборщик мусора. *UnityScript* использует сборщик мусора, что облегчает работу с памятью и уменьшает количество ошибок, связанных с управлением памятью;

- поддержка ООП. *UnityScript* поддерживает объектно-ориентированное программирование, что делает его более удобным для создания больших проектов.

Однако, стоит отметить, что *UnityScript* является устаревшим языком программирования и не подойдет для создания современных игровых приложений.

1.4.2 Boo

Boo - это еще один язык программирования, поддерживаемый в *Unity*. Он был разработан Родни Лопезом в 2003 году, чтобы облегчить создание игр и приложений на основе *.NET Framework*.

Особенности *Boo*:

- объектно-ориентированная парадигма. *Boo* имеет синтаксис, который ориентирован на объектно-ориентированную парадигму программирования. Он предлагает классы, наследование, интерфейсы и другие объектно-ориентированные конструкции;

- динамическая типизация. *Boo* поддерживает динамическую типизацию, что означает, что типы данных могут быть определены во время выполнения программы, а не во время компиляции. Это позволяет разработчикам писать код более гибко и быстро;

- простой и лаконичный синтаксис. *Boo* имеет простой и лаконичный синтаксис, который позволяет разработчикам быстро создавать код. Он также имеет поддержку интегрированной документации, что позволяет разработчикам легко понимать, как использовать различные функции и классы.

Однако, из-за отсутствия поддержки и отсутствия обновлений, не рекомендуется использовать *Boo* для новых проектов или в проектах, которые

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		28

должны быть поддерживаемы в долгосрочной перспективе.

1.4.3 C#

C# является одним из основных языков программирования для создания игр в *Unity*. Рассмотрим его особенности:

- строгая типизация. C# является языком со строгой типизацией, что означает, что каждая переменная и объект имеет определенный тип данных. Это повышает надежность кода и позволяет избежать многих ошибок;

- большая поддержка в *Unity*. C# является одним из языков, которые наиболее активно поддерживаются в *Unity*. Он имеет большое сообщество разработчиков, а также множество инструментов и плагинов, которые делают его удобным для использования;

- высокая производительность. C# обладает высокой производительностью благодаря многим оптимизациям, которые были внедрены в язык. Это делает его хорошим выбором для создания игр с высокими требованиями к производительности;

- множество возможностей. C# имеет множество возможностей, которые делают его удобным для создания игр различных жанров. Он поддерживает объектно-ориентированное программирование, а также функциональное программирование;

- широкая поддержка сторонних библиотек. C# имеет широкую поддержку сторонних библиотек, что позволяет разработчикам использовать уже существующие инструменты и решения для своих проектов.

C# является мощным и универсальным языком программирования, который подходит для создания игр различных жанров и сложности. Он имеет множество инструментов и библиотек, которые делают его удобным для использования в *Unity* [1].

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		29

1.5 Выводы

Каждый язык программирования имеет свои преимущества и недостатки, и выбор языка зависит от конкретных задач, которые необходимо решить. Однако, в контексте разработки игр в *Unity*, наиболее предпочтительным языком является *C#*.

Unity обеспечивает полную поддержку языка *C#*, и большинство функций и компонентов движка *Unity* доступны для использования на *C#*. Также существует большое сообщество разработчиков, которые активно делятся своими знаниями и создают инструменты для упрощения работы в *Unity* с использованием *C#*.

Кроме того, *C#* имеет множество преимуществ перед другими языками программирования, используемыми в *Unity*. Например, в отличие от *UnityScript* и *Boo*, *C#* является полностью типизированным языком, что позволяет обнаруживать ошибки в коде на этапе компиляции, а не только во время выполнения. Также *C#* имеет высокую скорость работы и хорошую оптимизацию кода [2].

В целом, выбор языка программирования зависит от определенных факторов, но в контексте разработки игр в *Unity*, *C#* является предпочтительным языком благодаря его функциональности, мощности и поддержке сообщества.

1.6 Выбор инструментов разработки

1.6.1 Выбор среды разработки

Существует множество интегрированных сред разработки (*IDE*) для программирования на *C#*. Некоторые из наиболее популярных *IDE*, которые поддерживают *C#*, это *Microsoft Visual Studio*, *JetBrains Rider*, *SharpDevelop*.

Visual Studio - это интегрированная среда разработки (*IDE*) от *Microsoft*, предназначенная для разработки программного обеспечения под *Windows*, *Android*, *iOS* и многие другие платформы.

Первая версия *Visual Studio* была выпущена в 1997 году. Она предназначалась для разработки приложений под операционную систему *Windows*

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		30

NT 4.0 и предлагала средства для работы с языками программирования, такими как *Visual Basic*, *C++* и *Visual J++*.

С тех пор *Visual Studio* получила множество обновлений и значительно расширила свой функционал. В 2002 году была выпущена версия *Visual Studio .NET*, которая включала в себя поддержку языка *C#* и платформы *.NET Framework*. В 2005 году была выпущена *Visual Studio 2005*, которая добавила поддержку разработки веб-приложений и инструменты для разработки мобильных приложений.

Сегодня *Visual Studio* является одной из наиболее популярных *IDE* для разработки программного обеспечения, которая поддерживает большое количество языков программирования, включая *C++*, *C#*, *F#*, *Visual Basic*, *JavaScript*, *TypeScript*, *Python* и другие.

К плюсам *Visual Studio* можно отнести:

- поддержка *Unity*;
- существует бесплатная *Community* версия;
- богатая функциональность и множество инструментов для разработки, отладки и тестирования кода;
- высокая степень интеграции с другими продуктами *Microsoft*, такими как *Azure* и *Git*;
- поддержка широкого спектра языков программирования.

Минусы:

- требует значительных ресурсов для запуска и использования, что может быть проблемой для менее мощных компьютеров;
- платная версия может быть дорогой для некоторых пользователей.

JetBrains Rider - это *IDE*, разработанная компанией *JetBrains*. Предназначена для работы с языками *C#*, *VB.NET*, *F#* и другими, запускается на платформе *.NET Framework* и предназначена для операционных систем *Windows*, *macOS* и *Linux*.

История разработки *Rider* началась в 2016 году, когда компания *JetBrains* объявила о запуске новой *IDE* для платформы *.NET*. Разработка проекта шла более

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		31

двух лет и была основана на технологиях *ReSharper* и *IntelliJ IDEA*. *JetBrains* представила первую версию *Rider* в январе 2017 года.

С момента выхода первой версии *JetBrains Rider* постоянно работает над улучшением и добавлением новых функций в *IDE*. На сегодняшний день *Rider* предоставляет разработчикам широкий спектр инструментов и функций, включая поддержку *Unity*, интеграцию с *.NET Core* и *ASP.NET*, систему автодополнения кода, функции рефакторинга, улучшенную навигацию по коду и многое другое.

Одним из ключевых преимуществ *JetBrains Rider* является наличие инструмента *ReSharper*, который позволяет увеличить производительность разработки и улучшить качество кода. Кроме того, *JetBrains Rider* имеет отличную поддержку для *.NET Core*, что делает его привлекательным инструментом для разработки кроссплатформенных приложений.

К плюсам *Rider* можно отнести:

- поддержка *Unity*;
- отлично подходит для разработки на *C#* и других языках программирования;
- удобный и продвинутый интерфейс с поддержкой многих функций;
- поддержка платформ *Linux* и *macOS*, что делает его доступным для разработчиков, работающих в этих средах;
- интеграция с *Git* и другими инструментами.

Минусы *Rider*:

- цена может быть высокой для некоторых пользователей;
- ресурсоемкость может быть не подходящей для менее мощных ПК;
- отсутствие бесплатной версии.

SharpDevelop - это свободно распространяемая интегрированная среда разработки (*IDE*) для языка программирования *C#* и других языков *.NET*. Она была разработана в 2000 году группой немецких программистов из компании *ICSharpCode* и изначально называлась "*#develop*". Позже она была переименована в *SharpDevelop*.

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		32

SharpDevelop была создана как бесплатная альтернатива коммерческой *IDE* *Microsoft Visual Studio*. Она была разработана на языке программирования *C#* с использованием *.NET Framework*. Однако, в отличие от *Visual Studio*, *SharpDevelop* была совместима с операционными системами, отличными от *Windows*, такими как *Linux* и *macOS*.

Плюс *SharpDevelop* состоит в том, что данная *IDE* распространяется абсолютно бесплатно и ее исходный код является открытым.

Минусы:

- не имеет такой же функциональности и богатства инструментов, как *Visual Studio* и *JetBrains Rider*;
- поддержка может быть ограниченной.

1.7 Архитектура приложения

В начале проектирования была составлена диаграмма развертывания для визуализации взаимодействия различных устройств на физическом уровне, она представлена на рисунке 18.

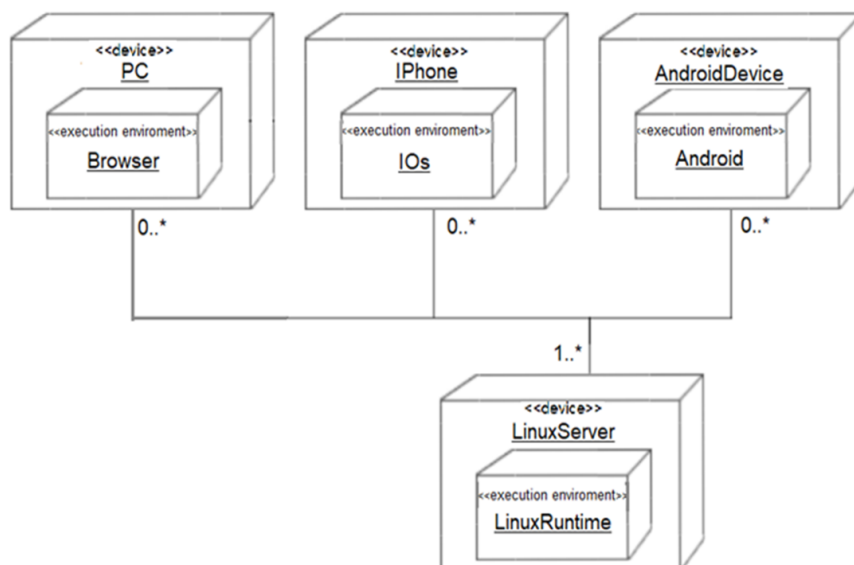


Рисунок 18 – *UML* – диаграмма развертывания

В данном приложении применяются несколько паттернов проектирования и

принципы *SOLID*. Они способствуют эффективной организации кода и управлению состоянием приложения. Некоторые из применяемых паттернов это *Singleton*, *publish-subscribe pattern*. Помимо этого, применение принципов *SOLID* обеспечивает гибкость и легкость поддержки кода.

1.7.1 Принципы *SOLID*

Принципы *SOLID* - это набор принципов объектно-ориентированного программирования, которые помогают разработчикам создавать гибкие, расширяемые и поддерживаемые системы. Акроним *SOLID* означает следующие принципы:

- принцип единственной ответственности (*Single Responsibility Principle, SRP*): Каждый класс должен иметь только одну причину для изменения. Он должен быть ответственным только за одну часть функциональности;

- принцип открытости/закрытости (*Open/Closed Principle, OCP*): Программные сущности должны быть открыты для расширения, но закрыты для изменения. Это достигается путем использования абстракций и интерфейсов;

- принцип подстановки Лисков (*Liskov Substitution Principle, LSP*): Объекты в программе должны быть заменяемыми своими наследниками без нарушения корректности программы. Наследуемые классы должны сохранять поведение базового класса;

- принцип разделения интерфейса (*Interface Segregation Principle, ISP*): Клиенты не должны зависеть от интерфейсов, которые они не используют. Интерфейсы должны быть специфичными и небольшими, соответствующими потребностям клиентов;

- принцип инверсии зависимостей (*Dependency Inversion Principle, DIP*): Модули верхнего уровня не должны зависеть от модулей нижнего уровня. Оба должны зависеть от абстракций. Абстракции не должны зависеть от деталей, а детали - от абстракций.

1.7.2 Паттерн *Singleton*

Паттерн *Singleton*, схема которого изображена на рисунке 19, относится к

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		34

классу порождающих паттернов (*creational patterns*) и используется для ограничения создания только одного экземпляра класса и предоставления глобальной точки доступа к этому экземпляру.

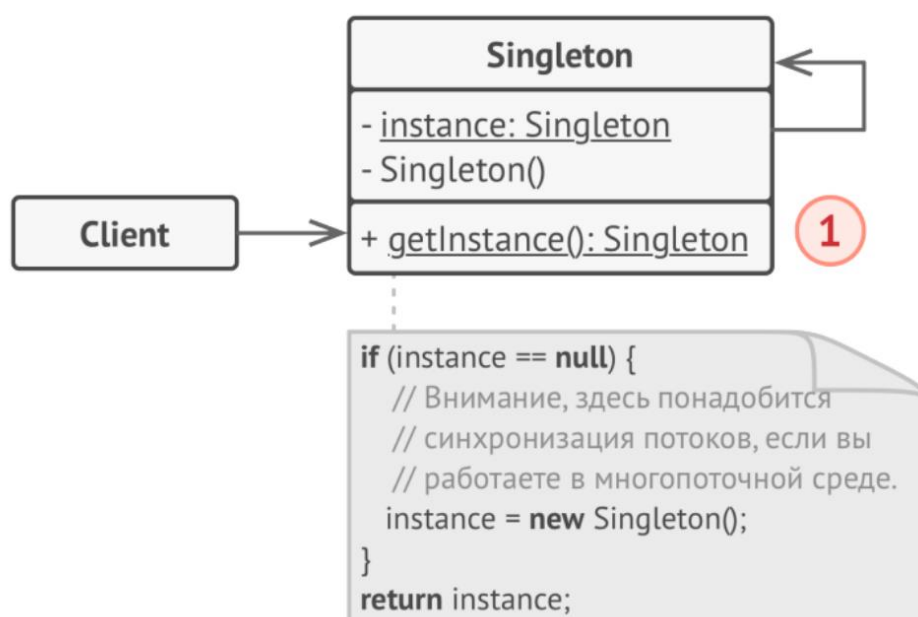


Рисунок 19 – Схематичное изображение паттерна *Singleton*

Основная идея паттерна *Singleton* заключается в том, чтобы класс имел приватный конструктор, чтобы предотвратить создание экземпляров класса извне. Вместо этого класс сам создает и хранит единственный экземпляр этого класса. Для доступа к этому экземпляру предоставляется статический метод или свойство.

Преимущества использования паттерна *Singleton* включают:

- гарантированное наличие только одного экземпляра класса;
- глобальная точка доступа, которая обеспечивает удобный способ получения доступа к экземпляру класса из любой части программы;
- ленивая инициализация экземпляра класса, что означает, что экземпляр создается только при первом обращении к нему;
- централизованное управление состоянием и операциями класса.

Однако, паттерн *Singleton* также имеет некоторые недостатки:

- нарушение принципа единственной ответственности (*SRP*), так как класс не

только отвечает за свои основные обязанности, но и за создание и управление экземпляром;

- усложняет тестирование, так как *Singleton* может создавать зависимости, которые трудно изолировать для модульного тестирования;

- может привести к созданию скрытых зависимостей и сложностей в проектировании системы.

1.7.3 Паттерн *Publish-Subscribe*

Паттерн *Publish-Subscribe* (или *Pub-Sub*), изображение которого находится на рисунке 20, относится к классу паттернов поведения (*behavioral patterns*) и используется для организации коммуникации между компонентами системы без явного связывания между ними. В этом паттерне компоненты взаимодействуют между собой через централизованную точку, называемую брокером сообщений.



Рисунок 20 – Схематичное изображение паттерна *Singleton*

Основная идея паттерна *Publish-Subscribe* заключается в том, что компоненты, называемые издателями (*publishers*), генерируют сообщения о событиях или изменениях состояния. Затем эти сообщения передаются брокеру сообщений, который является посредником между издателями и компонентами, называемыми подписчиками (*subscribers*). Подписчики регистрируются у брокера и указывают, на какие типы сообщений они хотят подписаться. Когда издатель генерирует сообщение, брокер передает его только тем подписчикам, которые

заинтересованы в данном типе сообщений.

Преимущества использования паттерна *Publish-Subscribe* включают:

- развязка компонентов. Издатели и подписчики не зависят друг от друга напрямую, что позволяет легко добавлять или удалять компоненты без внесения изменений в остальную систему;
- гибкость и масштабируемость. Компоненты могут быть добавлены или удалены без изменения самого паттерна и без влияния на остальные компоненты системы;
- асинхронность. Подписчики получают сообщения асинхронно, что позволяет реагировать на события или изменения состояния системы в реальном времени.

Однако, паттерн *Publish-Subscribe* также имеет некоторые недостатки:

- отсутствие гарантии доставки. Брокер сообщений не гарантирует, что каждое сообщение будет доставлено всем подписчикам, особенно в случае сетевых сбоев или ошибок;
- сложность отладки. Из-за отсутствия явной связи между издателями и подписчиками может быть сложно отследить, кто и когда получает определенные сообщения.

1.8 Разработка приложения

Образовательное игровое приложение «Магия знаний» состоит из четырех эпох, каждая эпоха содержит от пяти до шести площадей. В каждой площади рассказывается о каком-либо событии из истории нашей страны.

Эпохи:

- Древняя Русь. Данная эпоха состоит из шести площадей и охватывает период истории от прихода Рюрика до завершения татаро-монгольского ига;
- цари и императоры. В этой эпохе из пяти площадей рассказывается о периоде от начала правления Ивана Грозного до правления Екатерины II;

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		37

- Российская Империя. В третьей эпохе, состоящей из шести площадей, повествуется о периоде истории от начала правления Александра I до конца правления Николая II;

- СССР и РФ Данная эпоха из шести площадей охватывает историю от революции до образования Российской Федерации.

Для наглядного представления функциональных возможностей и взаимодействия между актерами и прецедентами в приложении была составлена диаграмма прецедентов в рамках языка моделирования *UML*. На диаграмме прецедентов отображаются актеры, которые взаимодействуют с системой, и прецеденты, которые представляют собой функциональные возможности приложения. Актеры могут быть пользователями, внешними системами или другими компонентами, которые инициируют действия в системе.

Прецеденты определяются как конкретные сценарии использования, описывающие взаимодействие между актерами и системой. Диаграмма изображена на рисунке 21.



Рисунок 21 – *UML* – диаграмма прецедентов

Для визуализации работы приложения и наглядного представления была составлена диаграмма последовательностей в рамках языка моделирования *UML*, она представлена на рисунке 22.

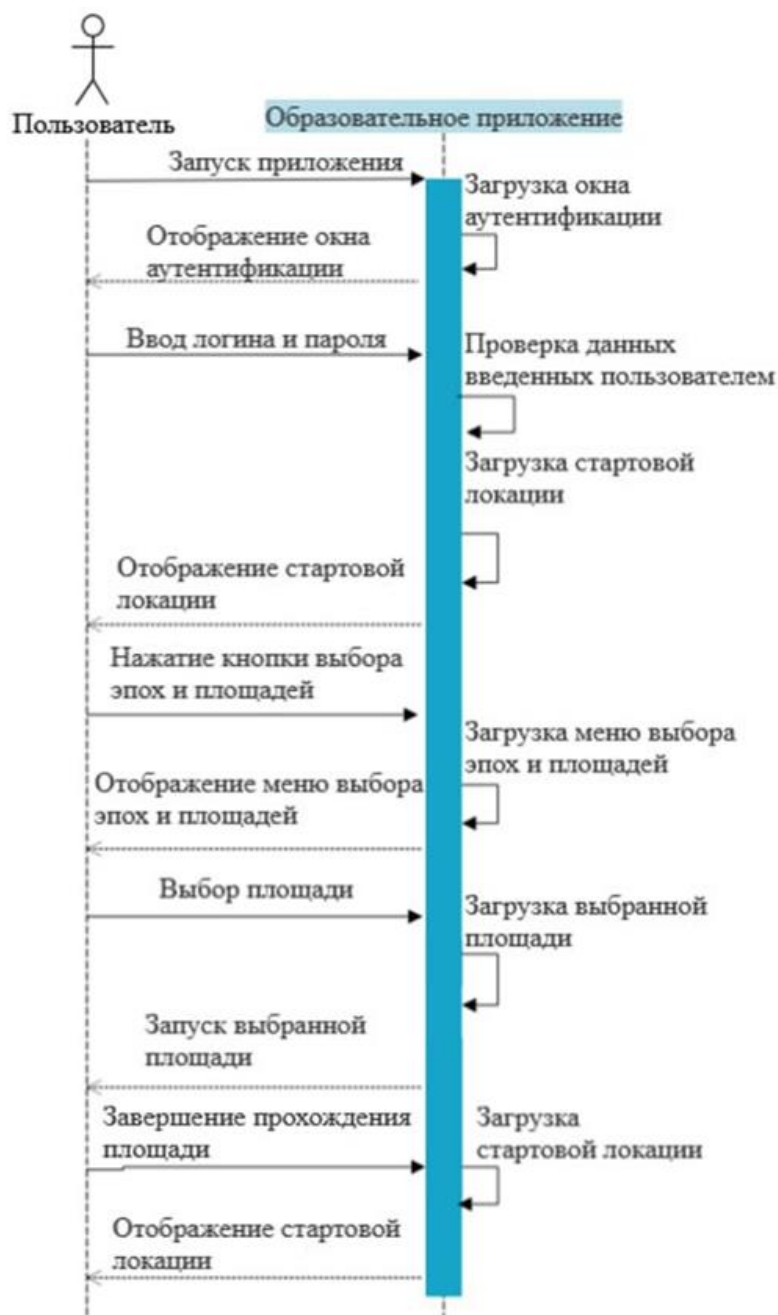


Рисунок 22 – *UML* – диаграмма последовательностей

1.8.1 Разработка системы диалогов

Диалоги играют важную роль в визуальных новеллах, поскольку они являются основой для передачи сюжета, характеров персонажей и создания

эмоциональной связи с игроком. Правильный выбор инструмента для создания диалоговых сцен становится критическим, чтобы обеспечить высокое качество и удобство разработки.

В этом контексте инструмент *Fungus* для *Unity* представляет собой отличное решение [3].

Fungus предоставляет разработчикам широкий спектр возможностей для создания диалоговых сцен в визуальных новеллах. Его основные достоинства включают:

- интуитивный интерфейс. *Fungus* предлагает простой и понятный визуальный интерфейс, который упрощает создание диалогов и легко освоить даже для новичков. Разработчики могут легко создавать диалоговые ветки, добавлять персонажей и настраивать текст и изображения;

- визуальное программирование. *Fungus* предлагает систему визуального программирования, основанную на блоках, что позволяет разработчикам легко управлять логикой и потоком диалогов. Блоки могут быть связаны вместе, создавая сложные сценарии и динамические ответы на выбор игрока;

- гибкость и настраиваемость. *Fungus* предлагает множество настраиваемых параметров и возможностей, которые позволяют разработчикам адаптировать его под свои потребности. Он предлагает различные стили диалогового окна, возможность добавления звуковых эффектов, анимаций и пользовательских скриптов;

- бесплатность и открытый исходный код;

- активное сообщество и ресурсы. *Fungus* имеет активное сообщество разработчиков, которые делятся своими знаниями, создают дополнительные расширения и предоставляют полезные ресурсы, такие как уроки и шаблоны, что помогает разработчикам быстро освоить инструмент и создавать качественные диалоги. Также инструмент обладает объемной документацией, расположенной на сайте *github*.

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		40

Fungus позволяет разработчикам сосредоточиться на сюжете и характерах персонажей, облегчая процесс создания диалогов и управления их логикой. Благодаря его возможностям визуального программирования, разработчики могут создавать сложные сценарии с разветвленными диалогами и динамическими ответами на выбор игрока.

Процесс создания диалога с помощью данного инструмента организуется с применением визуального программирования.

В окне *Flowchart* будет размещаться блоки диалогов, как на рисунке 23.

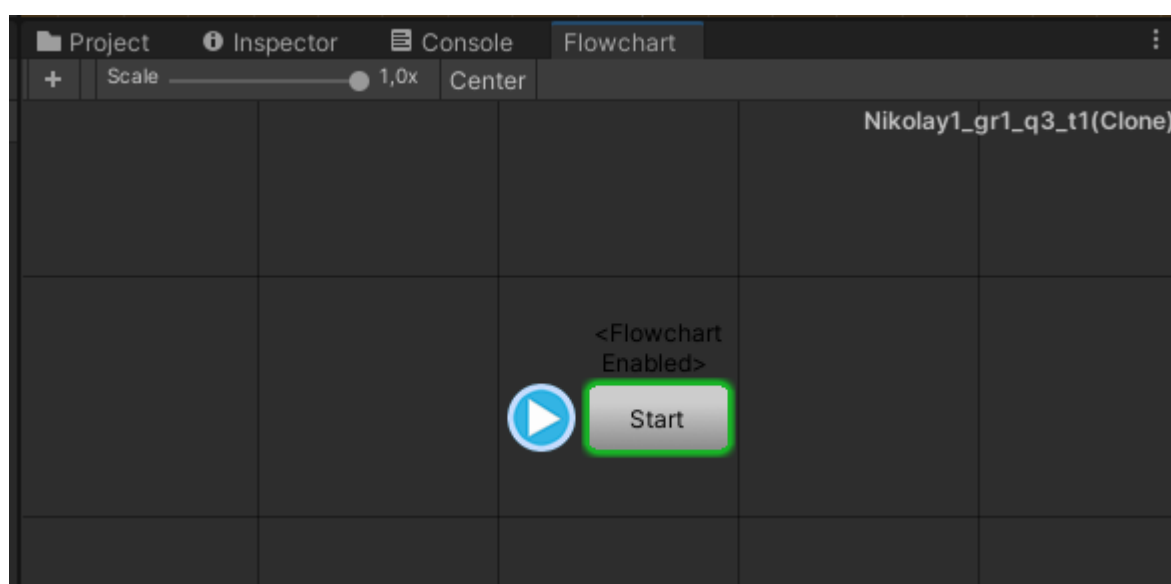


Рисунок 23 – Окно *Flowchart*

В окне *Inspector* задаются ветвления и команды, которые определяют очередность реплик и отображение персонажей в диалогах, пример на рисунках 24 и 25.

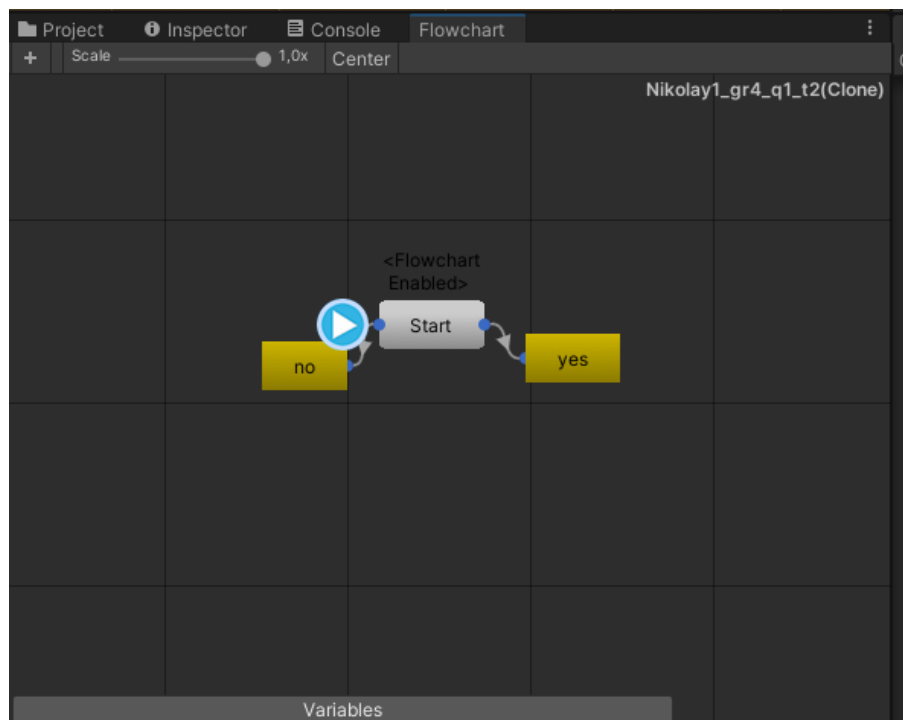


Рисунок 24 – Окно *Flowchart* с ветвлением диалога

Внутри блока создаются команды, которые отвечают, что и в каком порядке будет воспроизводиться на экране.

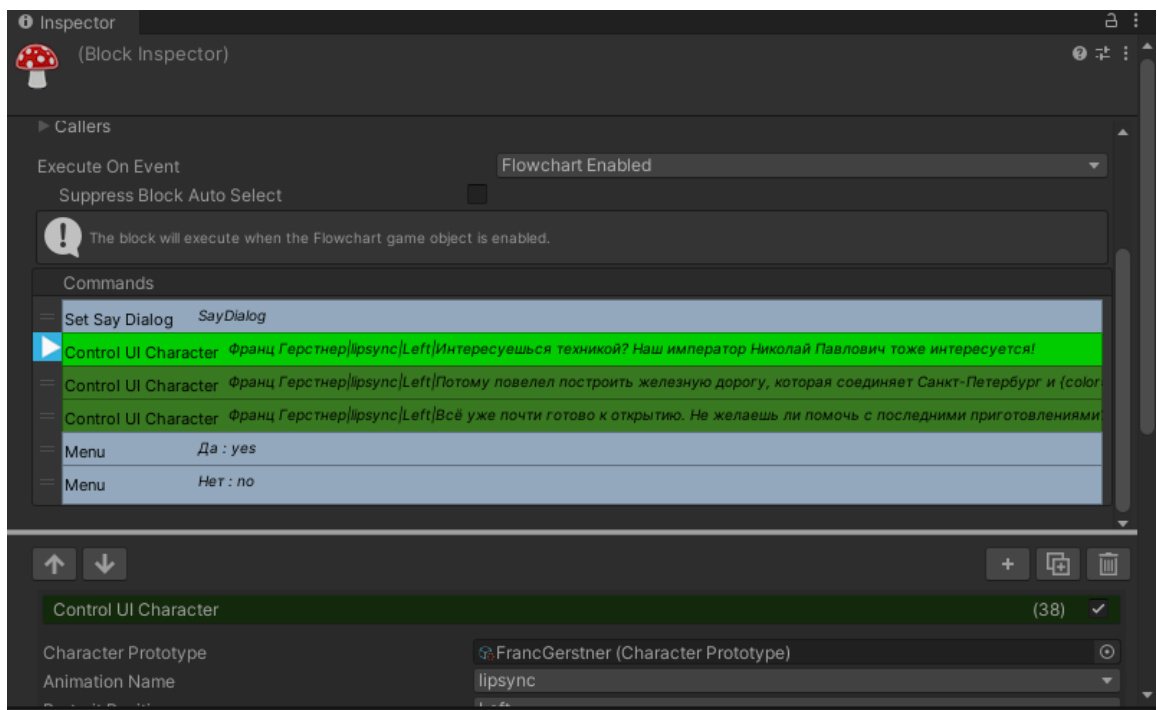


Рисунок 25 – Окно блока диалога

Основные используемые команды:

- *control UI character*. Данная команда осуществляет отображение персонажа и его реплики. Также здесь происходит настройка воспроизводимой анимации и звукового файла, рисунок 26;

- *menu*. Команда представлена на рисунке 27 и позволяет создать меню выбора, где пользователь может выбрать один из предложенных вариантов ответа. Вы можете указать варианты ответа и настроить соответствующие действия для каждого выбора;

- *call*. Данная команда позволяет вызвать другой блок диалога. Её можно использовать для организации переходов между различными частями диалога или для повторного использования определенных блоков.

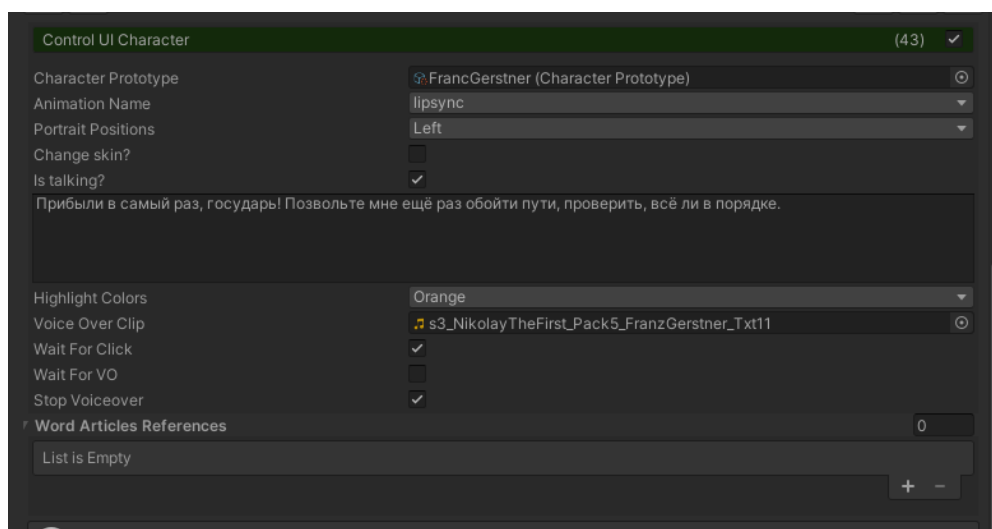


Рисунок 26 – Настройка *Control UI Character*

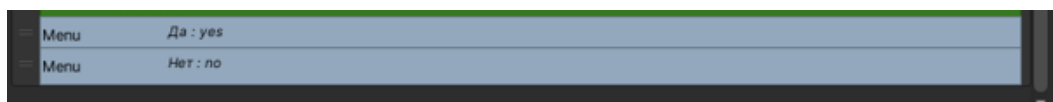


Рисунок 27 – Команда *Menu*

Существует 3 типа блоков:

- *event block* (Блок события). Этот тип блока активируется по определенному событию или условию. Он служит для инициации диалога или выполнения определенных действий в ответ на события в игре. Например, блок может быть активирован при входе персонажа в определенную зону, при

выполнении определенного условия или при нажатии кнопки.

- *branching block* (Блок ветвления). Этот тип блока позволяет создавать разветвления и выбор в диалоге. В блоке разветвления можно указать различные варианты ответов или действий, из которых игрок может выбрать. Каждый вариант может привести к разным последующим блокам диалога, в зависимости от выбора игрока.

- *standard block* (Стандартный блок). Этот тип блока представляет собой базовый блок диалога, который содержит текст или сообщение для отображения игроку. Стандартные блоки используются для представления непосредственного контента диалога, включая реплики персонажей, описания сцен или информацию для игрока.

Комбинирование этих трех типов блоков позволяет создавать сложные и интерактивные диалоговые сцены в *Fungus*. *Event Block* и *Branching Block* добавляют динамическую логику и вариативность в диалоги, а *Standard Block* используется для представления контента и информации в диалоге.

Логика всей сцены редко может уместиться в одном блоке, поэтому нам нужны способы, чтобы один блок передавал управление выполнением другим блокам. Есть несколько способов сделать это, наиболее распространенными из которых является:

- *the Call Command*;
- *the Menu Command*.

Для этого необходимо создать блок с вопросом и меню выбора, а также блоки с вариантами ответов как на рисунках 28, 29, 30, 31.

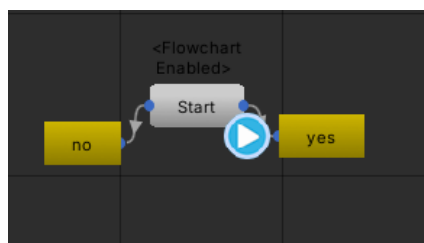


Рисунок 28 – Ветвление диалога

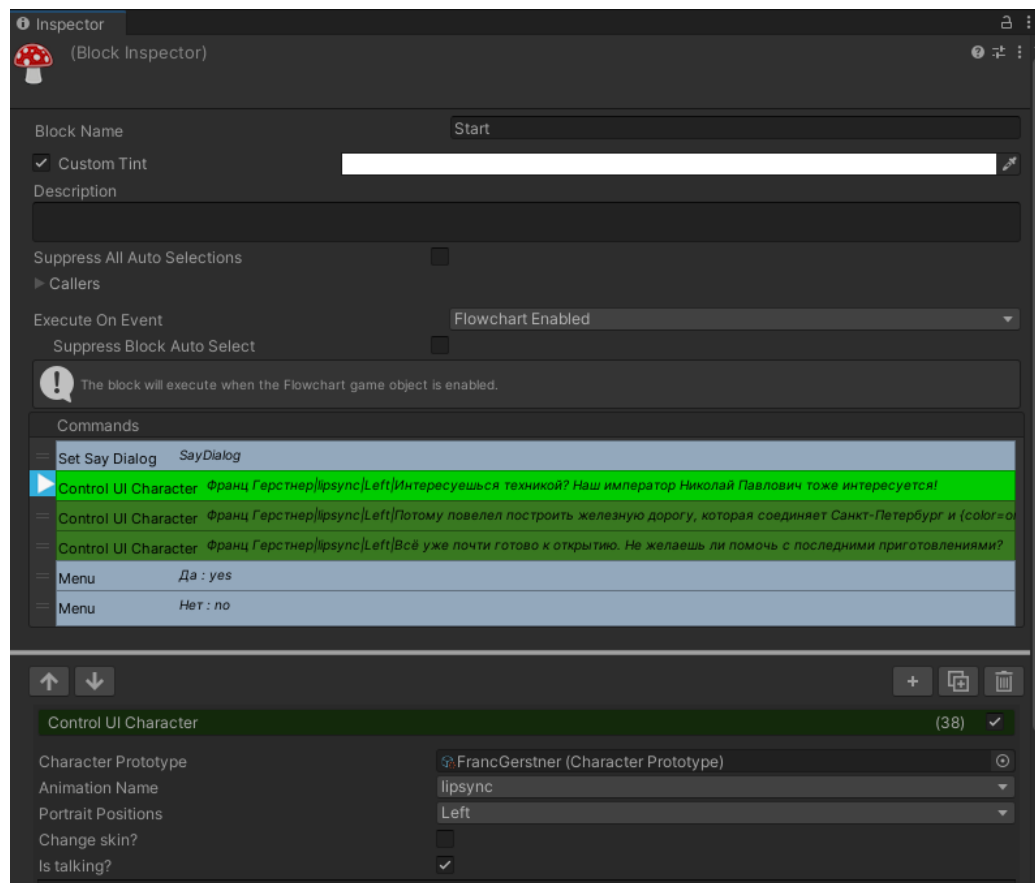


Рисунок 29 – Блок диалога с командами *меню*

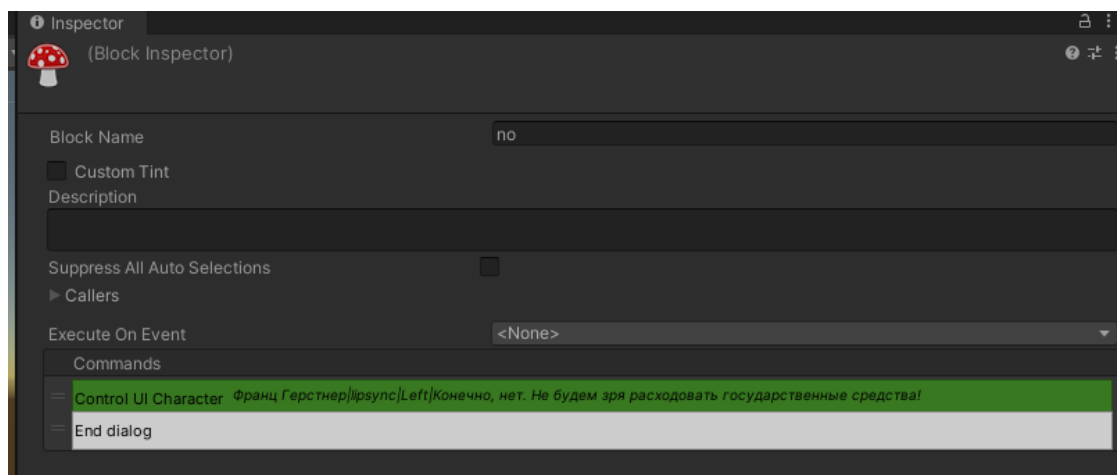


Рисунок 30 – Блок диалога для верного варианта ответа

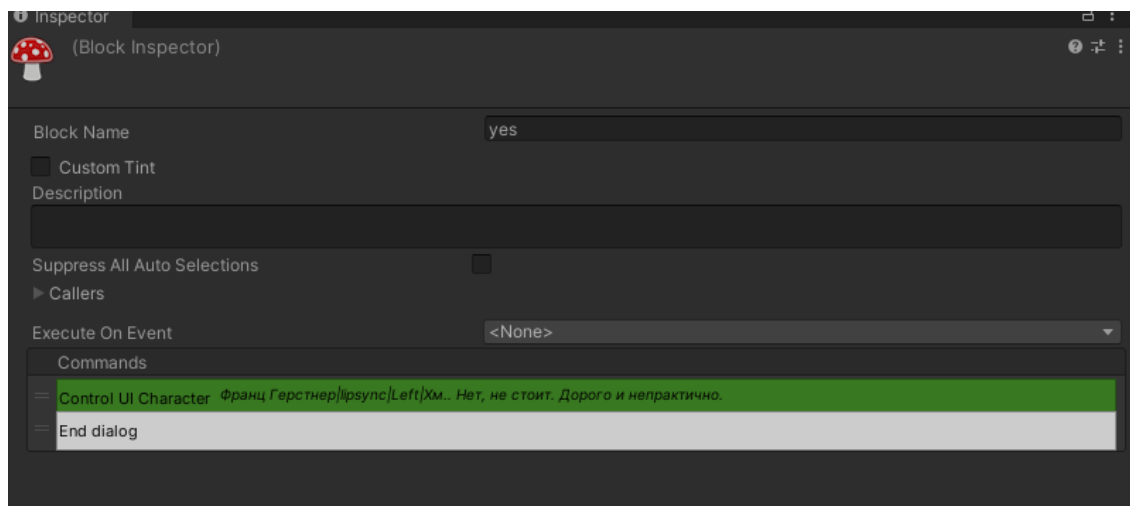


Рисунок 31 – Блок диалога для неверного варианта ответа

Пример диалога площади «Николай I» представлен на рисунках 32 и 33

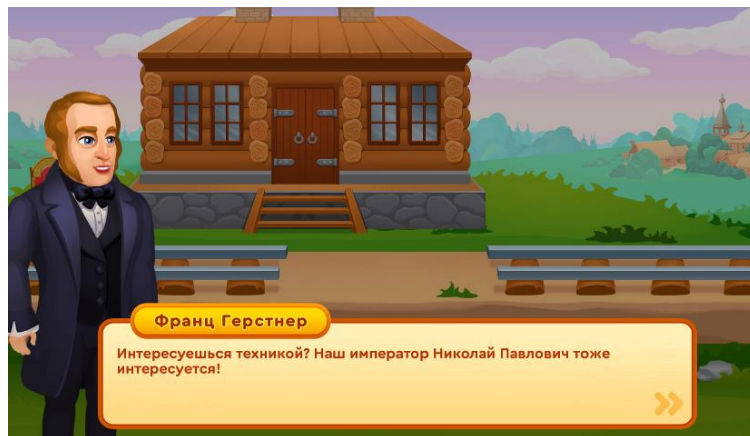


Рисунок 32 – Пример диалога площади

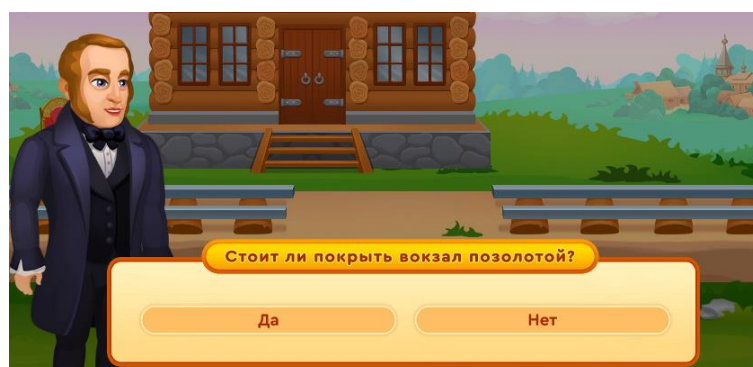


Рисунок 33 – Пример ветвления диалога площади

Таким образом, было создано более семисот объектов диалогов, согласно прописанному сценарию для каждой из четырех эпох на которые поделено приложение.

1.8.2 Разработка внутриигровой энциклопедии

Важным аспектом обучения школьников является возможность сохранения и быстрого доступа к полученным знаниям. В этом контексте разработана внутриигровая энциклопедия, которая предоставляет школьникам средство систематизации и хранения полученных знаний.

Внутриигровая энциклопедия, которая представлена на рисунке 34 - это специальный инструмент, встроенный в образовательное приложение, который позволяет сохранять и организовывать знания по различным следующим разделам:

- эпохи. Здесь происходит переключение между четырьмя эпохами, на которые поделено игровое приложение;
- места. В данный раздел добавляется информация о посещенных в ходе прохождения городах и странах, которые являются ключевыми для истории нашей страны;
- личности. Здесь хранится ключевая информация об исторических личностях, с которыми встретится пользователь во время изучения того или иного исторического периода;
- события. В данный раздел записывается информация о ключевых исторических событиях, в которых лично участвует пользователь или о которых ему повествуют ключевые исторические фигуры;
- предметы. Здесь хранится информация о предметах той или иной эпохи, с которыми пользователь уже взаимодействовал;
- слова. В данный раздел записываются определения слов, с которыми знакомится пользователь.

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		47



Рисунок 34 – Внутриигровая энциклопедия

В каждой записи энциклопедии присутствует анимированная модель или рисунок, это позволяет обучающемуся лучше представлять и визуализировать изучаемый материал, создается более яркое и запоминающееся впечатление, которое позволяет лучше понять и усвоить материал.

Кроме того, наличие рисунков и анимированных моделей в энциклопедии способствует визуальному обучению, которое является эффективным методом для многих людей. Визуальные элементы помогают воспринимать информацию более наглядно.

Работа энциклопедии определяется логикой, описанной в файлах:

- *EncyclopediaManager.cs*. В данном классе описывается логика загрузки статей энциклопедии из папки Assets;
 - *EncyclopediaUI.cs*. Логика описывающая отображение информации статей в пользовательском интерфейсе;
 - *EncyclopediaArticle.cs*. Здесь содержится класс статьи со всеми полями.
- UML диаграмма этих классов показана на рисунке 35 [4].

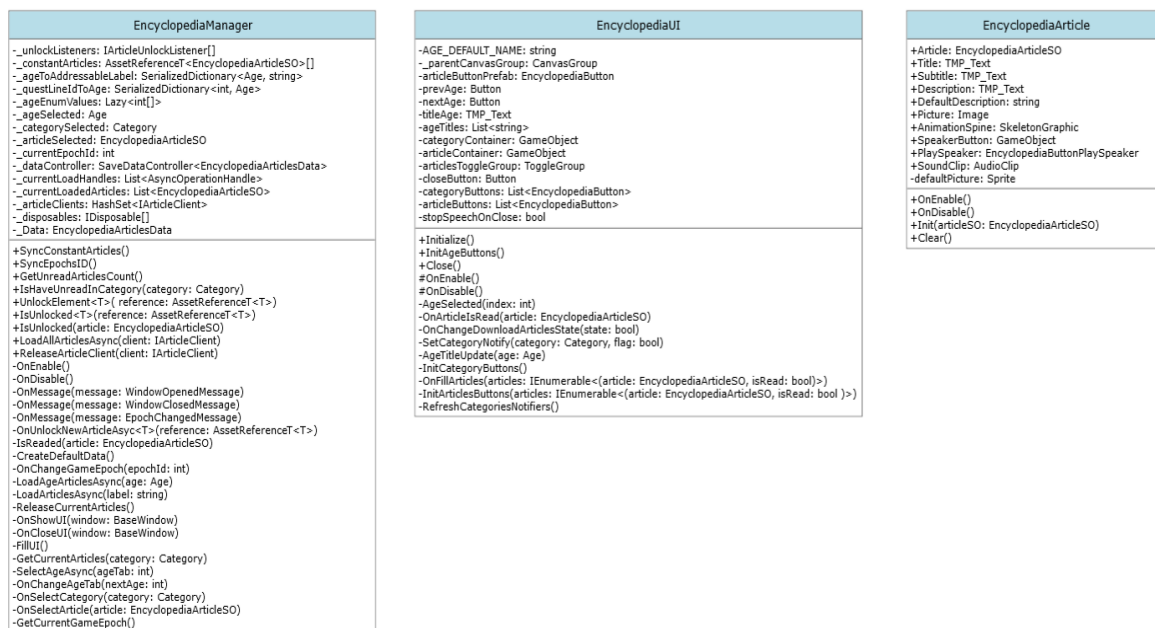


Рисунок 35 – UML диаграмма классов энциклопедии

1.8.2.1 Класс *EncyclopediaManager*

Данный класс содержит логику работы энциклопедией [5].

SyncConstantArticle - вывод кнопки в инспекторе *Unity* для запуска функции сбора прототипов статей, которые должны постоянно отображаться в энциклопедии.

SyncEpochsID - вывод кнопки в инспекторе *Unity* для запуска функции привязки квестовых линий и эпох к определенному разделу энциклопедии (*Age*).

GetUnreadArticlesCount – получение количества непрочитанных открытых статей.

IsHaveUnreadInCategory – проверка наличия непрочитанных статей.

UnlockElement – функция разблокирования статьи.

LoadAllArticlesAsync - загрузка статей при перелистывании эпох.

ReleaseArticleClient - освобождение памяти после загрузки статей и закрытии энциклопедии.

CreateDefaultData - создаются локальные сохранения если их нет.

LoadAgeArticlesAsync - загружаем статьи для выбранной эпохи (*Age*).

LoadArticlesAsync - Загружаем статьи из *Addressable* бандлов по тегу.
ReleaseCurrentArticles - освобождается память от загруженных статей.
OnShowUI - при показе окна загружаем статьи для выбранной эпохи.
OnCloseUI – при закрытии окна вызываем *ReleaseCurrentArticles*.
GetCurrentArticles - получаем все статьи из категории (при переключении вкладок сбоку).

OnChangeAgeTab - отслеживается переключение эпох.

OnSelectCategory - отслеживается выбор категорий.

OnSelectArticle - при выборе статьи загружается ее информацию для отображения.

GetCurrentGameEpoch - получаем текущую эпоху.

1.8.2.2 Класс *EncyclopediaUI*

Данный класс содержит логику отображения пользовательского интерфейса.

OnEnable - подписываемся на события менеджера энциклопедии, чтобы заполнять *UI*.

OnDisable - отписываемся от событий менеджера энциклопедии.

AgeSelected - при выборе эпохи смотрим что дошли до конца и выключаем возможность крутить дальше.

OnArticleIsRead - при прочтении статьи отключаем оповещение о непрочитанной статье.

OnChangeDownloadArticlesState - включаем возможность клика после загрузки всех статей эпохи.

SetCategoryNotify - включаем или выключаем иконку непрочитанной статьи.

AgeTitleUpdate - при выборе эпохи обновляем заголовок.

InitCategoryButtons - включаем доступные кнопки категорий.

OnFillArticles - обновляем *UI* при загрузке статей.

InitArticlesButtons - включаем кнопки для взаимодействия со статьями.

RefreshCategoriesNotifiers - обновляем иконки прочитано/непрочитано.

1.8.2.3 Класс *EncyclopediaArticle*

OnEnable - подписываемся на событие обновления информации о статье (выборе статьи).

Init - инициализируем отображение статьи.

Clear - очищаем отображение, если статьи не найдено

За время работы над проектом было создано 404 объекта статьи, которые заполняют энциклопедию по мере продвижения пользователя по сюжету истории.

Пример содержимого экземпляра объекта статьи представлен на рисунке 36. В нем указывается текстовое описание объекта, его спрайт и, если есть, звуковой файл, который будет воспроизводиться.

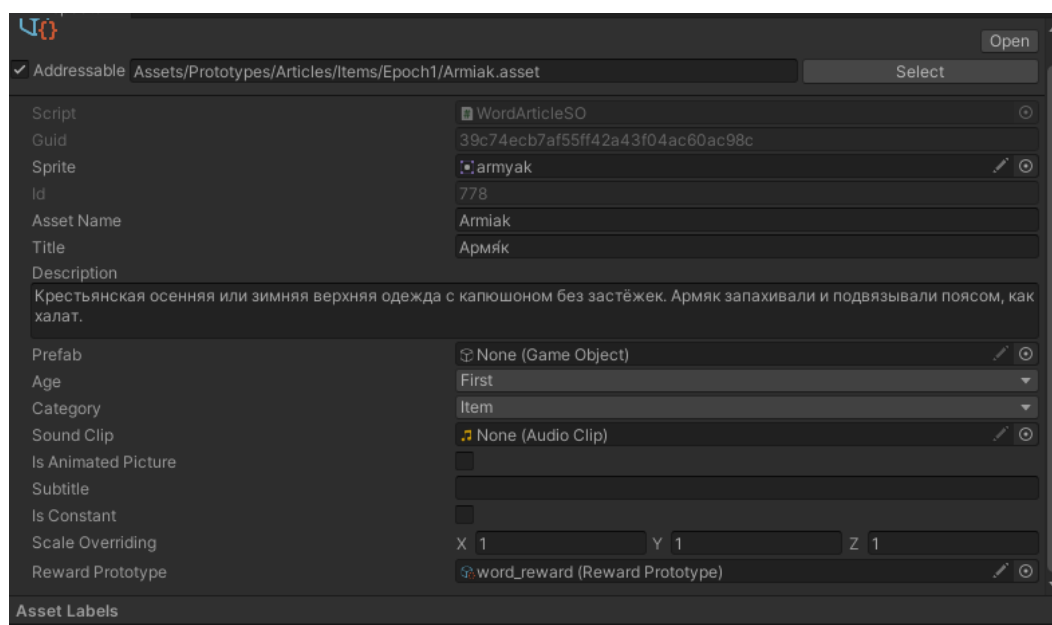


Рисунок 36 – Экземпляр класса статьи энциклопедии

1.9 Тестирование и отладка

После создания первых прототипов и вариантов площадей, команда столкнулась с задачей тестирования и выбора средства для отслеживания ошибок. Однако, в команде присутствовали не только программисты с высокими техническими знаниями, но и художники, аниматоры, а также вышестоящее руководство, которым требовался простой и понятный инструмент.

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		51

В такой ситуации было необходимо выбрать инструмент, который был бы доступным и удобным для всех участников команды. Важными критериями выбора были гибкость, совместное использование и простота в освоении. После тщательного анализа вариантов было принято решение использовать *Google Sheets* в качестве инструмента для отслеживания ошибок.

Google Sheets был выбран, так как предоставляет простой и интуитивно понятный интерфейс, который позволяет легко создавать и редактировать таблицы с багами. Это удобно как для программистов, так и для художников и руководства, которые могут вносить свои комментарии и отслеживать прогресс тестирования.

Кроме того, *Google Sheets* предлагает гибкость в настройке таблиц, что позволяет адаптировать их под конкретные потребности проекта. Таблицы могут быть легко совместно использованы и общедоступны для всех участников команды, что способствует эффективной коммуникации и совместной работе.

Поскольку *Google Sheets* не предназначен специально для работы в качестве среды отслеживания ошибок, команда столкнулась с необходимостью разработать способ сигнализации о состоянии каждой ошибки. Для этого было решено использовать различные цвета фона для пометки строк в таблице.

Когда ошибка обнаруживается, она записывается в таблицу с обычным белым фоном, как это изображено на рисунке 37. Однако, если баг является блокирующим и требует срочного внимания, его фон помечается красным цветом, как на рисунке 38. Это позволяет выделить его среди общего списка багов и обратить на него внимание.

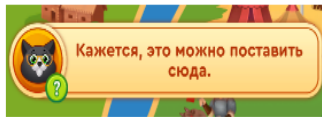
Площадь Александра Невского			
Описание проблемы	Шаги воспроизведения	Ожидаемый результат	Фактический результат
Не озвучен нотфаер в игре расположи войска (задание 5.2)	Перейти в задание 5.2 Кликнуть на подсказку		
https://drive.google.com/filed/1wJqZ0ASHRivjD4yVDGyWVCBT2OyC5/view?usp=share_link			

Рисунок 37 – Добавление ошибки в список

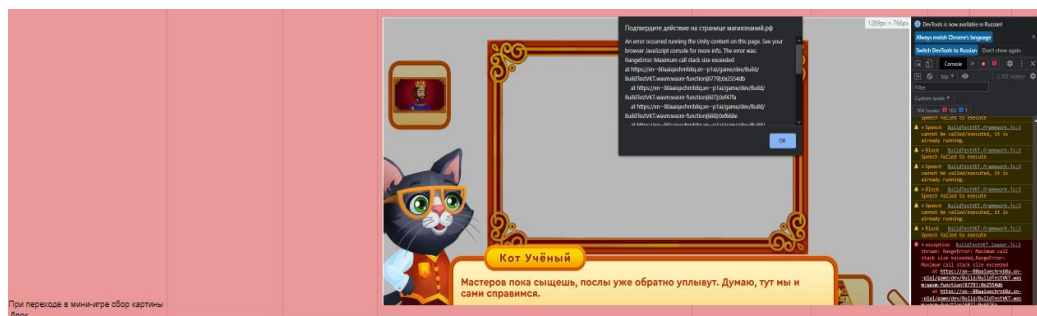


Рисунок 38 – Пример критической ошибки

Когда разработчик, художник или аниматор заканчивают работу над исправлением бага, они помечают его желтым цветом, пример на рисунке 39. Это сигнализирует о том, что баг был исправлен, но тестировщик не проверил это.

Площадь Александра Невского			
Описание проблемы	Шаги воспроизведения	Ожидаемый результат	Фактический результат
Не озвучен нотфаер в игре располочи войска (задание 5.2)	Перейти в задание 5.2 Кликнуть на подсказку		<p>Кажется, это можно поставить сюда.</p>

Рисунок 39 – Ошибка отмечена как исправленный

После очередного обновления сборки проекта тестировщик видит желтую строку и проверяет статус ошибки. Если исправление успешно внесено в проект и ошибка исправлена, она помечается зеленым цветом, как на рисунке 40 и перемещается в таблицу "Done" для отслеживания завершенных задач. Если исправление не вошло в сборку проекта, ошибка помечается серым цветом, и об этом сообщается руководителю или ответственному лицу.

Площадь Восточных Славян			
Описание проблемы	Шаги воспроизведения	Ожидаемый результат	Фактический результат
Пропали источники из всех мини-игр "переселения"	Перейти в мини-игру "переселения"	Ожидается, что все источники отображены корректно	<p>Клоун</p> <p>Славянка</p> <p>Пилот</p>

Рисунок 40 – Пример исправленной ошибки

Таким образом, использование разных цветов фона в таблице *Google Sheets* позволяет визуально отслеживать состояние каждого бага: от обнаружения до исправления и завершения задачи. Эта система помогает команде эффективно координировать работу и обеспечивает четкое понимание текущего статуса каждого бага в процессе разработки.

В результате выбор *Google Sheets* как инструмента для отслеживания ошибок оказался успешным решением, удовлетворяющим потребности всех членов команды.

Всего в ходе работы над проектом «Магия знаний» было выявлено и исправлено более полутра тысяч ошибок и недочетов различной степени важности.

1.10 Описание приложения

1.10.1 Руководство пользователя

После запуска приложения на любой из платформ откроется главное окно программы, пример на рисунке 41. Окно содержит две кнопки для авторизации и регистрации, пример на рисунках 42 и 43.

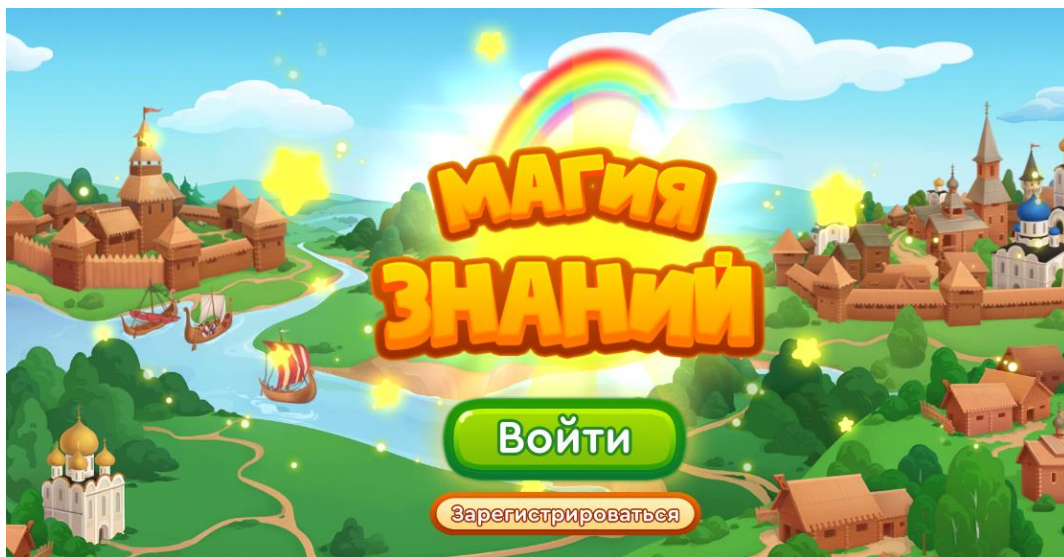


Рисунок 41 – Вкладки приложения

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		54

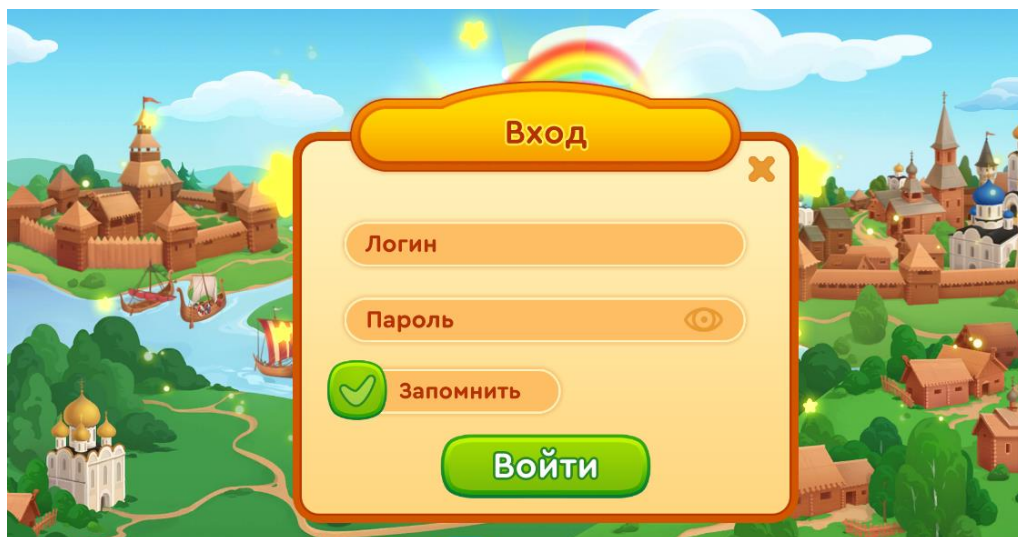


Рисунок 42 – Окно авторизации

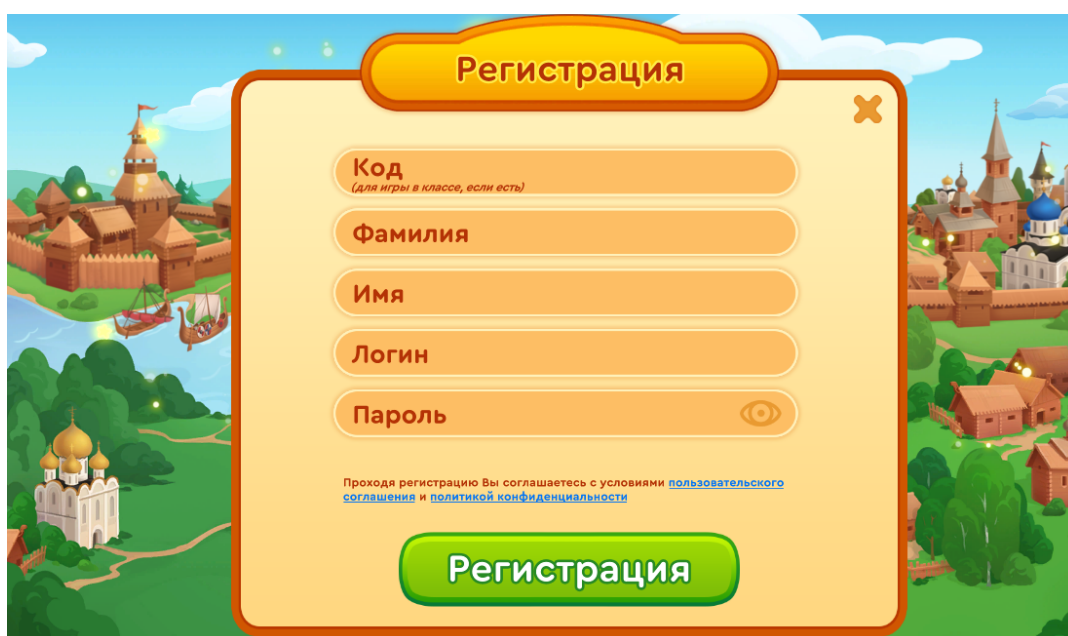


Рисунок 43 – Окно регистрации

Если пользователь взаимодействует с приложением в школе, то учитель должен сообщить ему сгенерированный код класса, это позволит объединить всех учеников одного класса.

Если пользователь не хочет присоединяться к классу, то поле можно оставить пустым.

После авторизации или создания аккаунта пользователь попадает в одну из

локаций игры, как это показано на рисунке 44.



Рисунок 44 – Локация

Также пользователь видит интерфейс, через который он взаимодействует с приложением.

При нажатии на кнопку с домовенком, как на рисунке 45, пользователь увидит информацию о своем персонаже, которую он вводил при регистрации, пример на рисунке 46.



Рисунок 45 – Кнопка открытия профиля персонажа



Рисунок 46 – Окно профиля

На рисунке 47, изображена кнопка, при нажатии на которую откроется окно с достижениями и ежедневными заданиями, как на рисунке 48.



Рисунок 47 – Кнопка открытия окна с заданиями



Рисунок 48 – Окно заданий

Кнопка для открытия окна с коллекциями изображена на рисунке 49, пример окна коллекций продемонстрирован на рисунке 50.

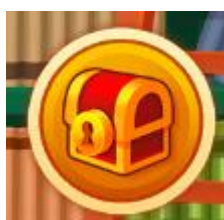


Рисунок 49 – Кнопка открытия окна коллекций



Рисунок 50 - Окно коллекций

Сами предметы пользователь собирает во время прохождения площадей.
Для того что бы изменить внешний вид своего персонажа необходимо нажать на кнопку, изображенную на рисунке 51.



Рисунок 51 – Кнопка для открытия окна изменения внешнего вида

Окно изменения внешнего вида представлено на рисунке 52.

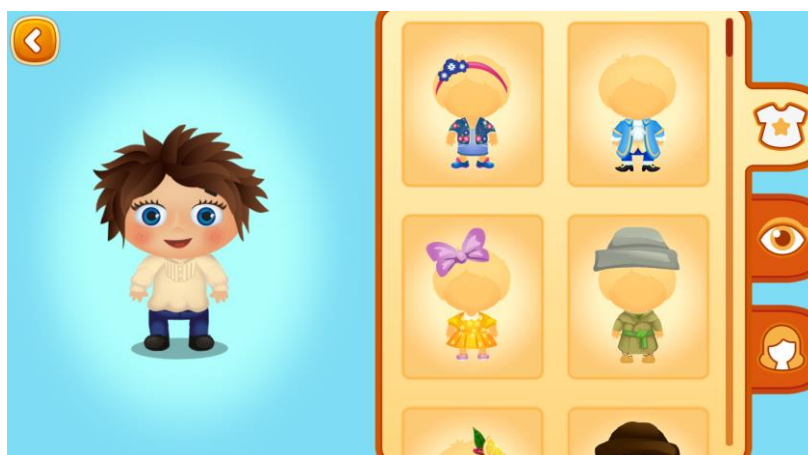


Рисунок 52 – Окно изменения внешнего вида персонажа

Что бы открыть окно энциклопедии необходимо нажать на кнопку, изображенную на рисунке 53.



Рисунок 53 – Кнопка открытия окна энциклопедии

Окно энциклопедии изображено на рисунке 54.



Рисунок 54 – Окно энциклопедии

Для перехода на локацию школы необходимо нажать на кнопку, изображенную на рисунке 55. Из школы пользователь может перейти в классы, с дополнительными мини-играми, локация школы изображена на рисунке 56.



Рисунок 55 – Кнопка перехода в локацию школы

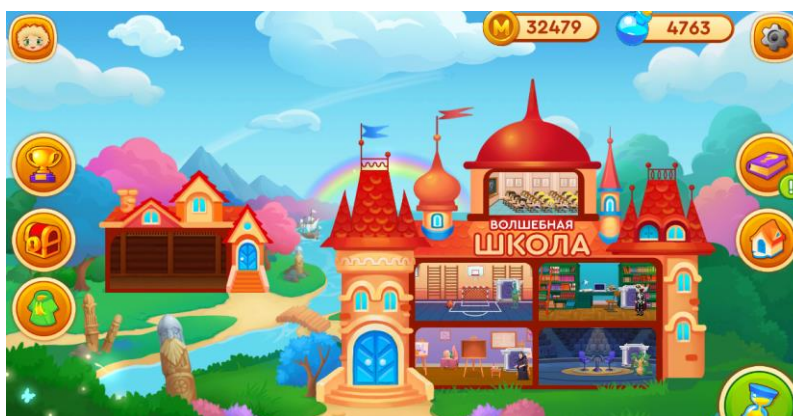


Рисунок 56 – Локация школы

Для перехода к выбору площади необходимо нажать на кнопку, изображённую на рисунке 57.

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		60



Рисунок 57 – Кнопка для перехода к выбору эпохи и площади

Далее пользователь может выбрать эпоху и площадь, пример на рисунке 58.



Рисунок 58 – Меню выбора эпохи и площади

2 Охрана труда

2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения

Работа с вычислительной техникой непременно сопровождается ограниченной подвижностью и длительным статическим мышечным напряжением, а также может привести к серьезному перенапряжению зрительных органов. Эти неблагоприятные факторы приводят к переутомлению, снижению работоспособности, психологическому раздражению, появлению физических недомоганий и общей усталости.

Что бы снизить количество вредоносного воздействия средств вычислительной техники на здоровье человека необходимо придерживаться ряда

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		61

правил использования ПЭВМ. Основными, из которых являются:

- правильная организация рабочего места;
- обеспечение достаточной освещенности рабочего места;
- соблюдение правил электробезопасности;
- соблюдение правил пожарной безопасности;
- регуляция режима труда и отдыха.

2.1.1 Оборудование рабочего места

Основные предъявляемые требования к рабочему месту:

- расстояние от глаз до экрана должно быть в пределах 600-700 мм;
- монитор необходимо расположить ниже уровня глаз под углом 10-15 градусов к направлению прямого взгляда;
- высота рабочего стола должна регулироваться в пределах 680-800 мм. В случае отсутствия такой возможности, высота стола должна составлять 725 мм;
- клавиатуру следует размещать на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной регулируемой по высоте рабочей поверхности;
- рабочий стол должен иметь достаточное пространство для ног, включая высоту не менее 600 мм, глубину на уровне колен не менее 450 мм и на уровне вытянутых ног не менее 650 мм.

На рисунке 59 изображена организация рабочего места, которая отвечает указанным требованиям.

					ПГУ1.090301.13.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

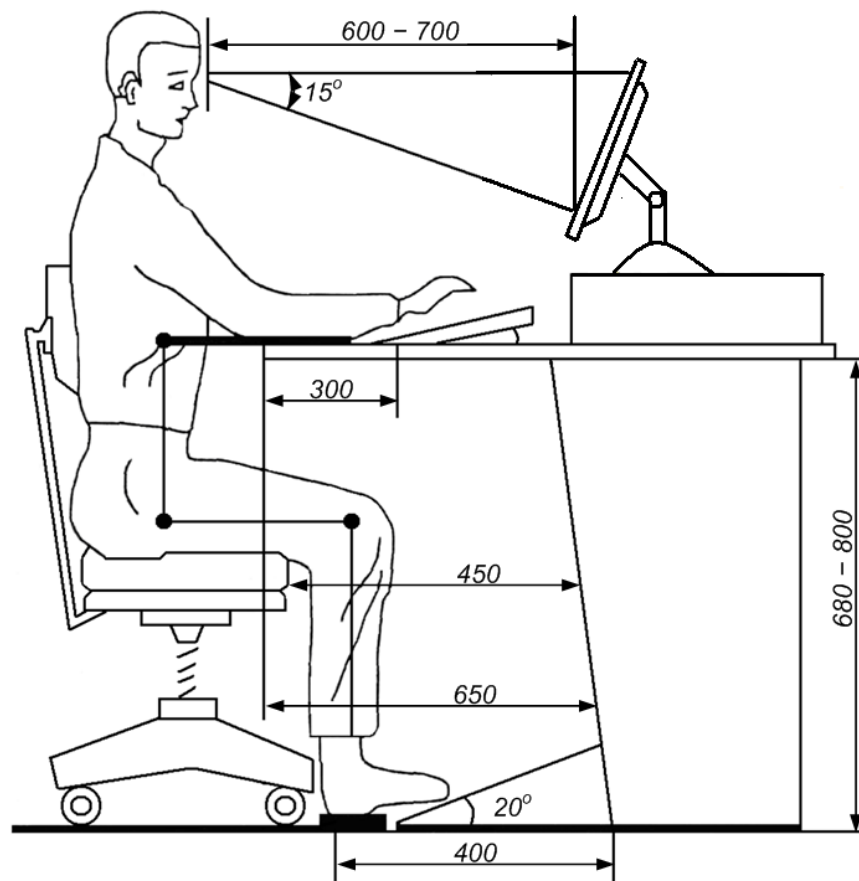


Рисунок 59 – Организация рабочего места

Важно, чтобы рабочее место с дисплеем обеспечивало удобство оператора в положении сидя и не создавало перегрузки костно-мышечной системы.

Организовав правильно рабочее место, также необходимо обеспечить правильную посадку пользователя ПК, правильная посадка изображена на рисунке 60

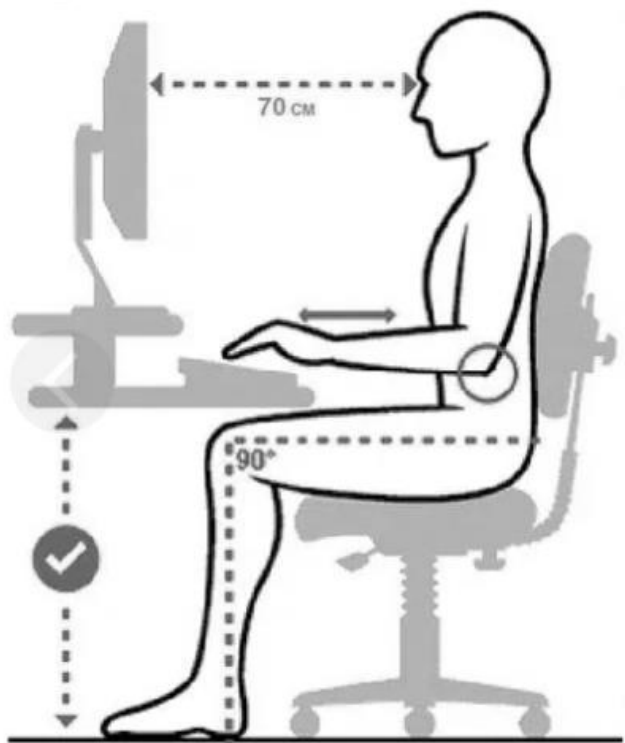


Рисунок 60 – Правильная посадка за ПК

Правильная посадка пользователя ПК обеспечивает выполнение следующих условий:

- запястья должны быть в прямом положении;
- плечи должны быть расслаблены;
- локти должны полностью лежать на столе, перпендикулярно полу (угол в локтевых суставах составляет 90-100 градусов).

2.1.2 Освещенность рабочего места

Стандарты освещенности определяются, учитывая характер работы, место и поверхность. Для пультов ЭВМ и дисплеев рекомендуется комбинированное освещение не менее 400 люкс и общее освещение не менее 200 люкс.

На рабочем месте необходимо обеспечивать равномерную яркость, исключая яркие и блестящие предметы. Для разнообразия в поле зрения рекомендуется использовать отдельные яркие поверхности.

В основном для освещения используются потолочные или встроенные светильники с люминесцентными лампами. Чтобы избежать засветки экранов,

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		64

светильники общего освещения следует устанавливать сбоку от рабочего места, параллельно линии зрения оператора.

2.1.3 Электробезопасность

Требования к электробезопасности регламентируются ГОСТ 12.1.019-2017 "ЭЛЕКТРОБЕЗОПАСНОСТЬ".

ПК работает от сети переменного тока 50 Гц с напряжением 230 В, что представляет опасность для жизни. При работе на ПК необходимо соблюдать следующие требования:

- все устройства должны быть подключены к одной фазе электрической сети;
- питание устройств ПК должно осуществляться только через розетки типа "Европа" с заземляющими контактами;
- все розетки для подключения устройств ПК должны иметь маркировку по напряжению;
- номинальное напряжение сети должно быть нанесено на стену или щит крупными символами яркой краской (высотой не менее 50 мм) возле розетки.

При использовании вычислительной техники и периферийного оборудования необходимо проявлять осторожность в отношении электропроводки и приборов, помнить, что нарушение правил безопасности может привести к угрозе здоровью и жизни человека.

Необходимо регулярно следить за исправностью электропроводки, выключателей и розеток, используемых для подключения оборудования.

При обнаружении неисправностей немедленно отключите электрооборудование. Продолжение работы возможно только после устранения всех неисправностей.

2.1.4 Пожарная безопасность

Наиболее распространенными причинами пожаров при использовании электроустановок и ПК являются:

- короткое замыкание проводов и электрических устройств;
- воспламенение горючих материалов рядом с электрическими устройствами;

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		65

-большие переходные сопротивления в контактных соединениях.

Профилактические меры по пожарной безопасности проводятся в соответствии с федеральным законом от 21.12.1994 №69-ФЗ (ред. от 28.09.2010) "О пожарной безопасности". Для обеспечения пожарной безопасности необходимо принять ряд организационных мероприятий, направленных на обеспечение безопасности людей, предотвращение пожара и его распространения, а также создание условий для эффективного тушения пожаров. Для повышения противопожарной безопасности помещений следует принимать следующие меры:

- проходы и запасные выходы не должны быть загромождены предметами, препятствующими свободному прохождению;
- средства пожаротушения должны быть всегда доступны;
- система вентиляции должна быть оснащена устройствами, автоматически отключающими ее при пожаре, а также огнезадерживающими клапанами;
- автоматическая пожарная сигнализация и средства пожаротушения должны быть в исправном состоянии и соответствовать типовым правилам технического содержания установок пожарной автоматики.

2.1.5 Режим труда и отдыха

Работник, занятый работой с ПК, должен соблюдать режим труда и отдыха в зависимости от продолжительности, вида и категории трудовой деятельности. Различаются следующие виды трудовой деятельности, связанные с использованием персонального компьютера:

- группа А. Работа по чтению информации с экрана видеодисплейного терминала с предварительным запросом;
- группа Б. Работа по вводу информации;
- группа В. Творческая работа в режиме диалога с ПЭВМ.

При выполнении работ, относящихся к различным видам трудовой деятельности, основной вид деятельности с ПК определяется как тот, который занимает не менее 50% времени в течение всей рабочей смены или рабочего дня.

Для каждого вида трудовой деятельности устанавливаются 3 категории

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		66

тяжести и напряженности работы с ПК:

- для группы А категория определяется суммарным числом считываемых знаков за рабочую смену (не более 60000 знаков за смену);

- для группы Б категория определяется суммарным числом считываемых или вводимых знаков за рабочую смену (не более 40000 знаков за смену);

- для группы В категория определяется суммарным временем прямой работы с ПК за рабочую смену (не более 6 часов за смену).

Таблица 2.1 отображает общее время перерывов для каждой категории и вида работы.

Таблица 2.1 – Время перерывов

Категория работы с ПЭВМ или ВДТ	Уровень нагрузки за рабочую смену при видах работы с ВДТ			Суммарное время регламентированных перерывов, мин	
	группа А, количество знаков	группа Б, количество знаков	группа В час	при 8- часовой смене	при 12- часовой смене
1	До 20000	До 15000	До 2	50	80
2	До 40000	До 30000	До 4	70	110
3	До 60000	До 40000	До 6	90	140

Заключение

В ходе выполнения выпускной квалификационной работы было разработано образовательное приложение «Магия знаний» на движке *Unity*, с использованием языка *C#*, как основного языка программирования. Диалоговая система реализована с использованием инструмента *Fungus*.

Разработанное приложение позволяет изучать историю нашей страны с древнейших времен по настоящее время.

Одним из достоинств разработанного приложения является наличие энциклопедии и системы диалогов с персонажами, а также наличие версий для всех основных платформ и их корректная работа.

В дальнейшем можно доработать разработанное приложение, улучшив оптимизацию, для того чтобы игра работала стабильнее на более слабых устройствах. Создать собственную систему диалогов, которая автоматизирует создание диалогов, например, из *excel* таблиц. Добавить большее количество мини-игр, чтобы разнообразить игровой процесс и сделать его еще более интересным. Расширить взаимодействие между пользователями, сделать процесс более кооперативным.

					ПГУ1.090301.13.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

Список используемых источников

1. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд.: Пер. с англ. — СПб. : ООО “Диалектика”, 2018 — 1328 с.
2. Зиборов В.В. MS Visual C# 2010 в среде .NET. Библиотека программиста. — СПб.: Питер, 2012 — 320 с.: ил.
3. Документация к инструменту создания диалогов в Unity. [Электронный ресурс] / Режим доступа: <https://github.com/fungus/wiki> свободный. (дата обращения: 03.04.2022).
4. Дж. Рамбо, М. Блаха UML 2.0. Объектно-ориентированное моделирование и разработка, 2-е издание. – СПб.: Питер, 2007 — 544 с.
5. Документация к движку Unity. [Электронный ресурс] / Режим доступа: <https://docs.unity3d.com/> свободный. (дата обращения: 01.04.2023).

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		69

Приложение А

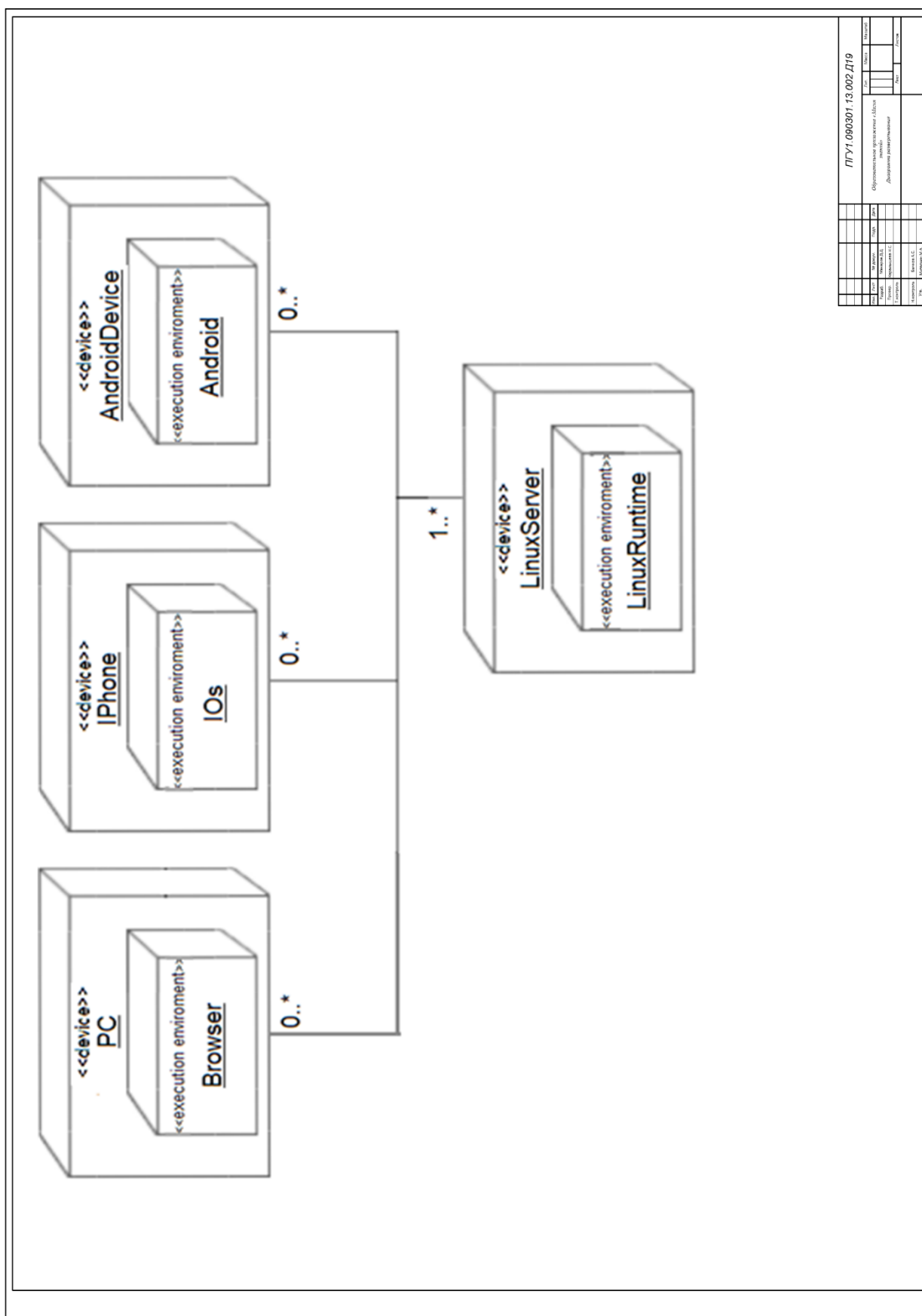
(Обязательное)

Образовательное игровое приложение «Магия знаний»

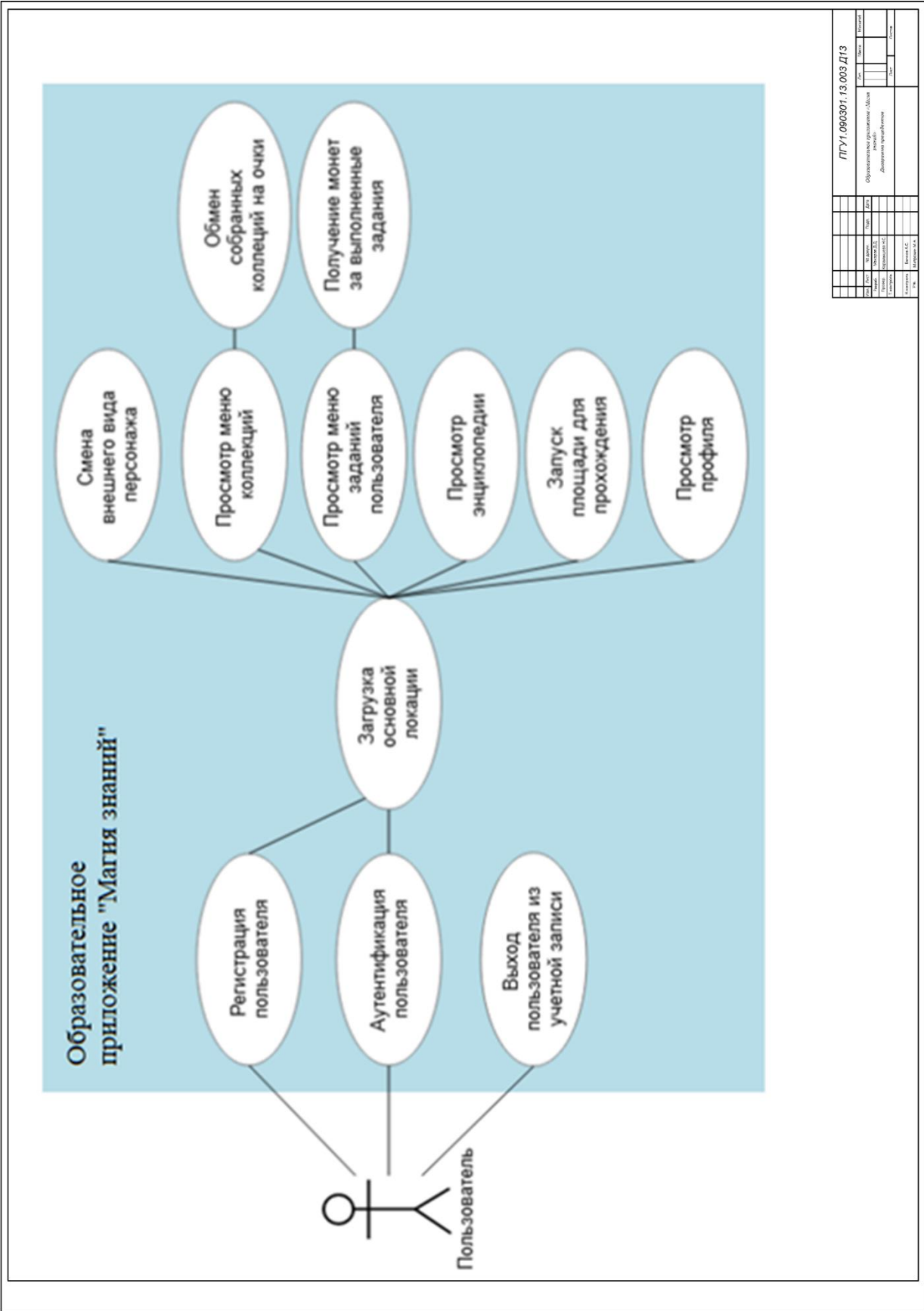
UML диаграммы

					ПГУ1.090301.13.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А. 1 – *UML* – диаграмма развертывания



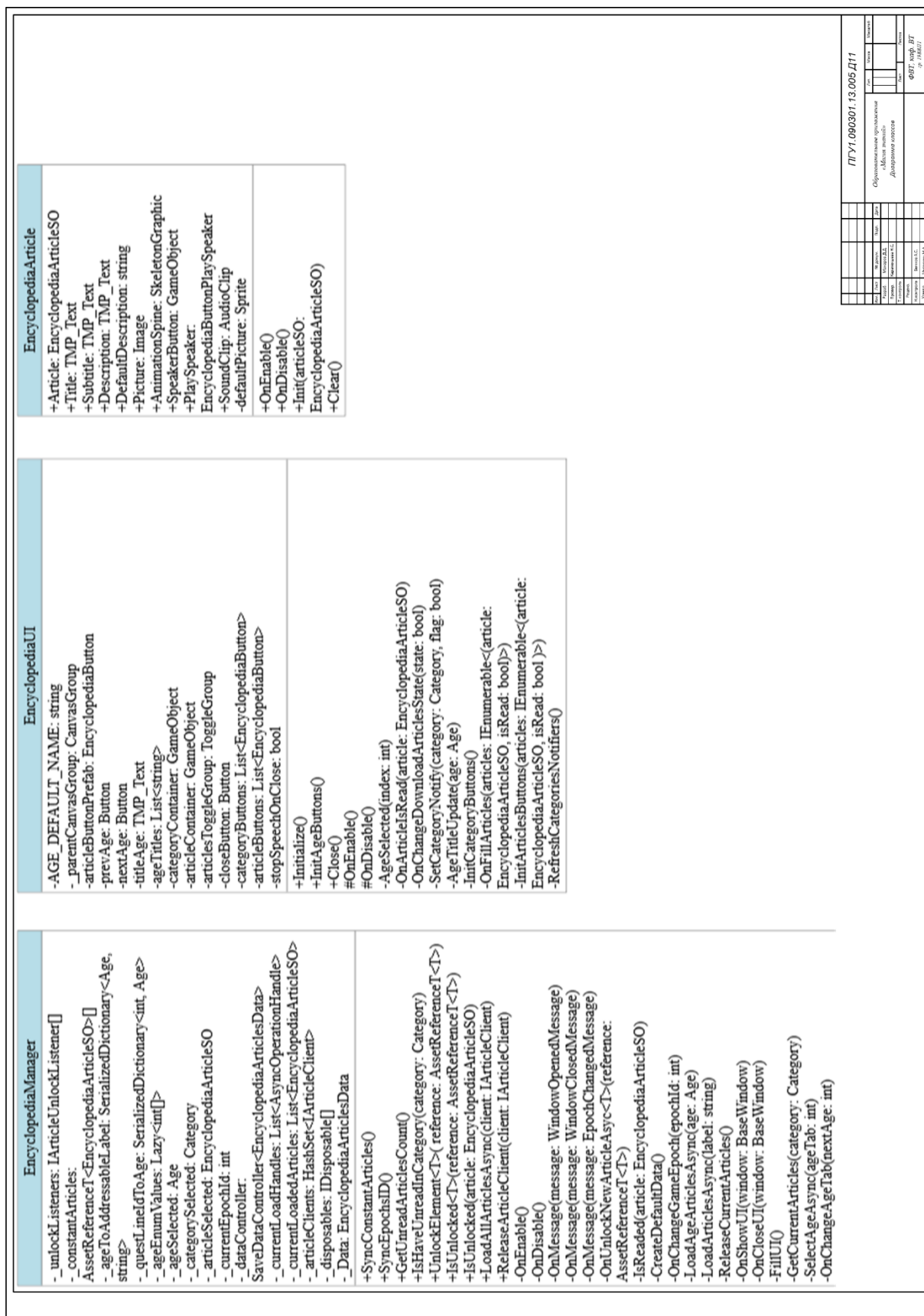
Приложение А.2 – UML – диаграмма прецедентов



Приложение А.3 –UML – диаграмма последовательностей



Приложение А.4 – UML – диаграмма классов



ПГУ1.090301.13.005 Д11									
Образовательная организация									
Министерство образования									
Департамент образования									
Учебно-методический кабинет									
Дата: 08.07.2017									
Подпись: _____									
Инициалы: _____									

Приложение Б

(Обязательное)

Образовательное приложение «Магия знаний»

Листинг программы

					ПГУ1.090301.13.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

Файл *EncyclopediaManager.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using Cysharp.Threading.Tasks;
using UnityEngine;
using UnityEngine.AddressableAssets;
using UnityEngine.Events;
using UnityEngine.ResourceManagement.AsyncOperations;
using VKT.Architecture.Addressables;
using VKT.Encyclopedia.ArticleClient;
using VKT.Encyclopedia.ArticleUnlocker;
using VKT.Encyclopedia.EncyclopediaCollection;
using VKT.Preservation;
using VKT.Windows;
using WainWork.Extensions;
using WainWork.Helpers;
using VKT.Architecture;
using VKT.Epochs;
using VKT.QuestSystem;
using Sirenix.OdinInspector;
using VKT.Messages;

namespace VKT.Encyclopedia
{
    public class EncyclopediaManager :
    SingletonMonoBehaviour<EncyclopediaManager>,
    IArticleUnlockController
    {
        #region ACTIONS

        public static event UnityAction<int> OnAgeSelected;

        public static event UnityAction<EncyclopediaArticleSO>
        OnArticleIsRead;
        public static event
        UnityAction<IEnumerable<(EncyclopediaArticleSO article, bool
        isRead)>> OnFillArticleList;
        public static event UnityAction<EncyclopediaArticleSO>
        OnFillArticleData;
        public static event UnityAction<EncyclopediaArticleSO>
        OnNewArticleUnlocked;
        public static event Action<bool>
        OnChangeDownloadArticlesState;

        public static UnityAction<Category> OnCategorySelected;
        public static UnityAction<EncyclopediaArticleSO>
        OnArticleSelected;
    }
}
```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		76

```

public static UnityAction<int> OnAgeTabChanged;
#endregion

[SerializeReference] private IArticleUnlockListener[]
_unlockListeners;
[SerializeField] private
AssetReferenceT<EncyclopediaArticleSO>[] _constantArticles;
[SerializeField] private SerializedDictionary<Age,
string> _ageToAddressableLabel = new();
[SerializeField] private SerializedDictionary<int, Age>
_questLineIdToAge = new();

private readonly Lazy<int[]> _ageEnumValues =
new(Enum.GetValues(typeof(Age)).Cast<int>().ToArray());
private Age _ageSelected;
private Category _categorySelected;
private EncyclopediaArticleSO _articleSelected;
private int _currentEpochId;
private SaveDataController<EncyclopediaArticlesData>
_dataController;
private List<AsyncOperationHandle> _currentLoadHandles = new(20);

private List<EncyclopediaArticleSO> _currentLoadedArticles =
new(20);

private HashSet<IArticleClient> _articleClients = new(2);
private IDisposable[] _disposables;

private EncyclopediaArticlesData Data
{
    get
    {
        if (_dataController.IsNullOrDefault())
        {
            _dataController =
                new
SaveDataController<EncyclopediaArticlesData>(EncyclopediaArticlesDat
a.UniqId,
                                CreateDefaultData);
        }

        return _dataController.GetData();
    }
}

[Button] //вывод кнопки в инспекторе Unity для запуска
функции сбора прототипов статей, которые должны постоянно
отображаться в энциклопедии
public void SyncConstantArticles()
{

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		77

```

var articles =
AssetsEditorUtility.FindAndLoadAssets<EncyclopediaArticleSO>()

.Where(article => article.IsConstant)
.Select(article => new
AssetReferenceT<EncyclopediaArticleSO>(article.Guid));

_constantArticles = articles.ToArray();
AssetsEditorUtility.DirtyAndSave(this);
}

[Button]//вывод кнопки в инспекторе Unity для запуска функции
привязки квестовых линий и эпох к определенному разделу энциклопедии
(Age)

public void SyncEpochsID()
{
var epochs =
AssetsEditorUtility.FindAndLoadAssets<EpochPrototype>().ToArray();

var questLines =
AssetsEditorUtility.FindAndLoadAssets<QuestLinePrototype>();

_questLineIdToAge.Clear();

foreach (var questLine in questLines)
{
var linkedEpoch = epochs.FirstOrDefault(epoch
=>
epoch.GetQuestLines().Any(wrapper =>
wrapper.AssetGUID == questLine.Guid));
var age = linkedEpoch != null ? linkedEpoch.Age
: Age.Master;

_questLineIdToAge[questLine.Id] = age;
}

AssetsEditorUtility.DirtyAndSave(this);
}

#region MONO

private void OnEnable() // вызывается при старте -
устанавливает обработчики нажатия кнопок в интерфейсе, и слушателей
событий открытия новой статьи и открытия/закрытия окна энциклопедии

{
foreach (var unlocker in _unlockListeners)
{
unlocker.Initialize(this);
}

OnAgeTabChanged += OnChangeAgeTab;

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		78

```

        OnCategorySelected += OnSelectCategory;
        OnArticleSelected += OnSelectArticle;

        _disposables = new[]

    {

        MessageUtility.Subscribe<WindowOpenedMessage>(OnMessage),
        MessageUtility.Subscribe<WindowClosedMessage>(OnMessage),
        MessageUtility.Subscribe<EpochChangedMessage>(OnMessage)
    };

    private void OnDisable() // обратное к предыдущему -
отвязываем слушателей
    {
        foreach (var unlocker in _unlockListeners)
        {
            unlocker.Release();
        }

        OnAgeTabChanged -= OnChangeAgeTab;
        OnCategorySelected -= OnSelectCategory;
        OnArticleSelected -= OnSelectArticle;
        _disposables?.Dispose();

        _disposables = null;
    }
    //вызываем соответствующие функции на открытие/закрытие
окна / перелистывание эпох в энциклопедии
    private void OnMessage(WindowOpenedMessage message)
    {
        OnShowUI(message.Window);
    }

    private void OnMessage(WindowClosedMessage message)
    {
        OnCloseUI(message.Window);
    }

    private void OnMessage(EpochChangedMessage message)
    {
        OnChangeGameEpoch(message.ID);
    }

    #endregion

    //получаем количество непрочитанных открытых статей
    public int GetUnreadArticlesCount()

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		79

```

        {
            return Data.Dates.Count(data => !data.IsRead);
        }
        //проверяем есть ли неп прочитанные статьи
        public bool IsHaveUnreadInCategory(Category category)
        {
            return GetCurrentArticles(category).Any(tuple =>

!tuple.isRead);
        }

        //разблокируем статью
        public void UnlockElement<T>(AssetReferenceT<T>
reference) where T : EncyclopediaArticleSO
        {
            if (!reference.IsHaveGuid())
            {
                return;
            }

            var data = Data.Dates.FirstOrDefault(dataContainer
=> dataContainer.Guid == reference.AssetGUID);

            if (data != default)
            {
                return;
            }

            var newContainer = new EncyclopediaArticleData
            {
                Guid = reference.AssetGUID,
                IsRead = false,
            };

            Data.Dates.Add(newContainer);
            _dataController.SaveData();

            OnUnlockNewArticleAsync(reference).Forget();
        }
        //проверка статьи на разблокировку (для мягких ссылок
из addressables)
        public bool IsUnlocked<T>(AssetReferenceT<T> reference)
        where T : EncyclopediaArticleSO
        {
            return Data.Dates.FirstOrDefault(data => data.Guid ==
reference.AssetGUID) != null;
        }
    }

```

					ПГУ1.090301.13.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

```

        //проверка статьи на разблокировку(для жестких ссылок из
addressables)
        public bool IsUnlocked(EncyclopediaArticleSO article)
        {
            return Data.Dates.FirstOrDefault(data => data.Guid
== article.Guid) != null;
        }
        //загрузка статей при перелистывании эпох
        public async UniTask<(bool success,
IEnumerable<EncyclopediaArticleSO> articles)>

        LoadAllArticlesAsync(IArticleClient client)
        {
            if (_articleClients.Count > 0)
            {
                _articleClients.Add(client);
                return (true, _currentLoadedArticles);
            }

            var result = await
AddressableManager.Instance.TryShowDownloadDialogAsync(_ageToAddress
ableLabel.Values,
                CancellationToken.None);
            if (!result)
            {
                return (false,
Enumerable.Empty<EncyclopediaArticleSO>());
            }

            _articleClients.Add(client);

            var labels = _ageToAddressableLabel.Values;
            var tasks = new UniTask[labels.Count];

            var index = 0;
            foreach (var label in labels)
            {
                tasks[index] = LoadArticlesAsync(label);
                index++;
            }

            await UniTask.WhenAll(tasks);

            return (true, _currentLoadedArticles);
        }
        //освобождение памяти после загрузки статей и закрытии
энциклопедии
        public void ReleaseArticleClient(IArticleClient client)
        {
            if (!_articleClients.Remove(client))
            {

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		81

```

        return;
    }

    if (_articleClients.Count == 0)
    {
        ReleaseCurrentArticles();
    }
}

private async UniTask
OnUnlockNewArticleAsync<T>(AssetReferenceT<T> reference) where T :
EncyclopediaArticleSO
{
    using var scope = await
AssetReferenceAsyncScope<T>.CreateAsync(reference);
    OnNewArticleUnlocked?.Invoke(scope.LoadedObject);

    await UniTask.Yield();
}

private bool IsReaded(EncyclopediaArticleSO article)
{
    var data = Data.Dates.FirstOrDefault(data =>
data.Guid == article.Guid);
    return data?.IsRead ?? true;
}
//создаем локальные сохранения если их нет
private EncyclopediaArticlesData CreateDefaultData()
{
    var encyclopediaArticlesData = new
EncyclopediaArticlesData();

    foreach (var constantArticle in _constantArticles)
    {
        encyclopediaArticlesData.Dates.Add(new
EncyclopediaArticleData
        {
            Guid = constantArticle.AssetGUID,
            IsRead = false,
        });
    }

    return encyclopediaArticlesData;
}

private void OnChangeGameEpoch(int epochId)
{
    _currentEpochId = epochId;
}

```

```

//загружаем статьи для выбранной эпохи (Age)
private async UniTask LoadAgeArticlesAsync(Age age)
{
    ReleaseCurrentArticles();

    if (!_ageToAddressableLabel.TryGetValue(age, out
var label))
    {
        return;
    }

    using (ListScope<object>.Create(out var keys))
    {
        keys.Add(label);
        var result = await
AddressableManager.Instance.TryShowDownloadDialogAsync(keys,
CancellationToken.None);

        if (result)
        {
            await LoadArticlesAsync(label);
        }
    }
}
//Загружаем статьи из Addressable банлов по тегу
private async UniTask LoadArticlesAsync(string label)
{
    try
    {
        var (handle, articles) =
            await
AddressableManager.Instance.LoadAllAssetsAsync<EncyclopediaArticleSO>(label);

        _currentLoadedArticles.AddRange(articles);

        _currentLoadHandles.Add(handle);
    }
    catch
    {
        Debug.Log($"Not found articles in addressable
with label: {label}");
    }
}
//освобождаем память от загруженных статей
private void ReleaseCurrentArticles()
{
    if (_currentLoadedArticles.Count == 0)
    {
        return;
    }
}

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		83

```

        _currentLoadedArticles.Clear();

        foreach (var handle in _currentLoadHandles)
        {
            handle.Release();
        }

        _currentLoadHandles.Clear();
    }

    #region UI
    //при показе окна загружаем статьи для выбранной эпохи
    private void OnShowUI(BaseWindow window)
    {
        if (window is not EncyclopediaUI)
        {
            return;
        }

        SelectAgeAsync(GetCurrentGameEpoch()).Forget();
    }

    private void OnCloseUI(BaseWindow window)
    {
        if (window is not EncyclopediaUI)
        {
            return;
        }

        ReleaseCurrentArticles();
    }

    private void FillUI()
    {
        OnFillArticleList?.Invoke(GetCurrentArticles(_categorySelected));
    }
    //получаем все статьи из категории (при переключивании
    вкладок сбоку)
    private IEnumerable<(EncyclopediaArticleSO article,
    bool isRead)> GetCurrentArticles(Category category)
    {
        if (_currentLoadedArticles == null)
        {
            return Enumerable.Empty<(EncyclopediaArticleSO
    article, bool isRead)>();
        }

        return _currentLoadedArticles

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		84

```

        .Where(loadedArticle => IsUnlocked(loadedArticle) &&
loadedArticle.Age == _ageSelected &&
loadedArticle.Category
== category)
        .Select(article => (article,
IsReaded(article)));
    }

    #endregion

    #region ValueChange
    //загружаем статьи при переключении эпох
    private async UniTask SelectAgeAsync(int ageTab)
    {
        _ageSelected = (Age)ageTab;

        OnChangeDownloadArticlesState?.Invoke(false);

        await LoadAgeArticlesAsync(_ageSelected);

        OnChangeDownloadArticlesState?.Invoke(true);

        OnAgeSelected?.Invoke((int)_ageSelected);
        OnSelectCategory(0);
    }
    //отслеживаем переключение эпох
    private void OnChangeAgeTab(int nextAge)
    {
        var age = (int)_ageSelected + nextAge;
        var values = _ageEnumValues.Value;

        if (age > values.Max())

age = values.Min();
        else if (age < values.Min())
            age = values.Max();

        SelectAgeAsync(age).Forget();
    }
    //отслеживаем выбор категорий

    private void OnSelectCategory(Category category)
    {
        _categorySelected = category;
        FillUI();
    }
    //при выборе статьи загружаем ее информацию для
отображения

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		85

```

private void OnSelectArticle(EncyclopediaArticleSO article)
{
    _articleSelected = article;
    if (_articleSelected != null)
    {
        var data = Data.Dates.FirstOrDefault(i =>
i.Guid == article.Guid);
        if (data is { IsRead: false })
        {
            data.IsRead = true;
            _dataController.SaveData();

            OnArticleIsRead?.Invoke(article);
        }
    }

    OnFillArticlData?.Invoke(article);
}
//получаем текущую эпоху
private int GetCurrentGameEpoch()
{
    var epochId = _currentEpochId;
    if (epochId == 0)
        epochId = QuestManager.MASTER_EPOCH_ID;

    if (!_questLineIdToAge.TryGetValue(epochId, out var
epochAge))
    {
        Debug.LogError($"Not found Age for epoch with
ID: {epochId}");
        return (int)Age.Master;
    }

    return (int)epochAge;
}

#endregion
}
}

```

Файл *EncyclopediaUI.cs*

```

using System.Collections.Generic;
using System.Linq;
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using VKT.Messages;
using VKT.Windows;

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		86

```

namespace VKT.Encyclopedia
{
    public class EncyclopediaUI : BaseWindow
    {
        const string AGE_DEFAULT_NAME = "ЭПОХА";

        [SerializeField] private CanvasGroup
_parentCanvasGroup;

        [Header("Prefab: Article")]

        [SerializeField] private EncyclopediaButton
articleButtonPrefab;

        [Header("Button: Age")]

        [SerializeField] private Button prevAge;
        [SerializeField] private Button nextAge;
        [SerializeField] private TMP_Text titleAge;
        [SerializeField] private List<string> ageTitles;

        [Header("Button: Containers")]

        [SerializeField] private GameObject categoryContainer;
        [SerializeField] private GameObject articleContainer;
        [SerializeField] private ToggleGroup
articlesToggleGroup;

        [Header("Button: Data")]

        [SerializeField] private Button closeButton;
        private List<EncyclopediaButton> categoryButtons = new
List<EncyclopediaButton>();

        private List<EncyclopediaButton> articleButtons = new
List<EncyclopediaButton>();

        [SerializeField] private bool stopSpeechOnClose;

        #region MONO
        //ПОДПИСЫВАЕМСЯ НА СОБЫТИЯ МЕНЕДЖЕРА ЭНЦИКЛОПЕДИИ,
чтобы заполнять визуал
        protected override void OnEnable()
        {
            base.OnEnable();
            EncyclopediaManager.OnAgeSelected += AgeSelected;
            EncyclopediaManager.OnFillArticleList +=
OnFillArticles;
            EncyclopediaManager.OnArticleIsRead +=
OnArticleIsRead;

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

```

EncyclopediaManager.OnChangeDownloadArticlesState +=
OnChangeDownloadArticlesState;

        _parentCanvasGroup.alpha = 0;
    }

    protected override void OnDisable()
    {
        EncyclopediaManager.OnAgeSelected -= AgeSelected;
        EncyclopediaManager.OnFillArticleList -=
OnFillArticles;
        EncyclopediaManager.OnArticleIsRead -=
OnArticleIsRead;
        EncyclopediaManager.OnChangeDownloadArticlesState -
= OnChangeDownloadArticlesState;
        base.OnDisable();
    }
    public override void Close()
    {

        if (stopSpeechOnClose)
            new EndDialogMessage().Publish();

        base.Close();
    }

    #endregion
    //при выборе эпохи смотрим что дошли до конца и
выключаем возможность крутить дальше
    private void AgeSelected(int index)
    {
        var toggle = categoryButtons[0].Toggle;
        if (toggle.isOn)
            toggle.isOn = false;
        toggle.isOn = true;
        AgeTitleUpdate((Age)index);
    }
    //при прочтении статьи вырубам значек оповещения о
непрочитанной статьи

    private void OnArticleIsRead(EncyclopediaArticleSO
article)
    {
        SetCategoryNotify(article.Category,
EncyclopediaManager.Instance.IsHaveUnreadInCategory(article.Category
));
    }
    //включаем возможность кликать после загрузки всех
статей эпохи
    private void OnChangeDownloadArticlesState(bool state)
    {

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		88

```

        _parentCanvasGroup.interactable = state;
    }
    //включаем или выключаем иконку непрочитанной статьи

    private void SetCategoryNotify(Category category, bool flag)
    {
        categoryButtons[(int)category].SetNotifyState(flag);
        print(category + " " + flag);
    }

    public override void Initialize()
    {
        base.Initialize();
        InitAgeButtons();
        InitCategoryButtons();
    }

    public void InitAgeButtons()
    {
        prevAge.onClick.AddListener(() =>
EncyclopediaManager.OnAgeTabChanged?.Invoke(-1));
        nextAge.onClick.AddListener(() =>
EncyclopediaManager.OnAgeTabChanged?.Invoke(1));
    }
    //при выборе эпохи обновляем заголовков
    private void AgeTitleUpdate(Age age)
    {
        if ((int)age >= ageTitles.Count)
        {
            titleAge.text = AGE_DEFAULT_NAME;

            return;
        }

        titleAge.text = ageTitles[(int)age];
    }

    //включаем доступные кнопки категорий
    private void InitCategoryButtons()
    {
        for (int i = 0; i <
categoryContainer.transform.childCount; i++)
        {
            var categoryButton =
categoryContainer.transform.GetChild(i).gameObject.GetComponent<Ency
clopediaButton>();

            var index = i;

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		89

```

        categoryButton.Toggle =
categoryButton.GetComponentInChildren<Toggle>();

categoryButton.Toggle.onValueChanged.AddListener(value=>
    {
        if (value)

EncyclopediaManager.OnCategorySelected((Category)index);
    });
categoryButtons.Add(categoryButton);
    }
}
//обновляем визуал при загрузке статей
private void
OnFillArticles(IEnumerable<(EncyclopediaArticleSO article, bool
isRead)> articles)
{
    RefreshCategoriesNotifiers();

    InitArticlesButtons(articles);

    _parentCanvasGroup.alpha = 1;
}
//включаем кнопки для взаимодействия со статьями

private void
InitArticlesButtons(IEnumerable<(EncyclopediaArticleSO article, bool
isRead)> articles)
{
    var count = articles.Count();
    int difference = count - articleButtons.Count;
    if (difference > 0)
    {
        for (int i = 0; i < difference; i++)
        {
            var item =
Instantiate(articleButtonPrefab.gameObject,
articleContainer.transform)
                .GetComponent<EncyclopediaButton>();
            articleButtons.Add(item);
            item.Toggle = item.GetComponent<Toggle>();
            item.Toggle.group = articlesToggleGroup;
        }

        difference = 0;
    }
    else
        difference = Mathf.Abs(difference);

```

					ПГУ1.090301.13.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

```

        difference = articleButtons.Count - difference;
        for (int i = 0; i < articleButtons.Count; i++)
            articleButtons[i].gameObject.SetActive(i <
difference);

        if (count == 0 || articleButtons.Count == 0)
        {
EncyclopediaManager.OnArticleSelected?.Invoke(null);
            return;
        }

        var index = 0;
        foreach (var (article, isRead) in articles)
        {
            articleButtons[index].Init(article, isRead);
            var toggle = articleButtons[index].Toggle;
            toggle.onValueChanged.RemoveAllListeners();
            toggle.onValueChanged.AddListener(value =>
            {
                if (value)

EncyclopediaManager.OnArticleSelected(article);
            });

            index++;
        }

        articleButtons[0].Toggle.isOn =
!articleButtons[0].Toggle.isOn;
    }
    //обновляем значки прочитано/непрочитано
    private void RefreshCategoriesNotifiers()
    {
        for (var i = 0; i < categoryButtons.Count; i++)
        {
            var category = (Category)i;
            SetCategoryNotify(category,
EncyclopediaManager.Instance.IsHaveUnreadInCategory(category));
        }
    }
}

```

Файл *EncyclopediaArticle.cs*

```

using Spine.Unity;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		91

```

namespace VKT.Encyclopedia
{
    public class EncyclopediaArticle : MonoBehaviour
    {
        public EncyclopediaArticleSO Article { get; private
set; }
        [field: SerializeField] public TMP_Text Title { get;
private set; }
        [field: SerializeField] public TMP_Text Subtitle { get;
private set; }
        [field: SerializeField] public TMP_Text Description {
get; private set; }
        [field: SerializeField, TextArea] public string
DefaultDescription { get; private set; }
        [field: SerializeField] public Image Picture { get;
private set; }
        [field: SerializeField] public SkeletonGraphic
AnimationSpine { get; private set; }
        [field: SerializeField] public GameObject SpeakerButton
{ get; private set; }
        [SerializeField] private Sprite defaultPicture;
        public EncyclopediaButtonPlaySpeaker PlaySpeaker { get;
private set; }
        public AudioClip SoundClip { get; private set; }
        //подписываемся на событие обновления информации о
статье(выборе статьи)
        public void OnEnable()
        {
            EncyclopediaManager.OnFillArticleData += Init;

            PlaySpeaker =
SpeakerButton.GetComponent<EncyclopediaButtonPlaySpeaker>();
        }
        public void OnDisable()
        {
            EncyclopediaManager.OnFillArticleData -= Init;
        }
        //Инициализируем визуал статьи
        public void Init(EncyclopediaArticleSO articleSO)
        {
            Article = articleSO;
            if (Article == null)
            {
                Clear();
                return;
            }
            Title.text = Article.Title;
            if (!string.IsNullOrEmpty(Article.Subtitle))
            {
                Subtitle.gameObject.SetActive(true);
            }
        }
    }
}

```

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		92

```

        Subtitle.text = Article.Subtitle;
    }
    else
        Subtitle.gameObject.SetActive(false);

    Description.text = Article.Description;
    if (Article.IsAnimatedPicture)
    {
        AnimationSpine.gameObject.SetActive(false);
        AnimationSpine.Clear();
        AnimationSpine.skeletonDataAsset =
Article.SkeletonAsset;

        AnimationSpine.startingAnimation = Article.AnimationName;
        AnimationSpine.transform.localScale =
Article.ScaleOverriding;
        AnimationSpine.initialSkinName = Article.SkeletonSkin;
        AnimationSpine.Initialize(true);
        AnimationSpine.gameObject.SetActive(true);
        Picture.gameObject.SetActive(false);
    }
    else
    {
        Picture.sprite = Article.Sprite != null ?
Article.Sprite : defaultPicture;
        AnimationSpine.gameObject.SetActive(false);
        Picture.gameObject.SetActive(true);
    }
    AudioClip = Article.SoundClip;
    if (SoundClip == null)
    {
        SpeakerButton.SetActive(false);
        return;
    }

    SpeakerButton.SetActive(true);
    PlaySpeaker.Init(this);
}
//очищаем визуал , если статьи не найдено
public void Clear()
{
    if (gameObject.activeInHierarchy == false) return;
    Title.text = default;
    Description.text = DefaultDescription;
    AnimationSpine.transform.localScale = Vector3.one;
    if (AnimationSpine.gameObject.activeInHierarchy)
{

```

					ПГУ1.090301.13.001 ПЗ	Лист
						93
Изм.	Лист	№ докум.	Подпись	Дата		

```

AnimationSpine.gameObject.SetActive(false);
Picture.gameObject.SetActive(true);
    }

    Subtitle.text = default;
    Picture.sprite = defaultPicture;
    SoundClip = default;
    SpeakerButton.SetActive(false);
}
}
}

```

					ПГУ1.090301.13.001 ПЗ	Лист
						94
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательное)

Образовательное приложение «Магия знаний»

Презентация

					ПГУ1.090301.13.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

Тема работы:
«Образовательное приложение «Магия знаний»»

Автор работы: студент группы 19ВВП1 Макаров Д.Д.

Научный руководитель: к.т.н., доцент каф. ВТ ПГУ Карамышева Н.С.

Пенза 2023

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		96

Актуальность работы

Данная тема актуальна благодаря своему сочетанию игры и обучения. Пользователям предстоит решать различные задачи, проходить тесты и отвечать на вопросы, которые помогут преодолеть препятствия.

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		97

Предмет и объект работы

Предмет

Технологии разработки образовательных игровых приложений, в жанре визуальная новелла.

Объект

Объектом работы является разработка образовательного приложения на Unity

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		98

Цель работы и постановка задачи

Цель работы

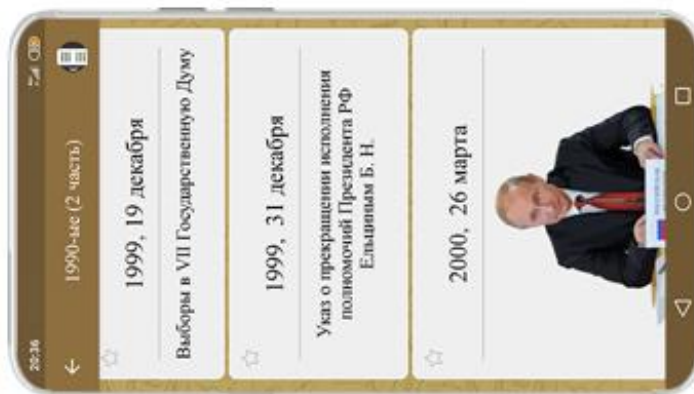
- Целью является разработка образовательного приложения, в жанре визуальной новеллы, которое в занимательной форме расскажет об основных периодах и событиях истории России.

Постановка задачи

Необходимо решить следующие задачи:

- приложение должно сочетать в себе элементы игры и обучения;
- предусмотреть сохранение полученной информации во внутриигровой энциклопедии;
- создать версии как для мобильных, так и для стационарных устройств;
- создать систему диалогов;
- корректная работа на каждой платформе.

Аналоги



«История России. Даты»

Аналоги



«История России и Мира. ЕГЭ»

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		101

История для всех.
Кроссворды

По горизонтали

1. Римская Армия
3. Колонное вспомогательное подразделение римской армии
5. Столица Персии, сожженная Александром Великим
7. Автор трагедии "Перикл"
9. Жестокое восточное правление в Риме
11. Поблизости от битвы при Мантуйе
13. Здание для проведения публичных состязаний в Афинах

По вертикали

2. Город, в котором стояло "чудо света" - маяк Александрии
4. Главный храм Египта
6. Должностное лицо в Риме
8. Город в Индии
10. Грек, математик
12. Гладиатор, борющийся с животными
14. Мен римского легионера
16. Богиня Артемиды в Риме
18. Афинский законодатель
20. Римский оратор

А Б В Г Д Е Ж З И Й
К Л

← Найдено слов: 3 / 8

Найдите 8 городов Древней Греции

Ч	Ц	Ф	И	А	Г	А	З
Э	П	И	Д	А	В	Р	К
С	Й	В	Е	Ж	Р	Г	О
П	Е	Ы	Л	В	И	О	Р
А	Д	Г	Ь	Б	Е	С	И
Р	Х	А	Ф	И	Н	Ы	Н
Т	У	Я	Ы	П	Ю	Э	Ф
А	М	И	К	Е	Н	Ы	Х

← Викторина: 1/10 ● 3

Во время морского сражения у острова Саламин, в котором персидский флот потерпел сокрушительное поражение от греческого флота, Ксеркс, наблюдая за битвой с холма, отметил действия Артемисии следующими словами:

- «Мужчины у меня превратились в женщин, а женщины стали мужчинами»

Верно ли, что Ксеркс такотреагировал, увидев уничтожение кораблем Артемисии флагманского корабля лакедемонян?

ВЕРНО НЕВЕРНО

13.001 ПЗ	Лист
	102

Аналоги



«ЕГЭ История 2023»

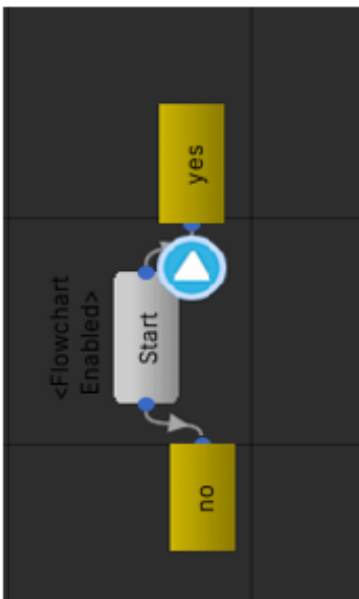
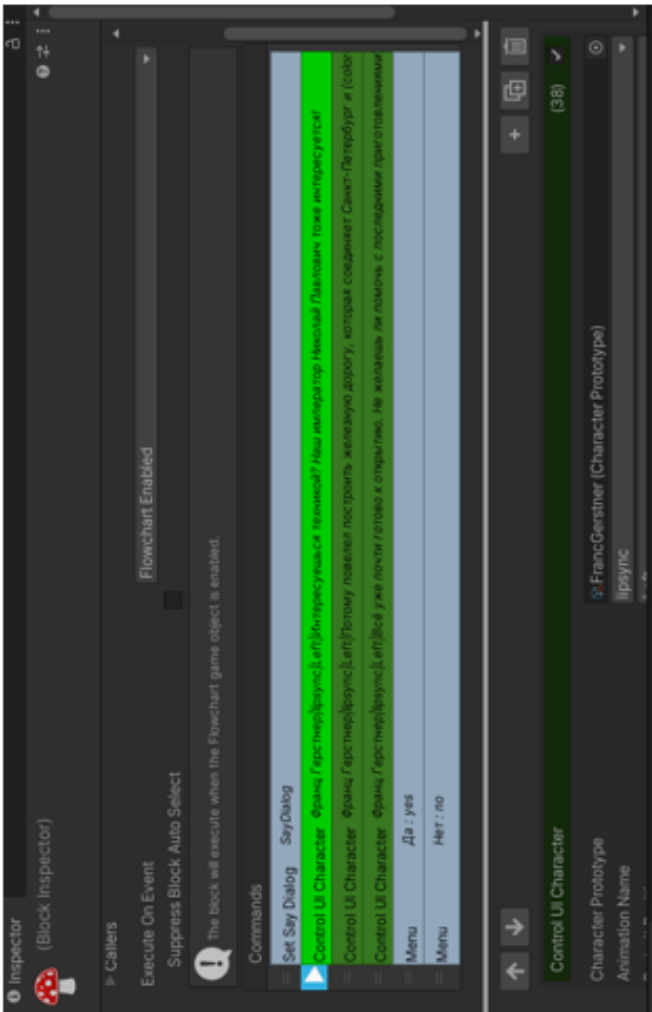
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.13.001 ПЗ

Стек технологий



Разработка системы диалогов



					ПГУ1.090301.13.001 ПЗ	Лист
						105
Изм.	Лист	№ докум.	Подпись	Дата		

Разработка внутриигровой энциклопедии



Диаграмма классов

EncyclopediaManager
<pre>- _unlockListeners: IList<IUnlockListener>[] - _constantArticles: AssetReference<EncyclopediaArticleSO>[] - _ageToAddressableLabel: SerializedDictionary<Age, string> - _questLineIdToAge: SerializedDictionary<int, Age> - _ageNumValues: Lazy<int[]> - _ageSelected: Age - _categorySelected: Category - _articleSelected: EncyclopediaArticleSO - _currentEpochId: int - _dataController: SaveDataController<EncyclopediaArticlesData> - _currentLoadedHandles: List<AsyncOperationHandle> - _currentLoadedArticles: List<EncyclopediaArticleSO> - _articleClients: HashSet<IArticleClient> - _disposables: IDisposable[] - _Data: EncyclopediaArticlesData + SyncConstantArticles() + SyncEpochsID() + GetUnreadArticlesCount() + IHaveUnreadInCategory(category: Category) + IsUnlocked<T>(<T> reference: AssetReference<T>) + IsUnlocked<T>(<T> reference: AssetReference<T>) + LoadAllArticlesAsync(client: IArticleClient) + ReleaseArticleClient(client: IArticleClient) - OnEnable() - OnDisable() - OnMessage(message: WindowOpenedMessage) - OnMessage(message: WindowClosedMessage) - OnMessage(message: EpochChangedMessage) - OnUnlockNewArticleAsync<T>(<T> reference: AssetReference<T>) - IsReaded(article: EncyclopediaArticleSO) - CreateDefaultData() - OnChangeGameEpoch(epochId: int) - LoadAgeArticlesAsync(age: Age) - LoadArticlesAsync(label: string) - ReleaseCurrentArticles() - OnShowUI(window: BaseWindow) - OnCloseUI(window: BaseWindow) - FillUI() - GetCurrentArticles(category: Category) - SelectAgeAsync(agerTab: int) - OnChangeAgerTab(nextAge: int) - OnSelectCategory(category: Category) - OnSelectedArticle(article: EncyclopediaArticleSO) - GetCurrentGameEpoch()</pre>

EncyclopediaUI
<pre>- AGE_DEFAULT_NAME: string - _parentCanvasGroup: CanvasGroup - articleButtonPrefab: EncyclopediaButton - prevAge: Button - nextAge: Button - titleAge: TMP_Text - ageTitles: List<string> - categoryContainer: GameObject - articleContainer: GameObject - articlesToggleGroup: ToggleGroup - closeButton: Button - categoryButtons: List<EncyclopediaButton> - articleButtons: List<EncyclopediaButton> - stopSpeechOnClose: bool + Initialize() + InitAgeButtons() + Close() + OnClick() + #OnDisable() + AgeSelected(index: int) - OnArticleRead(article: EncyclopediaArticleSO) - OnChangeDownloadArticlesState(state: bool) - SetCategoryNotify(category: Category, flag: bool) - AgeTitleUpdate(age: Age) - InitCategoryButtons() - OnFillArticles(article: IEnumerable<EncyclopediaArticleSO, isRead: bool>) - InitArticlesButtons(article: IEnumerable<EncyclopediaArticleSO, isRead: bool>) - RefreshCategoriesNotifiers()</pre>

EncyclopediaArticle
<pre>+ Article: EncyclopediaArticleSO + Title: TMP_Text + Subtitle: TMP_Text + Description: TMP_Text + DefaultDescription: string + Picture: Image + AnimationSpine: SkeletonGraphic + SpeakerButton: GameObject + PlaySpeaker: EncyclopediaButtonPlaySpeaker + SoundClip: AudioClip - defaultPicture: Sprite + OnEnable() + OnDisable() + Init(articleSO: EncyclopediaArticleSO) + Clear()</pre>

Диаграмма развертывания

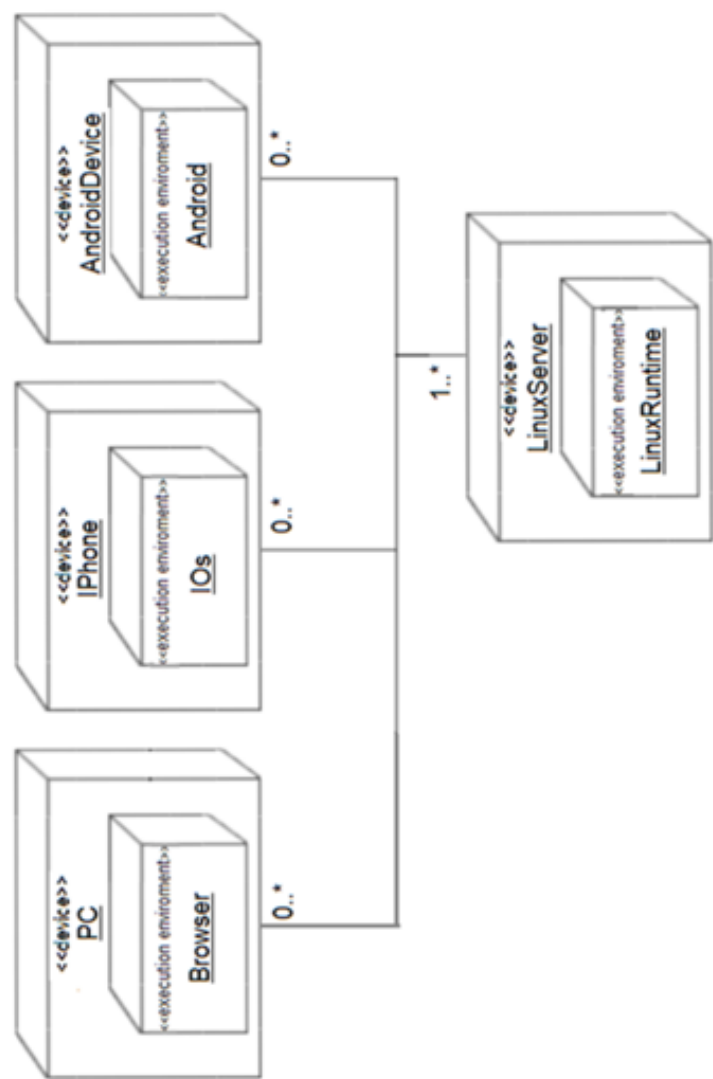
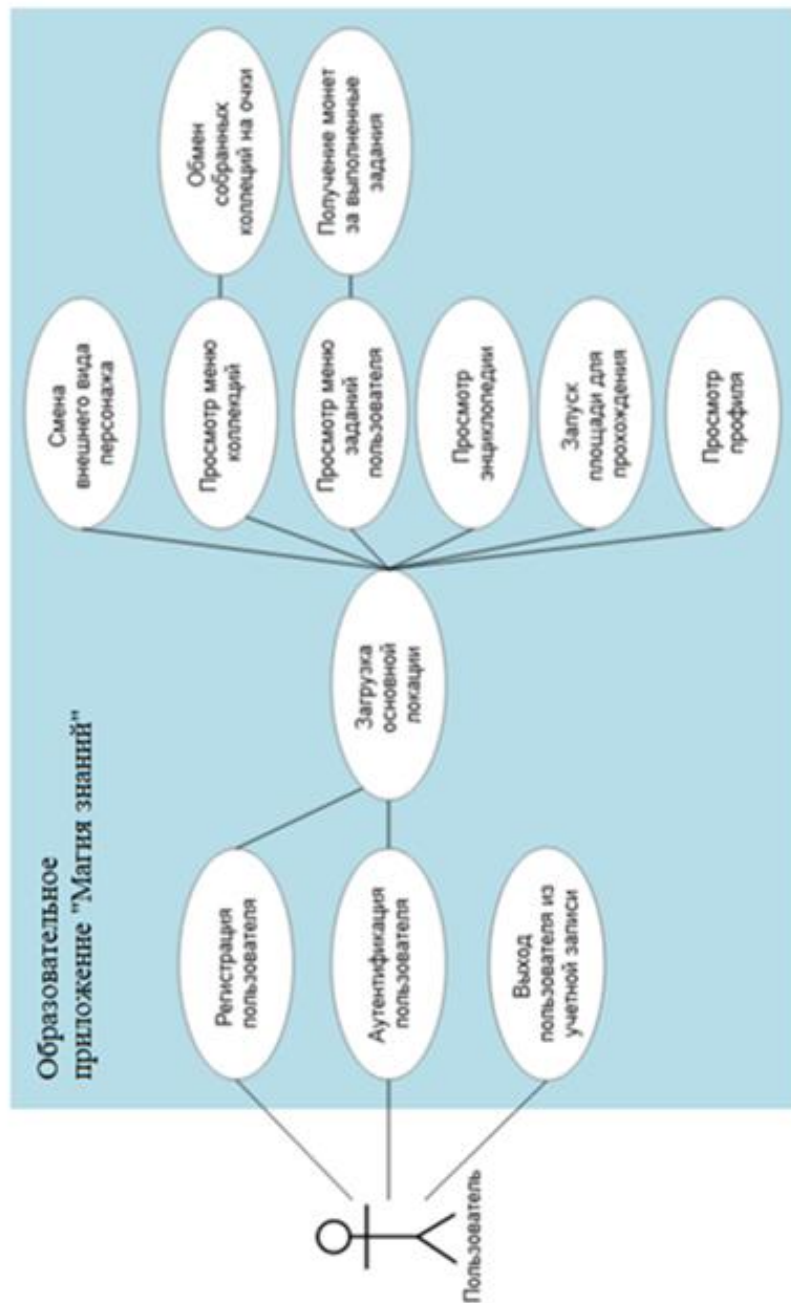


Диаграмма прецедентов



Тестирование и отладка

Площадь Александра Невского				
Описание проблемы	Шаги воспроизведения	Ожидаемый результат	Фактический результат	
Не освоил котфейер в игре располони войска (задание 5.2)	Перейти в задание 5.2 Кликнуть на подсказку			
При нажатии в окне игры обрз кадрами блок				
Площадь Александра Невского				
Описание проблемы	Шаги воспроизведения	Ожидаемый результат	Фактический результат	
Не освоил котфейер в игре располони войска (задание 5.2)	Перейти в задание 5.2 Кликнуть на подсказку			

Заключение

- В ходе выполнения выпускной квалификационной работы было разработано образовательное приложение «Магия знаний» на движке Unity, с использованием языка C#, как основного языка программирования. Диалоговая система реализована с использованием инструмента Fungus.
- Разработанное приложение позволяет изучать историю нашей страны с древнейших времен по настоящее время.
- Одним из достоинств разработанного приложения является наличие энциклопедии и системы диалогов с анимированными персонажами, а также наличие версий для всех основных платформ и их корректная работа.

Демонстрация



Сыграть в «Магию знаний» можно по ссылке magiyananiy.rf или скачать на странице в Rustore!

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.13.001 ПЗ

Лист

112

Спасибо за внимание!

					ПГУ1.090301.13.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		113

