МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра дискретной математики и информационных технологий

РАЗРАБОТКА ОБУЧАЮЩЕГО ПРИЛОЖЕНИЯ «LATEX _ LEARN»

КУРСОВАЯ РАБОТА

студентки 3 курса 321 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Ходаковской Варвары Андреевны

Научный руководитель
старший преподаватель _____ М.В. Белоконь

Заведующий кафедрой
к.ф.-м.н., доцент _____ Л.Б. Тяпаев

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время публикуется огромное количество научных работ, где требуется использовать специализированные средства набора формул и текста. Одним из наиболее популярных инструментов для этой цели является LaTeX.

LaTeX был создан в 1980 году для подготовки научных статей и диссертаций Дональдом Кнутом – профессором информатики в Университете Стэнфорда. Кнут начал работать над TeX (предшественником LaTeX) в 1977 году, но понял, что для многих пользователей он слишком сложен. Так он начал разработку более простого для использования пакета макрокоманд, который в последующем был назван LaTeX.

Данное программное обеспечение представляет собой набор макрорасширений TeX, который позволяет удобно и быстро создавать качественные тексты и формулы, в том числе и научные работы, статьи, диссертации и пр. Однако, для начинающих пользователей LaTeX может показаться достаточно сложным и непонятным.

В рамках данной курсовой работы будет разработано обучающее приложение «LaTeX _ Learn», оно будет представлять собой обучающий курс, который поможет овладеть базовыми знаниями LaTeX и научиться эффективно работать с данным инструментом, а также закрепить пройденный материал в викторине.

В качестве движка была выбрана платформа Unity. Это один из наиболее популярных инструментов, обеспечивающий возможность создания обучающих игр с высоким уровнем интерактивности и графическим интерфейсом пользователя.

Целью курсовой работы является разработка обучающего приложения «LaTeX _ Learn».

Данная тема актуальна, потому что на рынке нет мобильных приложений, обучающих LaTeX. Создание такого приложения поможет упростить процесс изучения, позволит пользователям быстро освоить необходимые навыки.

Для достижения цели работы были поставлены следующие задачи:

— Изучение возможностей движка Unity и его основных принципов работы.
— Разработка концепции обучающего приложения.
— Изучение Adobe Photoshop для создания дизайна.
— Написание программного кода на языке C#.
— Создание игровых механик, графики, анимаций.

— Аутентификация пользователей с помощью Firebase.
— Создание базы данных с помощью Firebase.
— Тестирование и отладка приложения.

1 Необходимая теория

1.1 Назначение LaTeX

LaTeX представляет собой систему компьютерной вёрстки для создания научных документов, написания книг, а также многих других форм публикаций. В его основе лежит парадигма редактирования WYSIWYG («что видишь, то и получишь»), то есть от пользователя требуется сосредоточиться только на содержимом документа, оставив его форматирование программе.

LaTeX позволяет не только создавать красиво оформленные документы, но также дает пользователям возможность очень быстро реализовывать такие сложные элементы печатного набора, как математические выражения, таблицы, ссылки и библиографии, получая согласованную разметку по всем разделам.

Благодаря доступности большого числа открытых библиотек возможности LaTeX становятся практически безграничны. Эти библиотеки расширяют возможности пользователей еще больше, позволяя добавлять сноски, рисовать схемы и прочее [1].

1.2 Основные преимущества LaTeX

1. Удобство использования: LaTeX предоставляет широкий набор инструментов для создания документов, включая таблицы, списки, формулы, изображения и многое другое. Кроме того, существует множество пакетов, которые позволяют настраивать внешний вид документа в соответствии с требованиями.
2. Высокая точность верстки: LaTeX обеспечивает точность и аккуратность верстки благодаря использованию математических формул, таблиц, списков и других элементов. Это позволяет избежать ошибки и улучшить читаемость документа.
3. Совместимость: LaTeX поддерживает множество форматов файлов, включая PDF, HTML и XML. Это делает его совместимым с различными программами и устройствами.
4. Расширяемость: LaTeX имеет открытый исходный код, что позволяет разработчикам создавать свои собственные пакеты и расширения для улучшения функциональности системы.
5. Поддержка научных и академических кругов: LaTeX широко используется в научных и академических кругах для создания статей, книг, диссертаций

и других научных работ.

1.3 Недостатки LaTeX

1. Сложность изучения. LaTeX требует некоторого времени и усилий для изучения и освоения всех возможностей системы.
2. Необходимость использования специальных инструментов. Для создания и редактирования документов в LaTeX необходимо использовать специальные программы, такие как TeXMaker или LaTeXWorks.
3. Ограниченность в использовании. Некоторые функции LaTeX не поддерживаются на таких устройствах, как смартфоны и планшеты [2].

   Каждый год многие студенты сталкиваются с трудностями при оформлении академических и научных работ в LaTeX, особенно новички. Поэтому было решено создать обучающее приложение на Unity, чтобы облегчить процесс изучения.

1.4 Среда разработки Unity

Unity – это кроссплатформенная среда разработки, которая позволяет создавать игры и приложения на разных платформах. Она была разработана компанией Unity Technologies и стала популярной благодаря своей простоте использования и возможности создавать игры разных жанров [3].

Эта платформа была выбрана, потому что она имеет множество плюсов:

1. Широкий выбор инструментов и функций: Unity предлагает широкий набор инструментов и функций, которые позволяют разработчикам создавать игры с высокой степенью контроля и гибкости.
2. Кроссплатформенность: Unity поддерживает создание игр для различных платформ, таких как Windows, macOS, Linux, Android и iOS. Это означает, что игры, созданные на Unity, могут быть запущены на различных устройствах, таких как ПК, мобильные устройства, приставки и т.д.
3. Простота использования: Unity имеет интуитивно понятный интерфейс, который упрощает процесс создания игры. Разработчики могут легко добавлять объекты, настраивать их свойства и создавать скрипты.
4. Поддержка сообщества: Unity имеет активное сообщество разработчиков, которые делятся своими знаниями и опытом в создании игр. Это помогает получить ответы на вопросы и решить проблемы, связанные с созданием приложения.

5. Гибкость: Unity позволяет разработчикам настраивать игровой процесс, добавлять новые функции и изменять существующие компоненты игры в соответствии с требованиями проекта.
6. Большой выбор готовых компонентов: Unity предлагает большое количество готовых компонентов, которые могут быть использованы для создания игр. Это сильно упростило процесс разработки приложения и позволило сосредоточиться на создании игрового процесса и дизайна.

Разобраться в работе помогают такие обучающие сервисы, как Unity for beginner [4].

Рабочая область Unity разделена на четыре части, где пользователь организовывает свой процесс создания проекта. Во вкладке «иерархия» находится список всех основных объектов. Окно сцены позволяет разработчику увидеть, как примерно в игре будут выглядеть объекты, может отображать двухмерную или трехмерную перспективу в зависимости от типа проекта. В окне проекта располагаются все активы, задействованные в игре. Окно «инспектор» позволяет просматривать и редактировать свойства объектов. Благодаря этим инструментам разработчик реализует свою идею. Визуализация показана на рисунке 1.

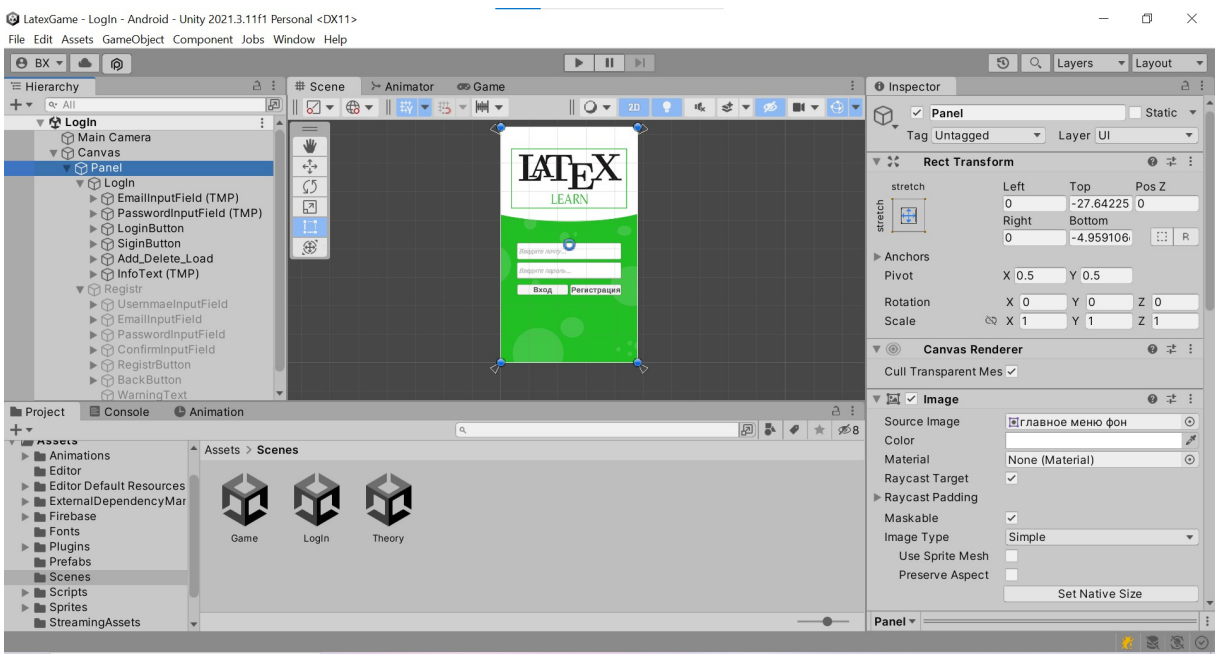

Рисунок 1 – Окно Unity

1.5 Язык программирования C#

Логика приложения, интерфейс для заполнения вопросов и вариантов ответов, переход между сценами, работа таймера, счет очков и т. д. будут реа-

лизованы с помощью языка программирования C#, в редакторе кода Microsoft Visual Studio.

C# – это объектно-ориентированный язык программирования, который был разработан Microsoft в 2000 году как часть платформы .NET Framework. Предназначен для создания различных типов приложений, включая консольные, десктопные и веб-приложения.

Язык программирования C# имеет несколько преимуществ, которые делают его очень популярным среди программистов:

- C# имеет мощный механизм сборки мусора, который облегчает работу с памятью и предотвращает утечки памяти.
- C# имеет строгую типизацию, что обеспечивает более безопасный и надежный код.
- C# имеет богатые возможности для работы с базами данных, включая поддержку LINQ.
- C# является языком, который активно развивается. Каждый год Microsoft выпускает новые версии C#, включая новые функции и возможности. C# также имеет несколько различных интегрированных сред разработки (IDE), таких как Visual Studio, SharpDevelop и MonoDevelop, которые облегчают разработку приложений на этом языке [5].

1.6 Microsoft Visual Studio

Microsoft Visual Studio – это линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения игры, так и приложения с графическим интерфейсом. Эта среда разработки обладает такими преимуществами как редактор исходного кода и отладка кода.

1.7 Firebase

Аутентификация пользователей и база данных с результатами игроков будут реализованы с помощью FireBase.

Firebase – это платформа для разработки мобильных приложений от компании Google, в которой есть самые современные функции для разработки, перекомпоновки и улучшения приложений. Некоторые из самых популярных

функций платформы Google Firebase включают в себя базы данных, аутентификацию, push-уведомления, аналитику, хранение файлов и многое другое [6].

Firebase – это очень универсальная и гибкая платформа. Она позволяет своим пользователям разрабатывать следующие категории приложений:

- Android.
- iOS.
- Web.

1.7.1 Преимущества и недостатки Firebase

Преимущества Firebase:

— Бесплатный начальный план.
— Скорость разработки.
— Сквозная платформа для разработки приложений.
— Работает на платформе Google.
— Разработчики могут сосредоточиться на фронтенде.
— Не требуется использовать сервер.
— Заложены возможности машинного обучения.
— Генерация трафика для вашего приложения.
— Мониторинг ошибок.
— Безопасность.

Недостатки Firebase:

— Нет открытого исходного кода.
— Firebase не присутствует во многих странах.
— Доступны только базы данных NoSQL.
— Медленные запросы.
— Не все службы работают бесплатно на базовом тарифе.
— Работает только в Google Cloud.
— Выделенные серверы и горпоративная техподдержка отсутствует.

1.7.2 Базы данных Firebase

Firebase включает в себя две базы данных, это Cloud Firestore и Realtime Database, которые являются полезными инструментами, удовлетворяющие современным требованиям разработки приложений.

Cloud Firestore, также известный как Google Firestore, является частью платформы разработки мобильных приложений Firebase. По сути, это облач-

ная база данных NoSQL для хранения и синхронизации данных. Пользователи Firebase могут получить доступ к базе данных Firestore через мобильное или веб-приложение собственного SDK. Cloud Firestore поддерживает различные языки программирования, такие как Unity, C++, Java, Node.js SDK, а также поддерживает REST API и RPC. База данных Firestore от Firebase заточена на обеспечение оптимальной производительности, надежности, автоматического масштабирования и удобства использования.

СУБД Firebase Realtime – это облачная база данных. Она облегчает хранение данных на основе JSON и выполняет синхронизацию данных в реальном времени с подключенными клиентами. Отдельные экземпляры базы данных функционируют как клиенты в процессе разработки кроссплатформенных приложений с использованием SDK iOS, JavaScript и Android. Это позволяет приложениям получать обновления и данные в реальном времени. В автономных приложениях данные никуда не теряются, поскольку SDK базы данных обеспечивает сохранение данных на диске. Это помогает синхронизировать устройства с серверами после восстановления подключения [7].

1.7.3 Firebase Authentication

Firebase Authentication, это функция Firebase, которая предлагает готовые к использованию библиотеки интерфейсов для пользователей, бэкенды и SDK для аутентификации пользовательских приложений. Аутентификация поддерживается с помощью телефонных номеров, паролей и поставщиков услуг, таких как Google, Twitter, Facebook и т.д. Аутентификация имеет интеграцию с различными сервисами Firebase, а также использует OpenID Connect и OAuth 2.0, позволяет выполнять бэкенд-интеграцию.

1.8 Adobe Photoshop

Графика игры будет создана в программе Adobe Photoshop.

Adobe Photoshop – это графический редактор, который используется для создания и редактирования изображений. Он был создан в 1990 году компанией Adobe Systems и с тех пор стал одним из самых популярных инструментов для работы с графикой. Одной из главных особенностей Adobe Photoshop является его широкий спектр функций. Он позволяет пользователю работать с изображениями любого размера и формата, включая фотографии, графику, логотипы и многое другое. Кроме того, Adobe Photoshop имеет множество инструментов

для обработки изображений, таких как фильтры, кисти, инструменты выделения и многие другие.

Еще одной важной функцией Adobe Photoshop является возможность работы со слоями. Слои позволяют пользователю разбивать изображение на части и работать с каждой из них отдельно. Это очень удобно при работе с большими изображениями или при создании сложных композиций.

Adobe Photoshop также имеет множество плагинов и расширений, которые позволяют добавлять новые функции и инструменты в программу [8].

К графическим объектам игры относятся окно регистрации, фон главного меню, все иконки и кнопки.

2 Разработка приложения

Современные технологии позволяют создавать многофункциональные и качественные игры, способные не только развлечь, но и обучить пользователей. Отделение игр от традиционных методов обучения и переход на новые методы, позволяющие сделать процесс обучения более интерактивным и увлекательным, приводят к увеличению популярности обучающих игр.

При разработки концепции игры было необходимо продумать идею, оформление и содержание игры.

Было решено, что приложение будет состоять из двух частей – блок теории по LaTeX и практическая часть в виде викторины. Пользователь изучает теорию по темам, а потом закрепляет знания с помощью игры. За каждый правильный ответ (время на ответ ограничено) пользователь получает определенное количество баллов, в зависимости от сложности вопроса.

Переход между сценами был реализован с помощью анимаций. Для их создания необходимо создать анимационный контроллер в Unity, который будет управлять анимацией объекта. Затем можно создать анимационные кривые, которые будут определять, как объект будет двигаться и изменяться в процессе анимации. За каждой анимацией закрепляется определенный «trigger», который в последующем будет вызываться из скрипта. Анимации помогли придать реалистичность и динамичность работе приложения.

Регистрация пользователей по Email реализована с помощью Firebase Authentication. Сначала необходимо добавить Firebase SDK в проект, далее настроить Firebase Console. Для регистрации нового пользователя используется метод CreateUserWithEmailAndPasswordAsync(). Для реализации входа в систему – SignInWithEmailAndPasswordAsync(). Для валидации почты – SendEmail VerificationAsync(). После успешной регистрации данные отобразятся в облаке Firebase, как показано на рисунке 2.

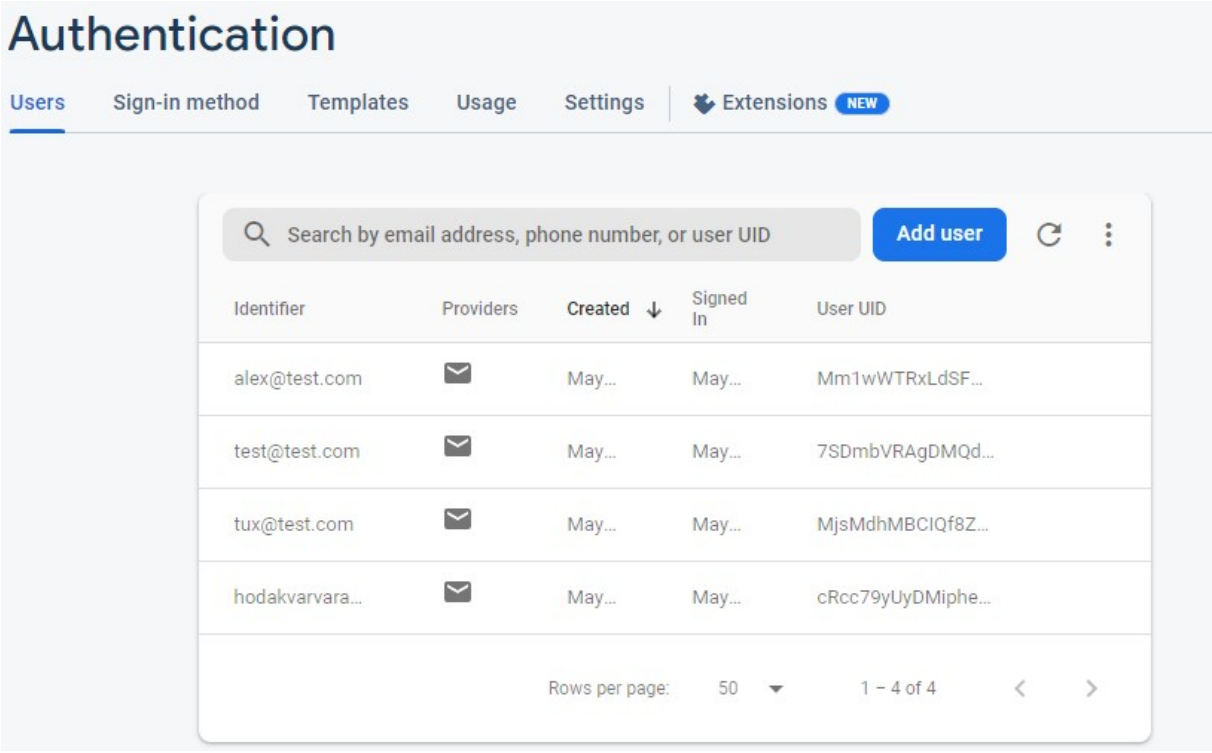

Рисунок 2 – Firebase Authentication

Сохранение результатов игроков можно реализовать с помощью Realtime Database.

Для начала, с помощью метода FirebaseDatabase.DefaultInstance.RootReference, следует получить ссылку на базу данных, а далее отправлять данные в Json формате с помощью метода SetRawJsonValueAsync(), чтобы получить данные нужно воспользоваться методом GetValueAsync(). Пример, созданной базы данных можно видеть на рисунке 3.

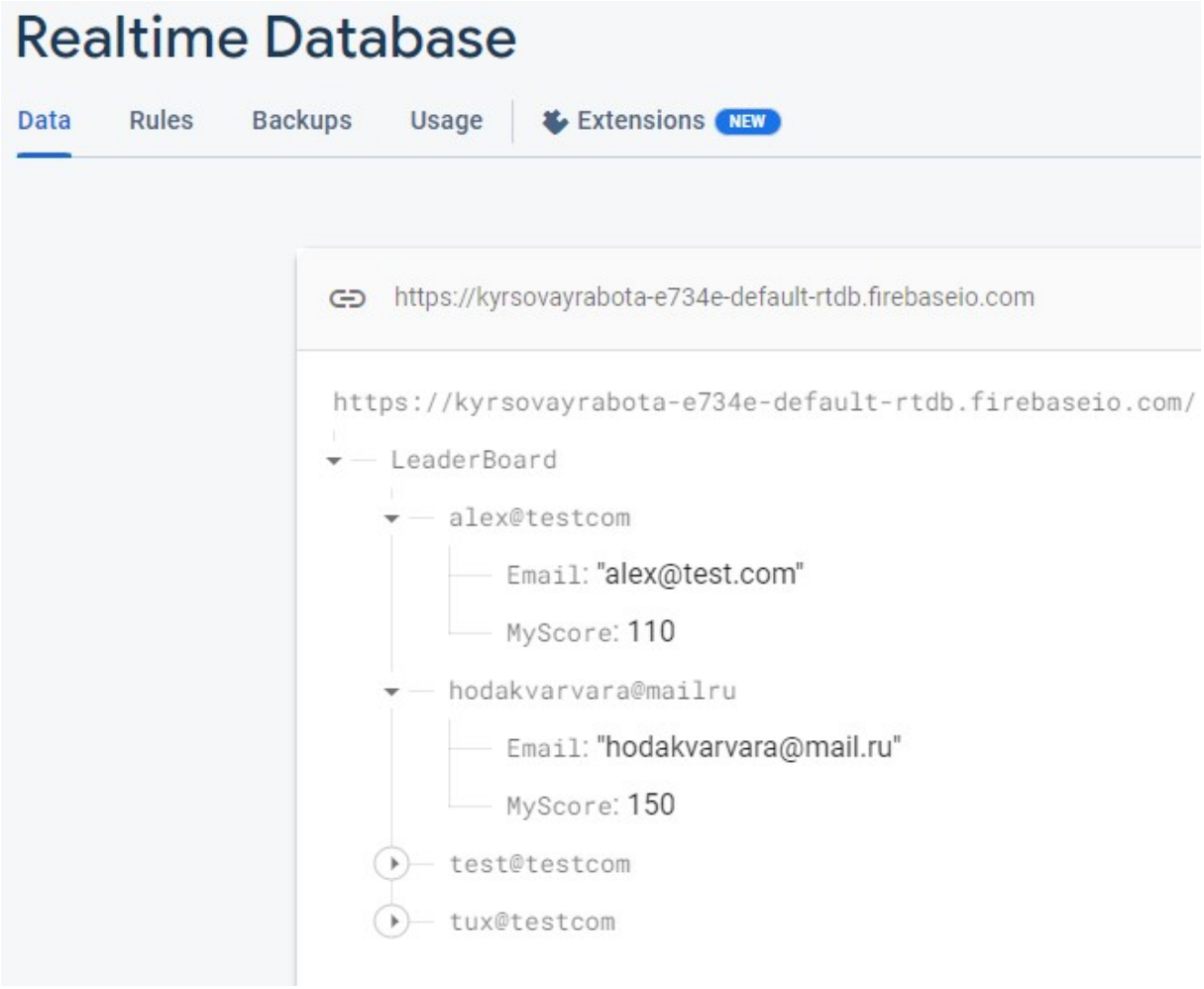

Рисунок 3 – Firebase Realtime Database

2.1 Функциональные компоненты проекта

При открытии приложения, пользователь видит сцену, на которой предлагается войти или зарегистрироваться, пример сцены представлен на рисунке 4.

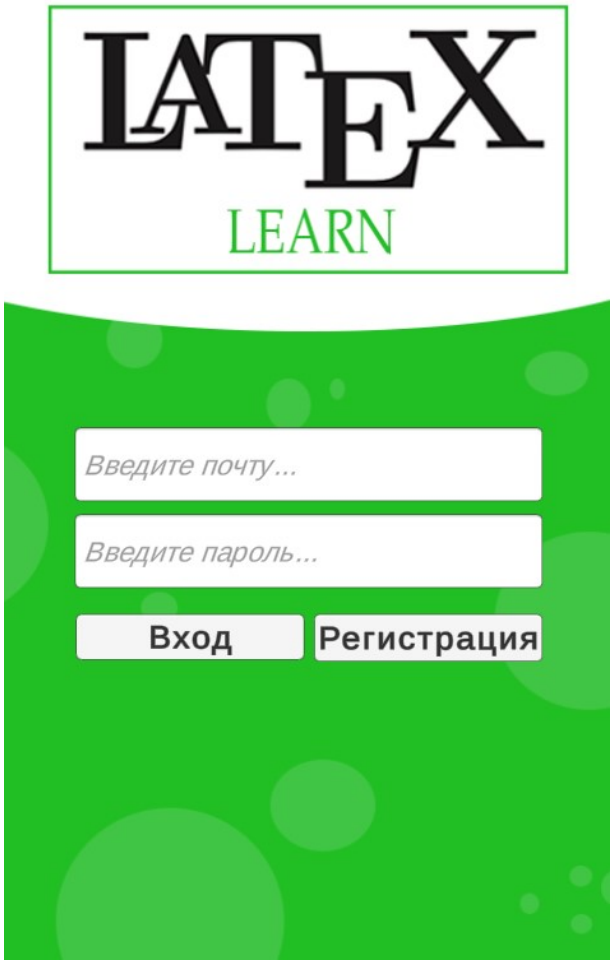

Рисунок 4 – Окно входа

При регистрации открывается окно, в котором необходимо ввести имя пользователя, почту, установить и подтвердить пароль, визуализация показана на рисунке 5.

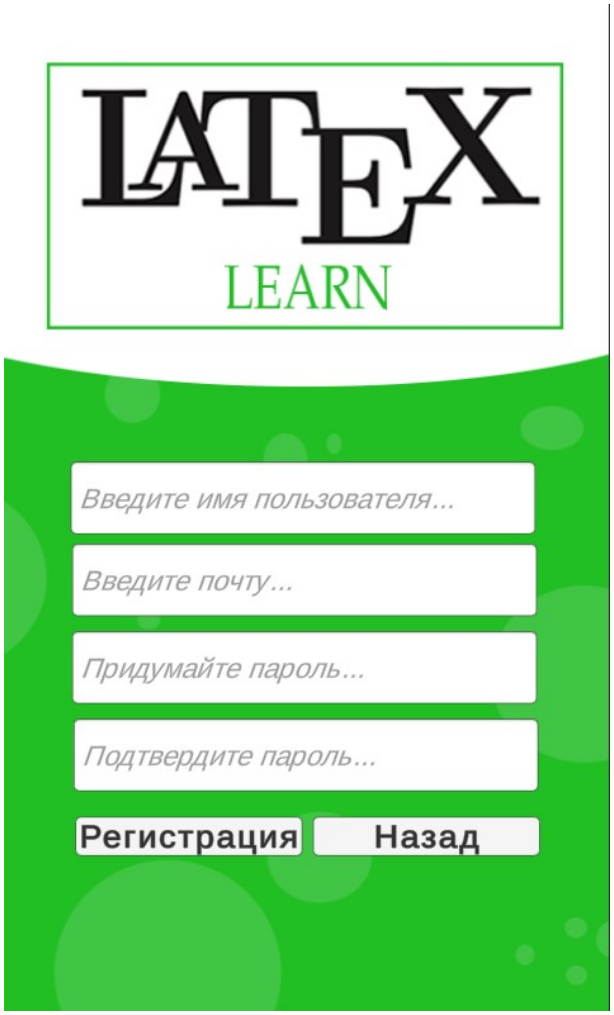

Рисунок 5 – Окно регистрации

Также реализована проверка на корректность ввода данных (все ли поля заполнены, совпадают ли пароли и т. д.), при неправильном вводе выдается сообщение об ошибке. После успешной регистрации, пользователю отправляется письмо на почту, с просьбой подтвердить регистрацию, как показано на рисунке 6.

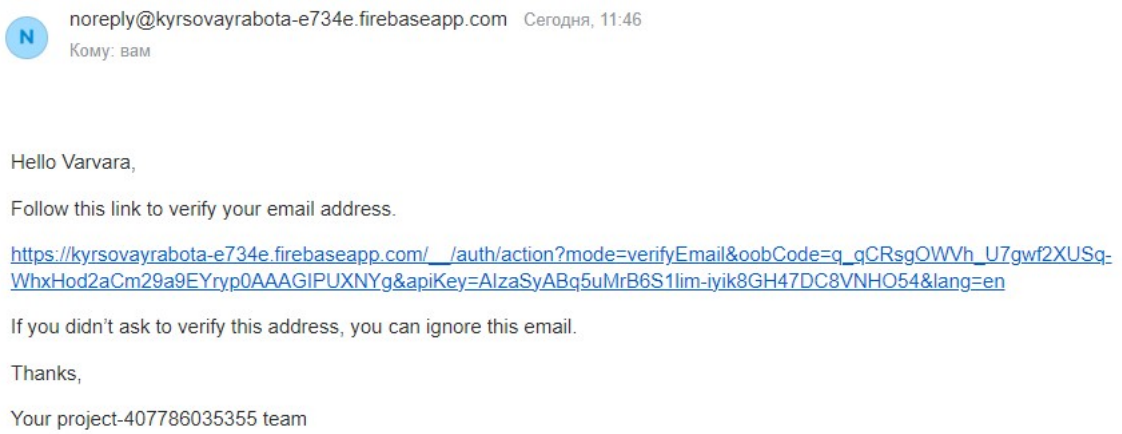

Рисунок 6 – Письмо для подтверждения регистрации

После успешного входа, открывается главное меню приложения – рисунок 7.

Рисунок 7 – Главное меню приложения

Главное меню содержит кнопки:

- Выход из аккаунта (в левом нижнем углу). При выходе все данные игрока будут сохранены.
- Рейтинг игроков (иконка с человечком). При нажатии открывается панель, на которой в порядке убывания баллов расположены игроки и их рекорды. Визуализация показана на рисунке 8.

Рисунок 8 – Рейтинг пользователей

- Теория. При нажатии переходим на сцену, содержащую разделы теории, в соответствии с рисунком 9.
  Всего 12 тем:
  1. Общие сведения о LaTeX.
  2. Пишем первый документ.
  3. Добавление заголовка.
  4. Изменение шрифта.
  5. Добавление изображений.
  6. Создание списков.
  7. Математические выражения.
  8. Базовое форматирование.
  9. Оформление глав и разделов.
  10. Создание таблиц.
  11. Добавление содержания.
  12. Справочная информация.

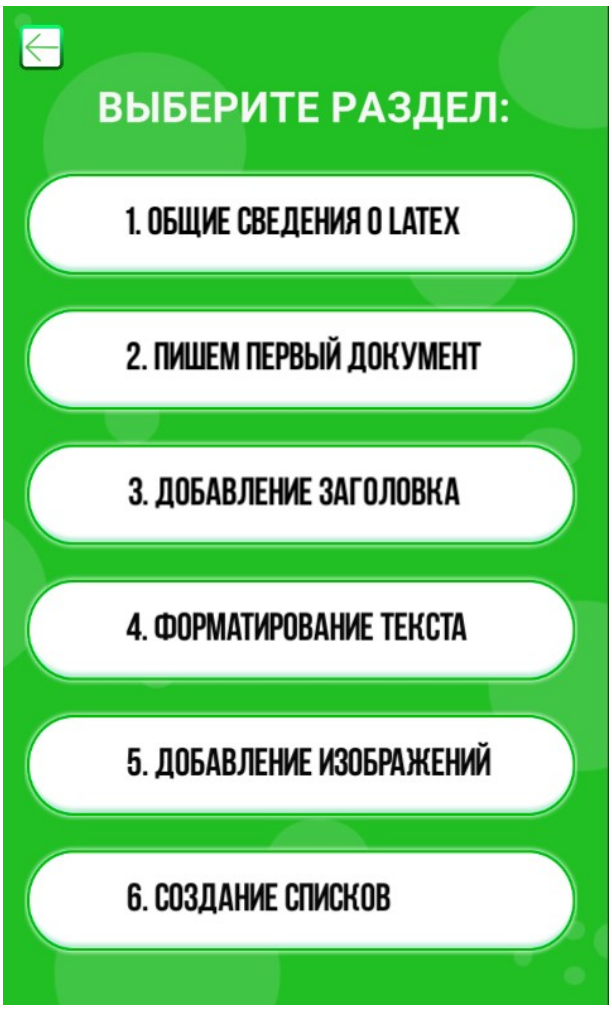

Рисунок 9 – сцена Теория

При клике на любую из тем открывается подробная теория с картинками. Визуализация темы «Создание списков» представлена на рисунке 10.

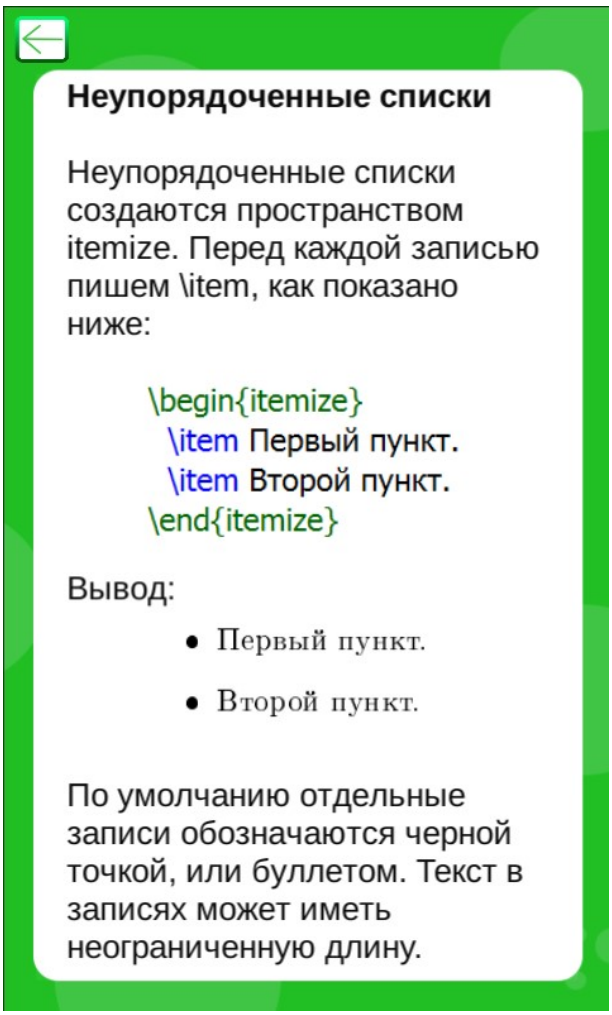

Рисунок 10 – тема «Создание списков»

- Тест. При нажатии предлагается выбрать уровень сложности:
  1. EASY – на выбор правильного ответа дается 35 секунд.
     Вопросы – легкие.
     Количество получаемых баллов за верный ответ – 10.
  2. MEDIUM – на выбор правильного ответа дается 25 секунд.
     Вопросы – средней сложности.
     Количество получаемых баллов за верный ответ – 20.
  3. HARD – на выбор правильного ответа дается 15 секунд.
     Вопросы – сложные.
     Количество получаемых баллов за верный ответ – 30.

  Далее, запускается тест с тремя вариантами ответов, визуализация показана на рисунке 11. В каждом тесте по 15 вопросов.

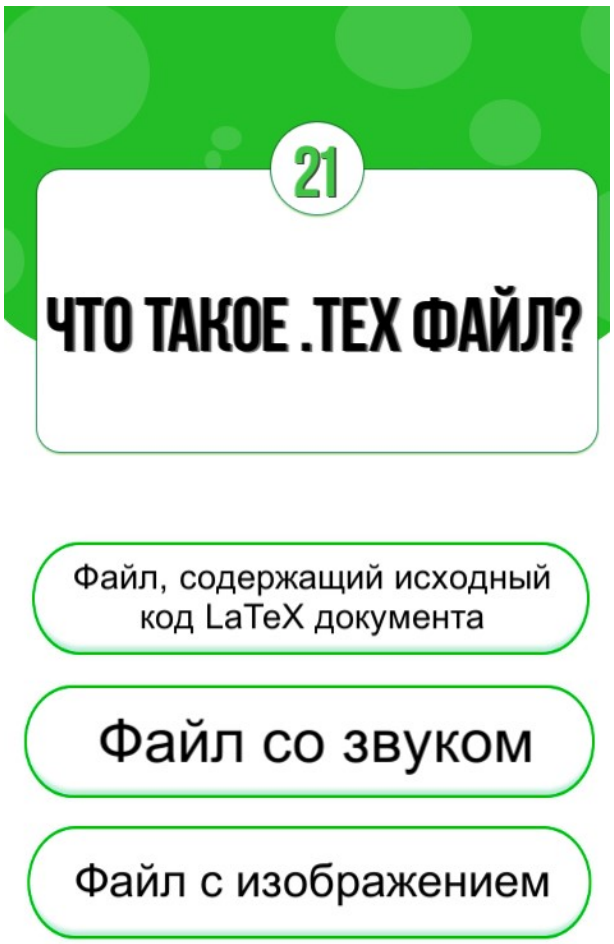

Рисунок 11 – Викторина

Основные библиотеки, используемые в приложении:

- UnityEngine – нужна для доступа к основным функциям и объектам, которые предоставляет движок Unity. Она позволяет разработчикам использовать готовые функции и компоненты, которые уже реализованы в Unity, что упрощает процесс создания игр и ускоряет разработку [8].
- UnityEngine.UI – позволяет использовать элементы пользовательского интерфейса (UI) в Unity, такие как кнопки, текстовые поля, изображения и т.д. Эта библиотека содержит классы и интерфейсы, которые позволяют управлять элементами UI и реагировать на их события.
- UnityEngine.SceneManagement – предоставляет различные методы и свойства для управления сценами в игре, такие как загрузка, сохранение, переключение между сценами и т.д. [9].
- Firebase.Auth используется для работы с сервисом аутентификации Firebase Authentication, который предоставляет различные методы для аутентификации пользователей, включая регистрацию, вход в систему, сброс пароля

и т.д. Эта библиотека позволяет разработчикам легко и удобно работать с сервисом Firebase Authentication, не беспокоясь о его реализации [10].

- Firebase.Database. Использование этой библиотеки позволяет создавать приложения, которые могут работать с базами данных без необходимости написания сложного кода. Она также предоставляет инструменты для создания, чтения, обновления и удаления данных в базе данных.

ЗАКЛЮЧЕНИЕ

В заключении можно отметить важность использования современных технологий в образовании и обучении. Unity – это мощный инструмент для разработки игр и приложений, который имеет широкий спектр применения в образовании.

Созданное приложение для изучения LaTeX, поможет упростить процесс обучения, позволит пользователям быстро освоить необходимые навыки и убедиться в том, что они с легкостью могут использовать LaTeX для работы над научными проектами и документами.

В ходе выполнения курсовой работы были изучены основные возможности и принципы работы Unity, а также созданы игровые механики и анимации. Основной дизайн спроектирован в Adobe Photoshop, что позволило более подробно изучить этот редактор. Были получены углубленные знания и практические навыки работы с языком C#. А также были применены сервисы Firebase Authentication и Firebase Realtime Database, позволяющие реализовать регистрацию пользователей, создать базу данных, настроить права доступа.

В результате написания курсовой работы было разработано обучающее приложением «LaTeX_Learn», которое может быть использовано в образовательных целях, в частности, как вспомогательный инструмент для изучения LaTeX в рамках учебной практики. Разработанное приложение на текущий момент не имеет аналогов на рынке мобильных приложений.

Таким образом, поставленная цель достигнута, задачи полностью выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 НОУ ИНТУИТ | ЛЕКЦИЯ | LATEX [Электронный ресурс] URL: https://intuit.ru/studies/courses/12708/1193/lecture/32733 (Дата обращения – 04.05.2023) – Загл. с экрана. – Яз. рус.

2 Что такое LaTeX, особенности, достоитнства, недостатки. [Электронный ресурс] URL: https://www.opennet.ru/docs/RUS/linux_base/node380.html (Дата обращения – 07.05.2023) – Загл. с экрана. – Яз. рус.

3 Unity (игровой движок) [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Unity_(игровой_движок) (Дата обращения – 07.05.2023) – Загл. с экрана. – Яз. рус.

4 UNITY FOR BEGINNERS [Электронный ресурс] URL:https://unity.com/learn/get-started (Дата обращения – 07.05.2023) – Загл. с экрана. – Яз. рус.

5 Объектно-ориентированное программирование (C#) [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/tutorials/oop (Дата обращения – 13.05.2023) – Загл. с экрана. – Яз. рус.

6 Firebase [Электронный ресурс] URL: https://firebase.google.com/ (Дата обращения – 16.05.2023) – Загл. с экрана. – Яз. англ.

7 Firebase Realtime Database [Электронный ресурс] URL: https://firebase.google.com/docs/database?hl=en (Дата обращения – 17.05.2023) – Загл. с экрана. – Яз. англ.

8 Unity Documentation [Электронный ресурс] URL: https://docs.unity3d.com/2022.1/Documentation/Manual/overview-of-dot-net-in-unity.html (Дата обращения – 20.05.2023) – Загл. с экрана. – Яз. англ.

9 HOW TO USE THE UNITY SCENEMANAGER [Электронный ресурс] URL: https://myriadgamesstudio.com/how-to-use-the-unity-scenemanager/ (Дата обращения – 20.05.2023) – Загл. с экрана. – Яз. англ.

10 Firebase Auth [Электронный ресурс] URL: https://www.rowy.io/blog/firebase-auth-rest-api (Дата обращения – 22.05.2023) – Загл. с экрана. – Яз. рус.

ПРИЛОЖЕНИЕ А
Программный код на языке C#, основной скрипт

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using Firebase.Auth;
using Firebase;
using Firebase.Database;
using TMPro;
using System.Linq;

public class GameScript : MonoBehaviour {
    #region Описание переменных
    [HideInInspector] // Позволяет не отображать переменную, которой он
назначен, в редакторе
    public QuestionsList[] Questions;
    [HideInInspector]
    public int publicTimeCount = 20;
    [HideInInspector]
    public Color trueCC, falseCC, defaultCC; // Цвет панели при ответе
    [HideInInspector]
    public int multiplierScore = 100;

    public GameObject PanelTrue;
    public GameObject PanelFalse;
    public Text questionText;
    public Button[] answerBttns = new Button[3];
    public Text[] answersText = new Text[3];
    public GameObject[] answersIcons; // 0 - trueIcon; 1 - falseIcon;
    public Image headPanel;
    public GameObject exitPanel;
```

```
public GameObject finalText;
public Text time;
public Text recordText;
public Text scoreText;
private int timeCount = 20;
private int score;
private float scoreForRecord;
private int currentQ = 1;
private bool answerClicked;
public Texture2D editorImg;
public Image bg;
private int playTime;
private bool trueColor, falseColor,defaultColor;
private int randQ;
private List<object> qList;
private QuestionsList crntQ;
private int ExitFlag = 0;
 //firebase

DatabaseReference dbRef;
FirebaseAuth auth;
int myScore;
[SerializeField] TMP_Text TextMyScore;
[SerializeField] TMP_Text TextLeaders;
#endregion

private void UpdateMyScoreText()
{
    TextMyScore.text = myScore.ToString();
}

public void ButtonLeave()
{
    PlayerPrefs.DeleteKey("MyScore");
```

```
        auth.SignOut();
    }

    /// <summary>
    ///  Вызывается один раз за кадр
    ///  Это основное событие для прорисовки кадра.
    /// </summary>
    void Update ()
    {
        myScore = PlayerPrefs.GetInt("MyScore");
        UpdateMyScoreText();

        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        auth = FirebaseAuth.DefaultInstance;


        scoreText.text = string.Format("Ваш счёт: {0:0}", score);
        scoreForRecord = PlayerPrefs.GetInt("score");
        recordText.text = string.Format("Ваш рекорд: {0:0}",
 scoreForRecord);

        // Назначение цвета панели в зависимости от ответа
        if (defaultColor)
        {
            headPanel.color = Color.Lerp(headPanel.color, defaultCC,
8 * Time.deltaTime);
        }
        else if (trueColor)
        {
            headPanel.color = Color.Lerp(headPanel.color, trueCC,
 8 * Time.deltaTime);
        }
        else if (falseColor)
        {
```

```
            headPanel.color = Color.Lerp(headPanel.color, falseCC,
8 * Time.deltaTime);
        }

        if (Input.GetKeyDown(KeyCode.Escape) && !exitPanel.activeSelf)
        {
            exitPanel.SetActive(true); Time.timeScale = 0;
        }
        else if (Input.GetKeyDown(KeyCode.Escape) &&
exitPanel.activeSelf)
        {
            exitPanel.SetActive(false); Time.timeScale = 1;
        }

    }

    /// <summary>
    /// Обработка нажатия кнопки "Тест" в главном меню
    /// </summary>
    public void playBttn(int time)
    {
        playTime = time;
        timeCount = playTime;
  // инициализируем список вопросов
        qList = new List<object>(Questions);
        generateQuestion();
        headPanel.GetComponent<Animation>().Play("HeadAnim");
        score = 0;
        myScore = 0;
        finalText.SetActive(false);
    }

    /// <summary>
    /// Генерация нового вопроса
```

```
    /// </summary>
    void generateQuestion()
    {
        if (qList.Count > 0)
        {
            if (scoreText.gameObject.activeSelf)
     scoreText.GetComponent<Anim>().Play("Bubble_Close_3");
            randQ = Random.Range(0, qList.Count);
            crntQ = qList[randQ] as QuestionsList;
            if (crntQ != null)
            {
                questionText.text = crntQ.Question;
                questionText.GetComponent<Anim>().Play("Bubble_Open_1");
                List<string> answers = new List<string>(crntQ.answers);
                for (int i = 0; i < crntQ.answers.Length; i++)
                {
                    int randA = Random.Range(0, answers.Count);
                    answersText[i].text = answers[randA];
                    answers.RemoveAt(randA);
                }
            }
            StartCoroutine(answersBttnsInAnim());
            timeCount = playTime;
            currentQ++;
        }
        else StartCoroutine(final());
    }
    public void answerBttn(int index)
    {
        answerClicked = true;
        StartCoroutine(trueOrFalse(answersText[index].text ==
crntQ.answers[0]));
    }
    IEnumerator final()
```

```
    {
        finalText.SetActive(true);
        yield return new WaitForSeconds(2);
        trueColor = false;
        defaultColor = true;
        headPanel.GetComponent<Animation>().Play("HeadAnimOut");
        scoreText.GetComponent<Animation>().Play("Bubble_Close_3");
        finalText.GetComponent<Animation>().Play("Bubble_Close_3");
        if (score > PlayerPrefs.GetInt("score"))
 PlayerPrefs.SetInt("score", score);
    }
    IEnumerator timer()
    {
        answerClicked = false;
        if (!time.gameObject.activeSelf)
time.gameObject.SetActive(true);
        else time.GetComponent<Animation>().Play("Bubble_Open_3");
        while (timeCount > -1)
        {
            if (!answerClicked)
            {
                time.text = timeCount.ToString();
                timeCount--;
                yield return new WaitForSeconds(1);
            }
            else yield break;
        }
        foreach (Button t in answerBttns) t.interactable = false;
        if (!answerClicked) StartCoroutine(timeOut());
    }
    IEnumerator answersBttnsInAnim()
    {
        foreach (Button t in answerBttns)
        {
```

```
            t.interactable = false;
        }
        int i = 0;
        yield return new WaitForSeconds(1);
        while (i < answerBttns.Length)
        {
            if (!answerBttns[i].gameObject.activeSelf)
            {
                answerBttns[i].gameObject.SetActive(true);
            }
            else
            {
                answerBttns[i].GetComponent<Anim>().Play("Bubble_Open");
            }

            i++;
            yield return new WaitForSeconds(1);
        }
        foreach (Button t in answerBttns) t.interactable = true;
        yield return StartCoroutine(timer());
    }
    IEnumerator timeOut()
    {
        foreach (Button t in answerBttns)
        {
            t.GetComponent<Animation>().Play("Bubble_Close_2");
        }
        falseColor = true;
        PanelFalse.SetActive(true);
        yield return new WaitForSeconds(0.5f);
        if (!answersIcons[2].activeSelf)
        {
            answersIcons[2].SetActive(true);
        }
```

```
        else
        {
            answersIcons[2].GetComponent<Anim>().Play("Bubble_Open_3");
        }
        questionText.GetComponent<Anim>().Play("Bubble_Close_1");
        yield return new WaitForSeconds(0.5f);
        if (!scoreText.gameObject.activeSelf)
        {
            scoreText.gameObject.SetActive(true);
        }
        else
        {
            scoreText.GetComponent<Anim>().Play("Bubble_Open_3");
        }
        yield return new WaitForSeconds(2);
        answersIcons[2].GetComponent<Anim>().Play("Bubble_Close_3");
        time.GetComponent<Anim>().Play("Bubble_Close_3");
        falseColor = false;
        defaultColor = true;
        PanelFalse.SetActive(false);
        headPanel.GetComponent<Anim>().Play("HeadAnimOut");
        if (score > PlayerPrefs.GetInt("score"))
        {
            PlayerPrefs.SetInt("score", score);
        }
    }
    /// <summary>
    /// BoardLeader
    /// </summary>
    /// <returns></returns>
    public IEnumerator GetLeaders()
    {
        var leaders = dbRef.Child("LeaderBoard").OrderByChild("MyScore")
```

```
        yield return new WaitUntil(predicate: () => leaders.IsCompleted);

        if (leaders.Exception != null)
        {
            Debug.LogError("ERROR: " + leaders.Exception);
        }
        else if (leaders.Result.Value == null)
        {
            Debug.LogError("Result.Value == null");
        }
        else
        {
            DataSnapshot snapshot = leaders.Result;

            int num = 1;
            foreach
(DataSnapshot dataChildSnapshot in snapshot.Children())
            {
                TextLeaders.text += "\n" + num + ". " +
dataChildSnapshot.Child("Email").Value.ToString() + " : " +
                    dataChildSnapshot.Child("MyScore").Value.ToString();
                num++;
            }
        }
    }

    public void openLeaderBoard()
    {
        StartCoroutine(GetLeaders());
    }



    public void BackButton()
    {
```

```
        TextLeaders.text = "";
    }


    /// <summary>
    /// Проверка правильный ли ответ был в тесте
    /// </summary>
    /// <param name="check"></param>
    /// <returns></returns>
    IEnumerator trueOrFalse(bool check)
    {
        defaultColor = false;
        foreach (Button t in answerBttns)
        {
            t.interactable = false;
        }
        yield return new WaitForSeconds(1);
        if (check)
        {
            //score = score + (multiplierScore * currentQ) +
 (timeCount * multiplierScore);
            score = score + 10; // Количество баллов


            //Таблица лидеров
            myScore = myScore + 10;
            UpdateMyScoreText();
            PlayerPrefs.SetInt("MyScore", myScore);

            dbRef.Child("LeaderBoard").Child(auth.CurrentUser.Email.
Replace(".", "")).Child("MyScore").SetValueAsync(myScore);
            dbRef.Child("LeaderBoard").Child(auth.CurrentUser.Email.
Replace(".", "").Child("Email").SetValueAsync(auth.CurrentUser.Email);
```

```
foreach (Button t in answerBttns)
{
    t.GetComponent<Animation>().Play("Bubble_Close_2");
}
trueColor = true;
PanelTrue.SetActive(true);
yield return new WaitForSeconds(0.5f);
if (!answersIcons[0].activeSelf)
{
    answersIcons[0].SetActive(true);
}
else
{
    answersIcons[0].GetComponent<Anim>().Play("Bubble_Open")
}
questionText.GetComponent<Anim>().Play("Bubble_Close_1");
yield return new WaitForSeconds(0.5f);
time.GetComponent<Anim>().Play("Bubble_Close_3");
qList.RemoveAt(randQ); // удаляем текущий вопрос
if (!scoreText.gameObject.activeSelf)
{
    scoreText.gameObject.SetActive(true);
}
else
{
    scoreText.GetComponent<Anim>().Play("Bubble_Open_3");
}
yield return new WaitForSeconds(1);
answersIcons[0].GetComponent<Anim>().Play("Bubble_Close_3");
trueColor = false;
defaultColor = true;
PanelTrue.SetActive(false);
generateQuestion(); // генерация нового вопроса
```

```
        }
        else
        {
            foreach (Button t in answerBttns) t.GetComponent<Anim>()
Play("Bubble_Close_2");
falseColor = true;
 PanelFalse.SetActive(true);
yield return new WaitForSeconds(0.5f);
if (!answersIcons[1].activeSelf)
answersIcons[1].SetActive(true);
 else answersIcons[1].GetComponent<Anim>().Play("Bubble_Open_3");
questionText.GetComponent<Anim>().Play("Bubble_Close_1");
yield return new WaitForSeconds(0.5f);
if (!scoreText.gameObject.activeSelf) scoreText.gameObject.SetActive(tru
else
 scoreText.GetComponent<Animation>().Play("Bubble_Open_3");
yield return new WaitForSeconds(1);
answersIcons[1].GetComponent<Animation>().Play("Bubble_Close_3");
time.GetComponent<Animation>().Play("Bubble_Close_3");
falseColor = false;
defaultColor = true;
PanelFalse.SetActive(false);
headPanel.GetComponent<Anim>().Play("HeadAnimOut");
scoreText.GetComponent<Anim>().Play("Bubble_Close_3");
if (score > PlayerPrefs.GetInt("score"))
 PlayerPrefs.SetInt("score", score);
yield return new WaitForSeconds(1.5f);
scoreText.gameObject.SetActive(false);
        }
    }

    /// <summary>
    /// Обработка выхожа когда идет тест
    /// выход на главный экран
```

```
    /// 0 - да
    /// 1 - нет
    /// </summary>
    /// <param name="bttn"></param>
    public void exitPan(int bttn)
    {
        if (ExitFlag == 1) // нужно выйти на окно регистрации?
        {
            if (bttn == 0)
            {
                SceneManager.LoadScene("LogIn");
            }
            else { exitPanel.SetActive(false); Time.timeScale = 1; }
            ExitFlag = 0;
        }
        else
        {
            if (bttn == 0)
            {
                if (score > PlayerPrefs.GetInt("score"))
PlayerPrefs.SetInt("score", score);
                Application.Quit();
            }
            else { exitPanel.SetActive(false); Time.timeScale = 1; }
        }

    }

    /// <summary>
    /// Обработка выхода из игры на главном экране
    /// переходим на окно регистрации
    /// </summary>
    public void Exitgame()
    {
```

```
        exitPanel.SetActive(true);
        ExitFlag = 1;
    }
}

[System.Serializable]
public class QuestionsList
{
    public string Question;
    public string[] answers = new string[3];
}
```

ПРИЛОЖЕНИЕ Б

Программный код на языке C#, скрипт OpenPanel

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System;


public class OpenPanels : MonoBehaviour
{
    public Button[] sectionTeoryBttns = new Button[3];
    //public Text[] sectionTeoryText = new Text[2];
    public Sprite[] sectionTeoryImage = new Sprite[3];
    public Text TeoryText; // текст на панели
    public Image TeoryImage;// изображение на панели

    private bool sectionTeoryClicked1 = false; // 1 раздел теории
    private bool sectionTeoryClicked2 = false; // 2 раздел теории
    private bool sectionTeoryClicked3 = false; // 3 раздел теории
    private bool sectionTeoryClicked4 = false; // 4 раздел теории
    private bool sectionTeoryClicked5 = false; // 5 раздел теории
    private bool sectionTeoryClickedIstoch = false;
    Animator openedPanel;

    /// <summary>
    /// Открытие панели с текстом теории при выборе раздела
    /// </summary>
    /// <param name="anim"></param>
    public void OpenPanel(Animator anim)
    {
        anim.gameObject.SetActive(true);
```

```
        // Анимация
        anim.SetTrigger("open");
        openedPanel = anim;

        if (sectionTeoryClicked1) // Если выбран раздел 1
        {
            StreamReader sr = new StreamReader
(Directory.GetCurrentDirectory() + "\\Теория\\Тема 1.txt");
            TeoryImage.enabled = false; // Выключаем изображение
            string line = "";
            TeoryText.text = "";
            while ((line = sr.ReadLine()) != null)
            {
                TeoryText.text += line + Environment.NewLine;
            }

            sectionTeoryClicked1 = false;
        }

        if (sectionTeoryClicked2) // Если выбран раздел 2
        {

            StreamReader sr = new StreamReader
(Directory.GetCurrentDirectory() + "\\Теория\\Тема 2.txt");

            // Заполнение текста
            string line = "";
            TeoryText.text = "";
            while ((line = sr.ReadLine()) != null)
            {
                TeoryText.text += line + Environment.NewLine;
            }
```

```
            TeoryImage.enabled = true; // Включаем изображение
            TeoryImage.sprite = sectionTeoryImage[1];

            sectionTeoryClicked2 = false;
        }

        if (sectionTeoryClicked3) // Если выбран раздел 3
        {
            StreamReader sr = new StreamReader
(Directory.GetCurrentDirectory() + "\\Теория\\Тема 3.txt");
            TeoryImage.enabled = false; // Выключаем изображение
            string line = "";
            TeoryText.text = "";
            while ((line = sr.ReadLine()) != null)
            {
                TeoryText.text += line + Environment.NewLine;
            }


            TeoryImage.enabled = true; // Включаем изображение
            TeoryImage.sprite = sectionTeoryImage[2];

            sectionTeoryClicked3 = false;
        }

        if (sectionTeoryClicked4) // Если выбран раздел 4
        {
            StreamReader sr = new StreamReader
(Directory.GetCurrentDirectory() + "\\Теория\\Тема 4.txt");

            string line = "";
            TeoryText.text = "";
            while ((line = sr.ReadLine()) != null)
            {
```

```
                TeoryText.text += line + Environment.NewLine;
            }

            TeoryImage.enabled = true; // Включаем изображение
            TeoryImage.sprite = sectionTeoryImage[3];
            sectionTeoryClicked4 = false;
        }

        if (sectionTeoryClicked5) // Если выбран раздел 5
        {
            StreamReader sr = new StreamReader
(Directory.GetCurrentDirectory() + "\\Теория\\Тема 5.txt");
            TeoryImage.enabled = false; // Выключаем изображение
            string line = "";
            TeoryText.text = "";
            while ((line = sr.ReadLine()) != null)
            {
                TeoryText.text += line + Environment.NewLine;
            }

            sectionTeoryClicked5 = false;
        }

        if (sectionTeoryClickedIstoch) // Если выбран раздел источники
        {
            StreamReader sr = new StreamReader
(Directory.GetCurrentDirectory() + "\\Теория\\Источники.txt");
            TeoryImage.enabled = false; // Выключаем изображение
            string line = "";
            TeoryText.text = "";
            while ((line = sr.ReadLine()) != null)
            {
                TeoryText.text += line + Environment.NewLine;
            }
```

```
        sectionTeoryClickedIstoch = false;
    }
}

/// <summary>
/// Закрытие панель с текстом теории при нажатии кнопки "Назад"
/// </summary>
public void ClosePanel()
{
    openedPanel.SetTrigger("close");
}

/// <summary>
/// Обработка нажатия на кнопку выбора раздела темы 1
/// </summary>
public void sectionBttn1()
{
    sectionTeoryClicked1 = true;
}

/// <summary>
/// Обработка нажатия на кнопку выбора раздела темы 2
/// </summary>
public void sectionBttn2()
{
    sectionTeoryClicked2 = true;
}

/// <summary>
/// Обработка нажатия на кнопку выбора раздела темы 3
/// </summary>
public void sectionBttn3()
{
```

```
        sectionTeoryClicked3 = true;
    }

    /// <summary>
    /// Обработка нажатия на кнопку выбора раздела темы 4
    /// </summary>
    public void sectionBttn4()
    {
        sectionTeoryClicked4 = true;
    }

    /// <summary>
    /// Обработка нажатия на кнопку выбора раздела темы 5
    /// </summary>
    public void sectionBttn5()
    {
        sectionTeoryClicked5 = true;
    }

    /// <summary>
    /// Обработка нажатия на кнопку источники
    /// </summary>
    public void sectionBttnIstochn()
    {
        sectionTeoryClickedIstoch = true;
    }
}
```

ПРИЛОЖЕНИЕ В

Программный код на языке C#, скрипт Scenes

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Scenes : MonoBehaviour
{
    [Header("Loading Panel")]
    [SerializeField] private Animator loadPanel;
    [SerializeField] private float animationDuration = 0.20f;
    [SerializeField] private bool playAnimationInStart;
    int currentScene;

    private void Start()
    {
        if (playAnimationInStart)
        {
            loadPanel.gameObject.SetActive(true);
            loadPanel.SetTrigger("end");
        }
    }

    /// <summary>
    ///  Переход на новую сцену
    /// </summary>
    /// <param name="sceneid"></param>
    public void NextLevel(int sceneid)
    {
        currentScene = sceneid;
        StartCoroutine(LoadCurrentScene());
    }
```

```
    IEnumerator LoadCurrentScene()
    {
        loadPanel.gameObject.SetActive(true); // включить панельку

        loadPanel.SetTrigger("start"); // начать анимацию

        // yield определяет возвращаемый элемент
        yield return new WaitForSeconds(animationDuration);

        SceneManager.LoadScene(currentScene); // начать загрузку сцены
    }
}
```

ПРИЛОЖЕНИЕ Г

Программный код на языке C#, скрипт AuthManager

```
using Firebase.Auth;
using Firebase;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Authmanager : MonoBehaviour
{
    //Firebase variables
    [Header("Firebase")]
    public DependencyStatus dependencyStatus;
    public FirebaseAuth auth;
    public FirebaseUser User;

    //Login variables
    [Header("Login")]
    public TMP_InputField emailLoginField;
    public TMP_InputField passwordLoginField;
    public TMP_Text warningLoginText;
    public TMP_Text confirmLoginText;

    //Register variables
    [Header("Register")]
    public TMP_InputField usernameRegisterField;
    public TMP_InputField emailRegisterField;
    public TMP_InputField passwordRegisterField;
    public TMP_InputField passwordRegisterVerifyField;
    public TMP_Text warningRegisterText;
```

```
    void Awake()
    {
        FirebaseApp.CheckAndFixDependenciesAsync().ContinueWith(task =>
        {
            dependencyStatus = task.Result;
            if (dependencyStatus == DependencyStatus.Available)
            {
                //если все верно подключаемся
                InitializeFirebase();
            }
            else
            {
                Debug.LogError("Could not resolve all Firebase
dependencies: " + dependencyStatus);
            }
        });
    }

    private void InitializeFirebase()
    {
        Debug.Log("Setting up Firebase Auth");
        //Установим объект экземпляра аутентификации
        auth = FirebaseAuth.DefaultInstance;
    }

    /// <summary>
    /// Обработка кнопки входа
    /// </summary>
    public void LoginButton()
    {
StartCoroutine(Login(emailLoginField.text, passwordLoginField.text));
    }

    /// <summary>
```

```
    /// Обработка кнопки регистрации
    /// </summary>
    public void RegisterButton()
    {
        StartCoroutine(Register(emailRegisterField.text,
passwordRegisterField.text, usernameRegisterField.text));
    }

    private IEnumerator Login(string _email, string _password)
    {
        var LoginTask = auth.SignInWithEmailAndPasswordAsync
(_email, _password);
        //вывод в консоль
        yield return new WaitUntil(predicate: () =>
 LoginTask.IsCompleted);

        if (LoginTask.Exception != null)
        {
            //Обработка ошибок, если они есть
            Debug.LogWarning(message: $"Failed to register task with
{LoginTask.Exception}");
            FirebaseException firebaseEx =
LoginTask.Exception.GetBaseException() as FirebaseException;
            AuthError errorCode = (AuthError)firebaseEx.ErrorCode;

            string message = "Ошибка Входа!";
            switch (errorCode)
            {
                case AuthError.MissingEmail:
                    message = "Введите адрес эл. почты";
                    break;
                case AuthError.MissingPassword:
                    message = "Введите пароль";
                    break;
```

```
                case AuthError.WrongPassword:
                    message = "Неверный пароль";
                    break;
                case AuthError.InvalidEmail:
                    message = "Неверный адрес эл. почты";
                    break;
                case AuthError.UserNotFound:
                    message = "Учетная запись не существует";
                    break;
            }
            warningLoginText.text = message;
        }
        else
        {
            //пользователь вошел в систему
            //get the result
            User = LoginTask.Result;
            Debug.LogFormat("Пользователь успешно вошел в систему: {0}
 ({1})", User.DisplayName, User.Email);
            warningLoginText.text = "";
            confirmLoginText.text = "Logged In";
            SceneManager.LoadScene("Game");
        }
    }

    private IEnumerator Register(string _email, string _password,
string _username)
    {
        if (_username == "")
        {
            warningRegisterText.text = "Не указано имя пользователя";
        }
        else
if (passwordRegisterField.text != passwordRegisterVerifyField.text)
```

```
        {
            //Если пароль не совпадает, отобразится предупреждение
            warningRegisterText.text = "Пароль не совпадает!";
        }
        else
        {
            var RegisterTask = auth.CreateUserWithEmailAndPasswordAsync
(_email, _password);
            yield return new WaitUntil(predicate: () =>
 RegisterTask.IsCompleted);

            if (RegisterTask.Exception != null)
            {
                Debug.LogWarning(message: $"Failed to register task
with {RegisterTask.Exception}");
                FirebaseException firebaseEx =
RegisterTask.Exception.GetBaseException() as FirebaseException;
                AuthError errorCode = (AuthError)firebaseEx.ErrorCode;

                string message = "Ошибка регистрации!";
                switch (errorCode)
                {
                    case AuthError.MissingEmail:
                        message = "Не указана эл. почта";
                        break;
                    case AuthError.MissingPassword:
                        message = "Не указан пароль";
                        break;
                    case AuthError.WeakPassword:
                        message = "Слишком простой пароль";
                        break;
                    case AuthError.EmailAlreadyInUse:
                        message = "Пользователь с такой эл. почтой
уже существует";
```

```
                    break;
                }
                warningRegisterText.text = message;
            }
            else
            {
                // пользователь создан
                User = RegisterTask.Result;

                if (User != null)
                {
                    UserProfile profile = new UserProfile { DisplayName
= _username };

                    var ProfileTask = User.UpdateUserProfileAsync
(profile);
                    yield return new WaitUntil(predicate: () =>
ProfileTask.IsCompleted);

                    User.SendEmailVerificationAsync().ContinueWith(task
 =>                 {
                        if (task.IsCanceled)
                        {
                            Debug.LogError
("SendEmailVerificationAsync was canceled.");
                            return;
                        }
                        if (task.IsFaulted)
                        {
                            Debug.LogError("SendEmailVerificationAsync
encountered an error: "+ task.Exception);
                            return;
                        }
```

```
                        Debug.Log("Email sent successfully.");
                    });


                    else
                    {
                        //Имя пользователя теперь установлено
                        //Теперь вернитесь к экрану входа в систему
                        UIManager.instance.LoginScreen();
                        warningRegisterText.text = "";
                    }
                }
            }
        }

    }
}
```

ПРИЛОЖЕНИЕ Д

Программный код на языке C#, скрипт DB

```
using TMPro;
using UnityEngine;
using UnityEngine.UI;
using Firebase.Database;
using System.Collections;
using Firebase.Auth;
using UnityEngine.SceneManagement;

public class DB : MonoBehaviour
{
    public TMP_Text text; // вывод текста из бд
    DatabaseReference dbRef;
    FirebaseAuth auth;

    public TMP_InputField email; // почта, введенная пользователем
    public TMP_InputField password; // пароль,введенный пользователем
    public TMP_Text InfoText; // вывод доп. информации при регистрации

    void Start()
    {
        dbRef = FirebaseDatabase.DefaultInstance.RootReference;
        auth = FirebaseAuth.DefaultInstance;
        auth.StateChanged += Auth_StateChanged;
        Auth_StateChanged(this, null);
        auth.SignOut(); // чтобы не было автоматического входа
    }

    private void Auth_StateChanged(object sender, System.EventArgs e)
    {
        if(auth.CurrentUser != null)
        {
            InfoText.text = "Вход выполнен " + auth.CurrentUser.Email;
```

```
            SceneManager.LoadScene("Game");
        }
        else
        {
            InfoText.text = "Вы уверены, что Email и password
указаны верно?";
        }
    }

    /// <summary>
    /// Вход
    /// </summary>
    public void LoginButton()
    {
        auth.SignInWithEmailAndPasswordAsync(email.text, password.text);
    }

    /// <summary>
    /// Регистрация
    /// </summary>
    public void SignInButton()
    {
        auth.CreateUserWithEmailAndPasswordAsync
(email.text, password.text);
    }

    #region Add_Load_Remove_Firebase
    /// <summary>
    /// сохранение данных в бд
    /// </summary>
    /// <param name="str"></param>
    public void SaveData(string name)
    {
        User user = new User(name, 20, "offline");
```

```
        string json = JsonUtility.ToJson(user);
        dbRef.Child("users").Child(name).SetRawJsonValueAsync(json);
    }

    /// <summary>
    /// Считывание данных из бд
    /// </summary>
    /// <param name="str"></param>
    public IEnumerator LoadData(string str)
    {
        var user = dbRef.Child("users").Child(str).GetValueAsync();

        yield return new WaitUntil(predicate: () => user.IsCompleted);
        // проверка на ошибки
        if (user.Exception != null)
        {
            Debug.LogError(user.Exception);
        }
        else if (user.Result == null) // нет данных о user
        {
            Debug.Log("Null");
        }
        else
        {
            DataSnapshot snapshot = user.Result;
            Debug.Log(snapshot.Child("age").Value.ToString()
+ snapshot.Child("name").Value.ToString());
            text.text = snapshot.Child("age").Value.ToString();
        }
    }

    /// <summary>
    /// Удаленние данных из бд
    /// </summary>
```

```
    public void RemoveData(string str)
    {
        dbRef.Child("users").Child(str).RemoveValueAsync();
    }
    #endregion Add_Load_Remove_Firebase
}

public class User
{
    public string name;
    public int age;
    public string status;

    public User (string name, int age, string status)
    {
        this.name = name;
        this.age = age;
        this.status = status;
    }
}
```